SAS® Data Integration Studio 4.9

User's Guide



SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2014. *SAS® Data Integration Studio 4.9: User's Guide*. Cary, NC: SAS Institute Inc.

SAS® Data Integration Studio 4.9: User's Guide

Copyright © 2014, SAS Institute Inc., Cary, NC, USA

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.

The scanning, uploading, and distribution of this book via the Internet or any other means without the permission of the publisher is illegal and punishable by law. Please purchase only authorized electronic editions and do not participate in or encourage electronic piracy of copyrighted materials. Your support of others' rights is appreciated.

U.S. Government License Rights; Restricted Rights: The Software and its documentation is commercial computer software developed at private expense and is provided with RESTRICTED RIGHTS to the United States Government. Use, duplication or disclosure of the Software by the United States Government is subject to the license terms of this Agreement pursuant to, as applicable, FAR 12.212, DFAR 227.7202-1(a), DFAR 227.7202-3(a) and DFAR 227.7202-4 and, to the extent required under U.S. federal law, the minimum restricted rights as set out in FAR 52.227-19 (DEC 2007). If FAR 52.227-19 is applicable, this provision serves as notice under clause (c) thereof and no other notice is required to be affixed to the Software or documentation. The Government's rights in Software and documentation shall be only those set forth in this Agreement.

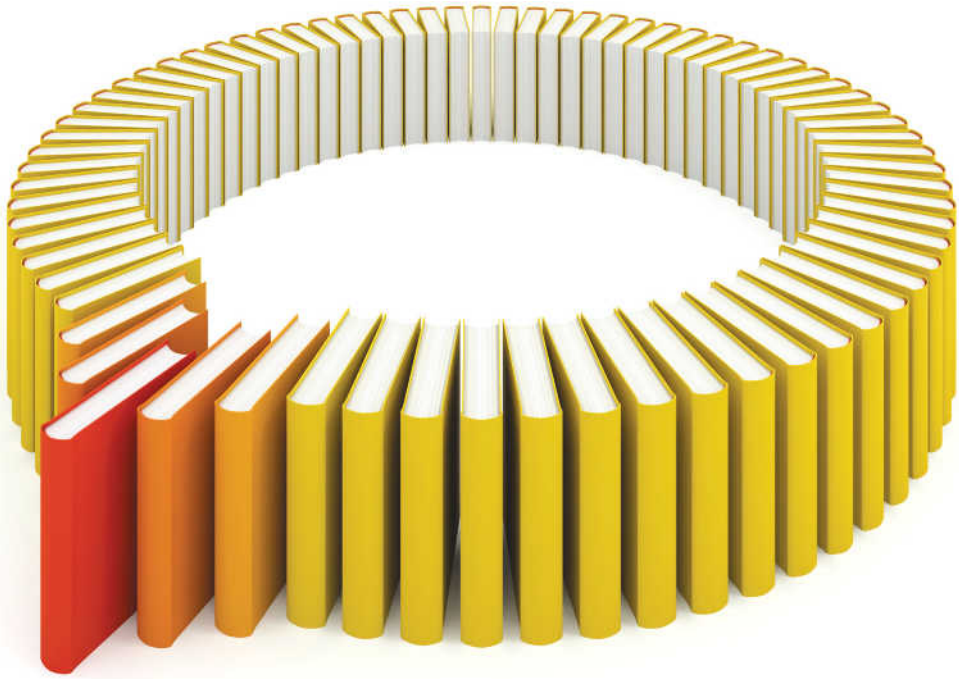
SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513-2414.

August 2014

SAS provides a complete selection of books and electronic products to help customers use SAS® software to its fullest potential. For more information about our offerings, visit **support.sas.com/bookstore** 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 trademarks of their respective companies.



Gain Greater Insight into Your SAS® Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.



support.sas.com/bookstore
for additional books and resources.



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 trademarks of their respective companies. © 2013 SAS Institute Inc. All rights reserved. S107969US.0613

Contents

What's New in SAS Data Integration Studio 4.9 . . . . . *xiii*

PART 1 Introduction 1

Chapter 1 • Overview of SAS Data Integration **3**
About SAS Data Integration 3
Advantages of SAS Data Integration 4
A Basic Data Integration Environment 4
How to Get Help for SAS Data Integration Studio 8
Installing SAS Data Integration Studio 8
Administrative Documentation for SAS Data Integration Studio 9
Accessibility Features in SAS Data Integration Studio 10

PART 2 General User Tasks 15

Chapter 2 • Getting Started **17**
Security for SAS Data Integration Studio 19
Main Tasks for Creating Process Flows 19
Starting SAS Data Integration Studio 20
Connecting to a SAS Metadata Server 22
Working with the Folders Tree 24
Selecting a Default SAS Application Server 27
Registering SAS Libraries 28
Working with User-Defined Formats 28
Registering Tables and Cubes 29
Overview of Transformations 32
Working with Stored Processes 43
Working with Web Services 46
Specifying Global Options in SAS Data Integration Studio 46
Working with Change Management 47
Search Metadata 52
Add a Note or Document to a Registered Object 55
View the Content of Notes or Documents 57

Chapter 3 • Importing, Exporting, and Copying Metadata **59**
Metadata Import and Export in SAS Data Integration Studio 60
Working with SAS Package Metadata 60
Preparing to Import or Export SAS Package Metadata 61
Exporting SAS Package Metadata 62
Importing SAS Package Metadata 63
Copying and Pasting Metadata Objects 65
Working with SAS Metadata Bridges 66
Usage Notes for Importing or Exporting with a SAS Metadata Bridge 66
Preparing to Import or Export with a SAS Metadata Bridge 67
Importing New Metadata with a SAS Metadata Bridge 68

Importing Updated Metadata with a SAS Metadata Bridge 70
Exporting Metadata with a SAS Metadata Bridge 75

Chapter 4 • Working with Tables **79**
About Tables 80
Registering Existing Tables with the Register Tables Wizard 81
Registering New Tables with the New Table Wizard 82
Viewing or Updating Table Metadata 84
Using a Physical Table to Update Table Metadata 85
Specifying Options for Tables 86
Supporting Case and Special Characters in Table and Column Names 88
Maintaining Column Metadata 92
Standardizing Columns 98
Maintaining Keys 104
Maintaining Indexes 109
Browsing Table Data 111
Editing SAS Table Data 114
Using the View Data Window to Create a SAS Table 117
Specifying Browse and Edit Options for Tables and External Files 118

Chapter 5 • Working with External Files **119**
About External Files 120
Registering a Delimited External File 120
Registering a Fixed-Width External File 124
Registering an External File with User-Written Code 128
Viewing or Updating External File Metadata 131
Overriding the Code Generated by the External File Wizards 132
Specifying NLS Support for External Files 133
Accessing an External File with an FTP Server or an HTTP Server 133
Viewing Data in External Files 135
Registering a COBOL Data File That Uses a COBOL Copybook 136
Using an External File in the Process Flow for a Job 138
Using a Format File to Register a Fixed-Width External File 141

Chapter 6 • Creating Jobs **145**
About Jobs 146
Creating an Empty Job 147
Creating a Process Flow for a Job 148
Creating a Job That Contains Jobs 150
Working with Default Temporary Output Tables 150
Specifying Options for Jobs 155
Documenting Process Flow Diagrams 155
Accessing Local and Remote Data 156
Viewing or Updating Job Metadata 158
Displaying the SAS Code for a Job 160
Common Code Generated for a Job 160

Chapter 7 • Managing Jobs **165**
About Managing Jobs 166
Submitting a Job for Immediate Execution 166
Meeting Prerequisites for Collecting Job Statistics 169
Reviewing a Successful Job 170
Diagnosing and Correcting an Unsuccessful Job 175
Adding a Transformation to an Existing Job 180
Understanding the Job Has Changed Warning 182
Understanding the Crossed Versions in a Job Warning 183

Displaying Run-Time Statistics in SAS Job Monitor . . . 184
Displaying Run-Time Statistics in SAS Web Report Studio or the SAS Stored Process Server . . . 184
Maintaining Column Mappings . . . 185
Managing the Scope of Column Changes in Jobs . . . 189
Managing Connections in Job Editor Windows . . . 193
Viewing the Code for a Transformation . . . 195
Specifying Options for Transformations . . . 196
Redirecting Temporary Output Tables . . . 196
Pushing ELT Job Code Down to a Database . . . 198
Using a Web Client to Orchestrate Jobs . . . 199

Chapter 8 • Restarting Jobs From Checkpoints . . . **201**
About Restarting Jobs . . . 201
Prerequisites for Restarting Jobs . . . 202
Adding Checkpoints to a Job . . . 202
Restarting a Job . . . 204

Chapter 9 • Managing the Status of Jobs and Transformations . . . **209**
About Status Handling for Jobs and Transformations . . . 209
Default Conditions, Actions, and Conditional Action Sets . . . 210
Prerequisites for Actions . . . 215
Perform Actions Based on the Status of a Job . . . 216
Perform Actions Based on the Status of a Transformation . . . 217
Macro Variables for Status Handling . . . 219

Chapter 10 • Deploying Jobs . . . **225**
About Deploying Jobs . . . 226
About Deploying Jobs for Scheduling . . . 227
Prerequisites for Deploying a Job for Scheduling . . . 227
Deploying Jobs for Scheduling . . . 227
Using a Command Line to Deploy Jobs . . . 229
Redeploying Jobs for Scheduling . . . 233
Using Scheduling to Handle Complex Process Flows . . . 235
Using Deploy for Scheduling to Execute Jobs on a Remote Host . . . 236
About Deploying Jobs as Stored Processes . . . 237
Prerequisites for Deploying a Job as a Stored Process . . . 237
Deploying Jobs as Stored Processes . . . 238
Redeploying Jobs to Stored Processes . . . 241
Viewing or Updating Stored Process Metadata . . . 242
About Deploying Jobs as Web Services . . . 243
Prerequisites for Web Service Jobs . . . 243
Requirements for Web Service Jobs . . . 244
Creating a Web Service Job . . . 244
Deploying a Web Service Job as a Stored Process . . . 249
Deploying a Stored Process as a Web Service . . . 251

Chapter 11 • Working with Versions . . . **253**
About Versions . . . 253
Prerequisites for Version Control . . . 254
Example Setup for an Apache Subversion (SVN) Server . . . 255
Creating a Version . . . 257
Reviewing and Managing Versions . . . 258
Comparing Versions . . . 260

Chapter 12 • Working with Generated Code . . . **263**

About Code Generated for Jobs . . . 263
Displaying the Code Generated for a Job . . . 267
Displaying the Code Generated for a Transformation . . . 268
Specifying Options for Jobs . . . 268
Specifying Options for a Transformation . . . 269
Modifying Configuration Files or SAS Start Commands for Application Servers . . . 269

Chapter 13 • Working with User-Written Code . . . **271**
About User-Written Code . . . 271
Adding User-Written Code to the Precode and Postcode Tab . . . 272
Adding a User Written Code Transformation to a Job . . . 274
Creating and Using a Generated Transformation . . . 277
Updating a Generated Transformation . . . 284
Editing the Generated Code for a Job or Transformation . . . 286
Replacing the Generated Code for a Job or Transformation . . . 287
Converting a SAS Code File to a Job . . . 288

Chapter 14 • Optimizing Process Flows . . . **293**
About Process Flow Optimization . . . 293
Managing Process Data . . . 294
Managing Columns . . . 297
Streamlining Process Flow Components . . . 299
Using Simple Debugging Techniques . . . 301
Using SAS Logs . . . 304
Reviewing Temporary Output Tables . . . 306
Additional Performance Optimization Information . . . 308

Chapter 15 • Working with Impact Analysis and Data Lineage . . . **311**
Impact Analysis and Data Lineage . . . 311
Performing an Impact Analysis . . . 313
Performing Impact Analysis on a Generated Transformation . . . 316
Performing Reverse Impact Analysis . . . 317
Using SAS Lineage . . . 318

Chapter 16 • Working with Reports . . . **321**
About Metadata Reports . . . 322
Opening the Reports Window . . . 322
Selecting the Reports Perspective . . . 323
Customizing the Tables Report . . . 324
Customizing the Job Documentation Report . . . 325
Running and Saving a Report . . . 326
Saving a Report As a Document Object . . . 328
Viewing a Report . . . 329
Creating Your Own Report . . . 331

Chapter 17 • Working with Data Management Platform . . . **333**
Integration with DataFlux Data Management Platform . . . 334
General Prerequisites for Data Quality Transformations . . . 336
Prerequisites for Running a DataFlux Job or Profile in a SAS Data Integration Studio Job . . . 338
Analyzing the Quality of Data Sources . . . 341
Standardizing Values with a Standardization Scheme . . . 341
Standardizing Values with a Definition . . . 346
Using Match Codes to Improve Record Matching . . . 347
Using a DataFlux Data Service in a SAS Data Integration Studio Job . . . 351
Using a DataFlux Job or Profile in a SAS Data Integration Studio Job . . . 355

PART 3 Working with Transformations 359

Chapter 18 • Working with Analysis Transformations 361
About Analysis Transformations 362
Creating a Correlation Analysis 362
Creating a Distribution Analysis 370
Generating Forecasts 377
Frequency of Eye Color By Hair Color Crosstabulation 385
One-Way Frequency of Eye Color By Region 398
Creating Summary Statistics for a Table 407
Creating a Summary Tables Report from Table Data 413

Chapter 19 • Working with Loader Transformations 421
About Loader Transformations 421
About the SPD Server Table Loader Transformation 422
Teradata Table Loader Transformation 423
About the Table Loader Transformation 424
About the Oracle Bulk Table Loader Transformation 425
About the DB2 Bulk Table Loader 426
Setting Table Loader Transformation Options 427
Selecting a Load Technique in the Table Loader 429
Removing Non-Essential Indexes and Constraints during a Load 432
Considering a Bulk Load 433

Chapter 20 • Working with SAS Sort Transformations 435
About Sort Transformations 435
Optimizing Sort Performance 435
Creating a Table That Contains the Sorted Contents of a Source 438

Chapter 21 • Working with SQL Join Transformations 441
About Join Transformations 443
Using the Designer Window 443
Reviewing and Modifying Clauses, Joins, and Tables in an SQL Query 445
Understanding Automatic Joins 447
Selecting the Join Type 450
Adding User-Written SQL Code 452
Debugging an SQL Query 453
Adding a Column to the Target Table 455
Adding a Join to an SQL Query on the Designer Tab 455
Creating a Simple SQL Query 457
Configuring a SELECT Clause 459
Adding a CASE Expression 461
Creating or Configuring a WHERE Clause 463
Adding a GROUP BY Clause and a HAVING Clause 465
Adding an ORDER BY Clause 468
Adding Subqueries 470
Validating or Submitting an SQL Query 475
Joining a Table to Itself 476
Using Parameters with an SQL Join 477
Constructing a SAS Scalable Performance Data Server Star Join 478
Optimizing SQL Processing Performance 479
Performing General Data Optimization 480
Influencing the Join Algorithm 481
Setting the Implicit Property for a Join 482
Enabling Explicit Pass-Through Processing for SQL Join Transformations 484
Using Properties Window Options to Optimize SQL Processing Performance 486

Chapter 22 • Working with Other SQL Transformations . . . 489
About Other SQL Transformations . . . 489
Inserting Rows into a Target Table . . . 491
Using the SQL Set Operators Transformation . . . 495
Enabling Explicit Pass-Through Processing for Other SQL Transformations . . . 502

Chapter 23 • Working with Iterative Jobs and Parallel Processing . . . 505
About Iterative Jobs . . . 505
Creating and Running an Iterative Job . . . 506
Creating a Parameterized Job . . . 509
Creating a Control Table . . . 512
About Parallel Processing . . . 515
Setting Options for Parallel Processing . . . 516

Chapter 24 • Working with Slowly Changing Dimensions . . . 521
About Slowly Changing Dimensions . . . 522
About Dimension Tables . . . 524
About Fact Tables . . . 527
Usage Notes for Slowly Changing Dimensions . . . 528
Loading a Dimension Table with Type 1 Updates . . . 528
Loading a Dimension Table with Type 1 and 2 Updates . . . 535
Comparing Tables . . . 538
Loading a Fact Table Using Dimension Table Lookup . . . 545
Loading a Table and Adding a Surrogate Primary Key . . . 551
Tracking Changes in Source Datetime Values . . . 554
Closing Out Rows in Datetime Change Tracking . . . 556

Chapter 25 • Working with Change Data Capture . . . 557
About the Change Data Capture Transformations . . . 557
About CDC Changed Data Tables . . . 559
About CDC Control Tables . . . 559
Capture Changed Data from Oracle . . . 560

Chapter 26 • Working with Message Queues . . . 567
About Message Queues . . . 567
Prerequisites for Message Queues . . . 568
Selecting Message Queue Transformations . . . 569
Processing a WebSphere Queue . . . 570
Polling a Websphere Message Queue . . . 572
Processing a Microsoft Queue . . . 574

Chapter 27 • Working with SPD Server Cluster Tables . . . 577
About SPD Server Cluster Tables . . . 577
Creating an SPD Server Cluster Table . . . 578
Maintaining an SPD Server Cluster . . . 579

Chapter 28 • Working with Hadoop and SAS LASR Analytic Server . . . 583
Overview of the Hadoop Transformations . . . 584
Prerequisites for the Hadoop Transformations . . . 585
Creating a Pig Job . . . 586
Creating a Hive Job . . . 591
Creating a Hadoop Container Job . . . 594
Monitoring Hadoop Clusters . . . 603
Overview of the High-Performance Analytics Transformations . . . 603
Prerequisites for the High-Performance Analytics Transformations . . . 605
Loading a Table on the SAS LASR Analytic Server . . . 606

Usage Notes for HPA Software and Hadoop 608

PART 4 Appendixes 611

Appendix 1 • Main Windows and Wizards **613**
Analysis Window 614
Checkouts Tree 615
Code Editor 615
Comparison Results Window 616
Connection Profile Window 617
Desktop 617
Details Pane 619
Expression Builder 620
Folders Tree 625
Inventory Tree 625
Job Editor 629
Properties Windows 631
Reports Window 634
Tools-Options Window 634
Tree View 635
View Data Windows 637
Wizards 639

Appendix 2 • Usage Notes **643**
General Usage Notes 645
Usage Notes for Register Tables Wizards and the New Table Wizard 653
Usage Notes for the View Data Window 658
Usage Notes for Iterative Jobs 662
Prerequisites for Running a Job When a DataFlux Server Is Used for Authentication 664
Usage Notes for Loaders 669

Appendix 3 • Miscellaneous Transformations **673**
Using a Business Rule Flow in a Job 675
Creating a Table That Appends Two or More Source Tables 679
Creating a Publish to Archive Report from Table Data 682
Validating Product Data 689
Creating a Publish to Email Report from Table Data 693
Integrating a SAS Enterprise Miner Model with Existing SAS Data 700
Creating a Publish to Queue Report from Table Data 706
Extracting Data from a Source Table 712
Creating Reports from Table Data 715
Create a Table That Ranks the Contents of a Source 721
Create Two Tables That Are Subsets of a Source 724
Moving Data Directly from One Machine to Another Machine 728
Creating Standardized Statistics from Table Data 732
Creating Transposed Data from Table Data 736
Converting a SAS or DBMS Table to an XML Table 741
Using ODS to Specify Output from the XML Writer 746
Using SOAP to Access a Third-Party Web Service 747
Using REST to Access a Third-Party Web Service 750
Generating Enterprise Decision Management Output 751
Running Conditional Processes 757

Appendix 4 • Java Code and Methods for Report Plug-ins **763**

Example Java Code for a Report Plug-in 763
Reporting Interface Methods 769

Glossary **775**
Index **783**

What's New in SAS Data Integration Studio 4.9

Overview

The main enhancements and changes for SAS Data Integration Studio 4.9 include the following:

- There is a new source designer wizard for Cloudera Impala.
- High Performance Analytics Transformations are now production.
- Enterprise Decision Management transformation is now production.
- Support was added for SAS Lineage.
- Code generation supports credential lookup for DataFlux Authentication Servers.
- SAS package wizards have better support for DataFlux Data Management Servers.

New Source Designer Wizard for Cloudera Impala

Source designer wizards enable you to register tables on the SAS Metadata Server. Registered tables are available to SAS applications that use the metadata server. A new source designer enables you to register tables in a Cloudera Impala environment. Cloudera Impala provides high-performance, low-latency SQL queries on data stored in popular Apache Hadoop file formats.

High Performance Analytics Transformations Are Now Production

The Transformations tree in SAS Data Integration Studio includes a **High Performance Analytics** folder. The transformations in this folder enable you to load and unload SASHDAT tables in a Hadoop cluster or a SAS LASR Analytic Server. For more information about the High Performance Analytics transformations, see the relevant topics in "Working with Hadoop and SAS LASR Analytic Server" on page 584.

Enterprise Decision Management Transformation Is Now Production

The Enterprise Decision Management transformation is now production. It incorporates SAS Decision Manager flows into jobs. The transformation maps physical data to decision flows. The results tables produce decision-making results from the mapped input data. For an example of how this transformation can be used, see "Generating Enterprise Decision Management Output" on page 751.

Support Added for SAS Lineage

If the SAS Lineage web client is available on your network, you can use it to display lineage for tables and external files that are used in SAS Data Integration Studio. For more information, see "Using SAS Lineage" on page 318.

This feature will be available in October 2014.

Code Generation Supports Credential Lookup for DataFlux Authentication Servers

By default, code generated by SAS Data Integration Studio do not contain user names and passwords for data connections. The user name and password are looked up at run time, using authentication domains that have been defined for the authenticating server (SAS Metadata Server or DataFlux Authentication Server). In the past, this feature was limited to SAS Metadata Servers.

SAS Package Wizards Have Better Support for DataFlux Data Management Server

If you export and import jobs that contain DataFlux Batch Job transformations or DataFlux Data Service transformations, you are now prompted to select the DataFlux Data Management Server for the target environment. In earlier releases, you had to make this change after importing the job.

Part 1

Introduction

Chapter 1
Overview of SAS Data Integration *3*

Chapter 1

Overview of SAS Data Integration

About SAS Data Integration 3

Advantages of SAS Data Integration 4

A Basic Data Integration Environment 4
Overview of a Data Integration Environment 4
SAS Management Console 5
SAS Data Integration Studio 5
Servers 6
Libraries 8
Additional Information 8

How to Get Help for SAS Data Integration Studio 8

Installing SAS Data Integration Studio 8

Administrative Documentation for SAS Data Integration Studio 9

Accessibility Features in SAS Data Integration Studio 10
Overview 10
Enabling Assistive Technologies 10
Accessibility Standards 10

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.

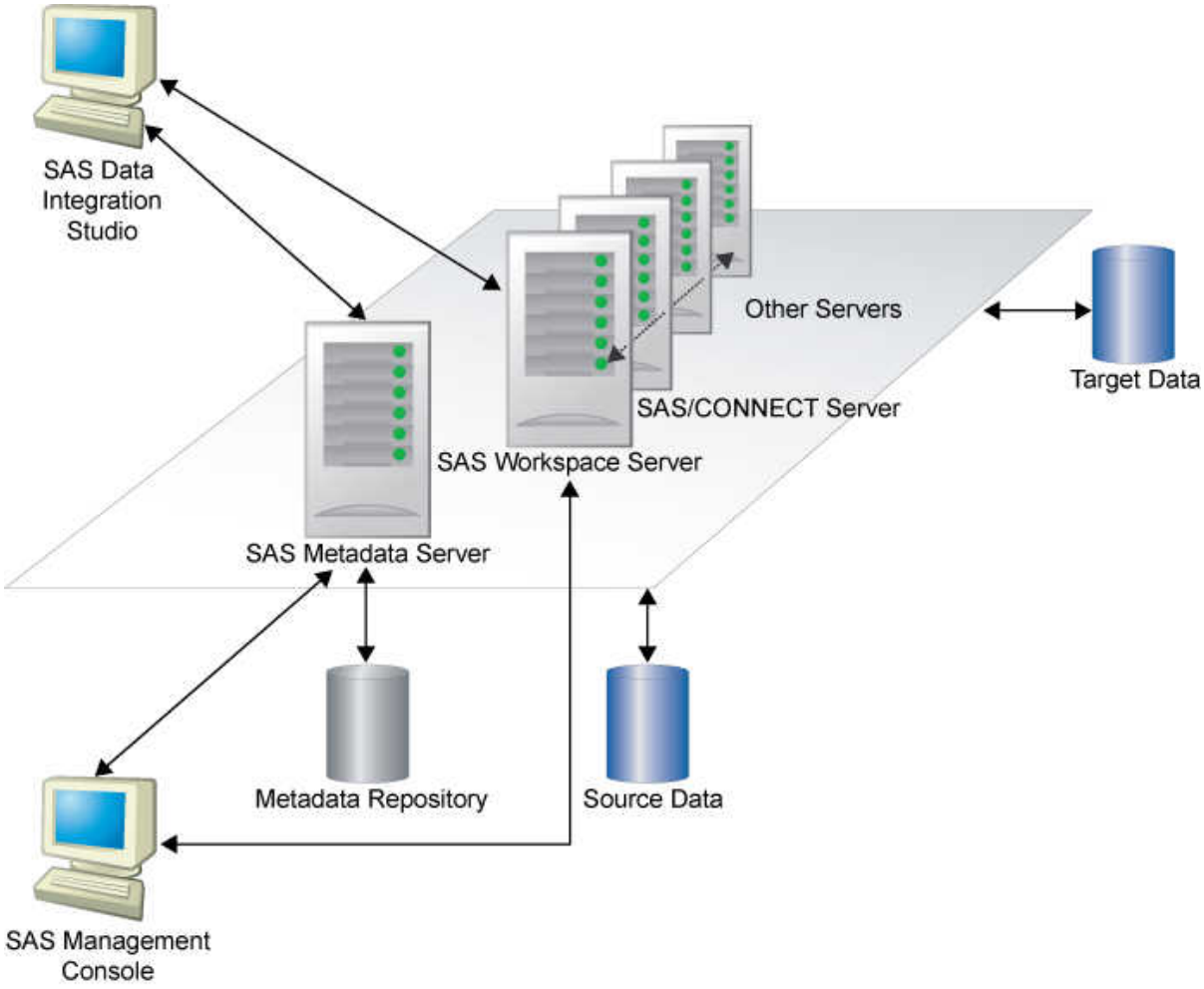
- SAS data integration reduces development time by enabling the rapid generation of data warehouses, data marts, and data streams.
- It controls the costs of data integration by supporting collaboration, code reuse, and common metadata.
- It increases returns on existing IT investments by providing multi-platform 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.
SAS OLAP Server	Creates cubes and processes queries against cubes.	As a component in a SAS Application Server object.

Server	How the Server Is Used	How the Server Is Specified
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=ad1min
path='V2o7223.world'
```

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

Enterprise Resource Management (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.

How to Get Help for SAS Data Integration Studio

Other help resources are available to you in addition to this *SAS Data Integration Studio User's Guide*. If you are working in a window, you can press Help or F1 to display contextual help. If you are migrating from a previous release, see the SAS Data Integration Studio chapter in *SAS Guide to Software Updates*.

The SAS Data Integration Studio product page includes links to useful technical papers. It also includes screencasts that show you how to perform common tasks. You can access the product page at **http://support.sas.com/software/products/etls/**. Finally, the Data Management community is available online for you to share your questions, experiences, and ideas with other users. Access the community at **https://communities.sas.com/community/support-communities/sas_enterprise_data_management_integration**.

Installing SAS Data Integration Studio

SAS Data Integration Studio is installed along with other software as part of a SAS offering. The offering includes the servers and other software that SAS Data Integration Studio requires.

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/documentation/onlinedoc/intellplatform/**.

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

Table 1.3 *SAS Intelligence Platform Documentation for SAS Data Integration Studio*

Administrative Task	Related Documentation
• Set up a folder structure for your site in the **Folders** tree. • Promote metadata (additional information and metadata export and import). • Start, stop, and check the status of servers. • Monitor the system and set up system logs. • Back up and restore your system. • Optimize the performance of the SAS Metadata Server. • Manage SAS metadata repositories.	SAS Intelligence Platform: System Administration Guide
• Set up security.	SAS Intelligence Platform: Security Administration Guide
• Set up data servers and libraries for common data sources.	SAS Intelligence Platform: Data Administration Guide
• Set up SAS Application Servers.	SAS Intelligence Platform: Application Server Administration Guide
• Set up grid computing (so that jobs can execute on a grid).	Grid Computing for SAS
• Set up scheduling for jobs that have been deployed for scheduling.	Scheduling In SAS

Administrative Task	Related Documentation
• Set up change management. • Set up servers and libraries for remote data (multi-tier environments). • Set up support for message queue jobs. • Set up support for Web service jobs and other stored process jobs. • Enable the bulk-loading of data into target tables in a DBMS. • Set up SAS Data Quality software. • Set up support for job status handling. • Set up support for FTP and HTTP access to external files.	SAS Intelligence Platform: Desktop Application Administration Guide
• Work with SAS OLAP cubes.	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`**.

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`**). 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	The software supports keyboard equivalents for all user actions. Tree controls in the user interface can be individually managed and navigated through using the keyboard. However, some exceptions exist. Some ALT key shortcuts are not functional. Also, some more advanced manipulations require a mouse. Still, the basic functionality for displaying trees in the product is accessible from the keyboard. Based on guidance from the Access Board, keyboard access to drawing tasks does not appear to be required for compliance with Section 508 standards. Accordingly, keyboard access does not appear to be required for the **Diagram** tab in the Job Editor window, or the **Designer** tab in the SQL Join properties window. Specifically, use of the **Diagram** tab in the Job Editor and the **Designer** tab in the SQL Join Properties window are functions that cannot be discerned textually. Both involve choosing a drawing piece, dragging it into the workspace, and designing a flow. These tasks 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. **Example:** Use of the **Diagram** 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. 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. 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. 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.

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. **Example:** 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 **Authorization** 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 **Authorization** tab or dialog box.

Accessibility Issue	Support Status	Explanation
Electronic forms and displays.	Supported with exceptions	When navigating with a keyboard to choose a path in the Browse dialog box, the focus disappears. To work around the problem, either (1) count the number of times that you press the TAB key and listen closely to the items, or (2) type the path explicitly. When the user sets the operating system settings to high contrast, some attributes of that setting are not inherited. **Example:** 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.
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 **Help** 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.	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. 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.

Part 2

General User Tasks

Chapter 2
Getting Started *17*

Chapter 3
Importing, Exporting, and Copying Metadata *59*

Chapter 4
Working with Tables *79*

Chapter 5
Working with External Files *119*

Chapter 6
Creating Jobs *145*

Chapter 7
Managing Jobs *165*

Chapter 8
Restarting Jobs From Checkpoints *201*

Chapter 9
Managing the Status of Jobs and Transformations *209*

Chapter 10
Deploying Jobs *225*

Chapter 11
Working with Versions *253*

Chapter 12
Working with Generated Code *263*

Chapter 13
Working with User-Written Code *271*

Chapter 14
Optimizing Process Flows *293*

Chapter 15
Working with Impact Analysis and Data Lineage *311*

Chapter 16
Working with Reports *321*

Chapter 17
Working with Data Management Platform *333*

Chapter 2

Getting Started

Security for SAS Data Integration Studio **19**
Overview of Security 19
Authorization Tab 19

Main Tasks for Creating Process Flows **19**

Starting SAS Data Integration Studio **20**
Problem 20
Solution 20
Tasks 20

Connecting to a SAS Metadata Server **22**
Problem 22
Solution 22
Tasks 23

Working with the Folders Tree **24**
Overview of the Folders Tree 24
Add a Folder 25
Add Metadata Objects to a Folder 25
Copy to Folder 26
Drag to Folder 26
Move to Folder 26
Rename a Folder 26
Considerations When You Change a Folder Path 26

Selecting a Default SAS Application Server **27**
Problem 27
Solution 27
Tasks 27

Registering SAS Libraries **28**
Problem 28
Solution 28
Tasks 28

Working with User-Defined Formats **28**
Problem 28
Solution 29
Tasks 29

Registering Tables and Cubes **29**
Problem 29
Solution 30
Tasks 30

Overview of Transformations **32**
Introduction to Transformations 32
Overview of the Transformations Tree 32
Access Folder 32
Analysis Folder 34
Archived Folder 35
Change Data Capture Folder 35
Control Folder 36
Data Folder 36
Data Quality Folder 38
Hadoop Folder 39
High-Performance Analytics Folder 40
Output Folder 41
Publish Folder 41
SPD Server Dynamic Cluster Folder 41
SQL Folder 42
Ungrouped Folder 43

Working with Stored Processes **43**
Overview 43
View the Version Number for a Stored Process 44
Deploy a Job as a Version 1.0 or Version 2.0 Stored Process 45
Create a Version 2.0 Stored Process 45
Convert a Stored Process from One Version to Another 46

Working with Web Services **46**

Specifying Global Options in SAS Data Integration Studio **46**
Problem 46
Solution 46
Tasks 47

Working with Change Management **47**
Problem 47
Solution 47
Tasks 49

Search Metadata **52**
Problem 52
Solution 52
Tasks 53

Add a Note or Document to a Registered Object **55**
Problem 55
Solution 55
Tasks 56

View the Content of Notes or Documents **57**
Problem 57
Solution 58
Tasks 58

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 47.

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 20
- "Specify Java Options" on page 20
- "Specify the Plug-in Location" on page 21
- "Specify the Error Log Location" on page 21
- "Redirect Local Files Created by SAS Data Integration Studio" on page 21
- "Specify Message Logging" on page 21
- "Change the Memory Allocated to SAS Data Integration Studio" on page 22

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 that you can specify on the command line or add to the **`distudio.ini`** file.

Specify Java Options

To specify the Java options when you start SAS Data Integration Studio, add a JavaArgs_ line in the **`distudio.ini`** file. For example, adding the following two lines specifies the locale as Japanese:

```
JavaArgs_13=-Duser.language=ja
```

```
JavaArgs_14=-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. Edit your **distudio.ini** file to include the additional Java argument **-DAddPluginDir** to point to the plug-in folder. Here is an example:

```
JavaArgs_15=-DAddPluginDir="c:\plugins"
```

Specify the Error Log Location

By default, SAS Data Integration Studio writes error information to a file named **errorlog.txt** in the working directory, such as **C:\Users\user_ID\AppData\Roaming**. 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
'<relative_filepath/filename>'
```

The *relative_filepath* is relative to the working directory unless you redirect the local files created by SAS Data Integration Studio.

Redirect Local Files Created by SAS Data Integration Studio

By default, SAS Data Integration Studio stores the log files, application default files, and connection profiles on the local host. To change the default storage location, follow these steps:

1. Close SAS Data Integration Studio.
2. Create the path and directory for the client files.
3. Open the file **distudio.ini** and add the following Java argument:

   ```
   JavaArgs_xx=-Dsas.appdatapath="new_path"
   ```

   where *xx* is the next available Java argument number, and *new_path* is a fully qualified path to the new directory. Here is an example:

   ```
   JavaArgs_12=-Dsas.appdatapath="\\adminServer02\DISClientFiles\Hostd17362"
   ```

   Note that a "SAS DataIntegrationStudio/version number" folder is created in the path you specified. This folder is the new log location.
4. Start SAS Data Integration Studio.

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 2.1 *Values for level_value*

Value	Description
ALL	All messages are logged.

Value	Description
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.

Change the Memory Allocated to SAS Data Integration Studio

The default amount of memory allocated to SAS Data Integration Studio is 128 megabytes. If you are using Citrix to access SAS Data Integration Studio, you might want to decrease the amount of memory allocated as appropriate for your environment.

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.

Edit the **`distudio.ini`** file (in the default location such as C:\Program Files \SASHome\SASDataIntegrationStudio\<version>) by increasing the memory setting of the JavaArgs_1 parameter to 1024. Add an additional argument to set the MaxPermsize option.

```
JavaArgs_1=Xmx1024m
JavaArgs_13=-XX:MaxPermSize=128m
```

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 23
- "Open a Connection Profile" on page 23
- "Update a Connection Profile" on page 24
- "Reconnecting to a Metadata Server" on page 24

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.

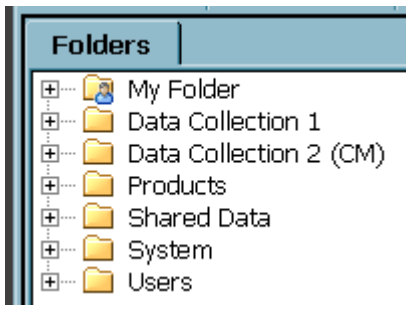
Working with the Folders Tree

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 26.

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 47.

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 26.

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 26.

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 26.

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 29.

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, and 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 type 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*. See also the notes about libraries in "General Usage Notes" on page 645.

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 29.

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 that you want to update, and 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. See "Register Tables or Cubes" on page 30. See also "Usage Notes for Register Tables Wizards and the New Table Wizard" on page 653.

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 28.

Table 2.2 *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 **File** ⇨ **Import** ⇨ **Metadata** from the menu bar to import the metadata. See "Working with SAS Metadata Bridges" on page 66.
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 **File** ⇨ **Import** ⇨ **SAS Package** from the menu bar to import the metadata. See "Working with SAS Package Metadata" on page 60.
One or more SAS tables or database management system tables (DBMS) tables that exist in physical storage.	Select **File** ⇨ **Register Tables** 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 **Register Tables**. See "Registering Existing Tables with the Register Tables Wizard" on page 81.
A table that is specified in a comma-delimited file or in another external file.	Select **File** ⇨ **New** ⇨ **External File** ⇨ **Delimited** from the menu bar, select the appropriate external file format, and then respond to the external file wizard. See "Working with External Files" on page 120.

Objects to be Registered	Method for Specifying Metadata
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 **New** ⇨ **Table** from the menu bar, and then respond to the New Table wizard. See "Registering New Tables with the New Table Wizard" on page 82.
One or more tables that are specified in an XML file.	Select **File** ⇨ **Register Tables** 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 *SAS Intelligence Platform: Data Administration Guide*.
A Microsoft Excel spreadsheet.	Select **File** ⇨ **Register Tables** 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 *SAS Intelligence Platform: Data Administration Guide*.
One or more tables that exist in physical storage and that can be accessed with an Open Database Connectivity (ODBC) driver.	Select **File** ⇨ **Register Tables** 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 *SAS Intelligence Platform: Data Administration Guide*.
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 **File** ⇨ **Register Tables** from the menu bar, select the **Generic** 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.
A SAS cube.	Select **File** ⇨ **New** ⇨ **Cube** from the menu bar, and then respond to the New Cube wizard.

Overview of Transformations

Introduction to Transformations

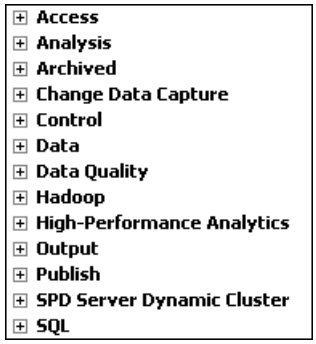
You want to select the right transformation to perform a specific task. The transformation enables you to include that task in a SAS Data Integration Studio job flow.

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 2.2 *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 2.3 *Access Folder Transformations*

Name	Description
DB2 Bulk Table Loader	Used to bulk load SAS and most DBMS source tables to a DB2 target table. For more information, see "About the DB2 Bulk Table Loader" on page 426.
File Reader	Reads an external file and writes to a SAS or DBMS table. For more information, see "Using an External File in the Process Flow for a Job" on page 138.
File Writer	Reads a SAS or DBMS table and writes to an external file. For more information, see "Using an External File in the Process Flow for a Job" on page 138..
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. For more information, see "Creating a Control Table" on page 512.
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. For more information, see "Processing a Microsoft Queue" on page 574.
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. For more information, see "Processing a Microsoft Queue" on page 574.
Oracle Bulk Table Loader	Enables bulk loading of SAS or Oracle source data into an Oracle target. For more information, see "About the Oracle Bulk Table Loader Transformation" on page 425.
REST	Enables you to use the REST approach to read from and write to a third-party web service in the context of a SAS Data Integration Studio job. For more information, see "Using REST to Access a Third-Party Web Service" on page 750.
SOAP	Enables you to use the SAS SOAP procedure to read from and write to a third-party web service in the context of a SAS Data Integration Studio job. For more information, see "Using SOAP to Access a Third-Party Web Service" on page 747.
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. For more information, see "About the SPD Server Table Loader Transformation" on page 422.
Table Loader	Reads a source table and writes to a target table. Provides more loading options than other transformations that create tables. For more information, see "About the Table Loader Transformation" on page 424.

Name	Description
Teradata Table Loader	Enables you to set table options unique to Teradata tables and supports the pushdown feature that enables you to process relational database tables directly on the appropriate relational database server. For more information, see "Teradata Table Loader Transformation" on page 423
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. For more information, see "Processing a WebSphere Queue" on page 570.
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. For more information, see "Processing a WebSphere Queue" on page 570.
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. For example, you cannot use the Table Loader transformation to load an XML table. For more information, see "Converting a SAS or DBMS Table to an XML Table" on page 741.

Analysis Folder

The following table describes the transformations in the **Analysis** folder in the Transformations tree.

Table 2.4 *Analysis Folder Transformations*

Name	Description
Correlations	Creates an output table that contains correlation statistics. For more information, see "Creating a Correlation Analysis" on page 362.
Distribution Analysis	Creates an output table that contains a distribution analysis. For more information, see "Creating a Distribution Analysis" on page 370.
Forecasting	Enables you to run the High-Performance Forecasting procedure (PROC HPF) against a warehouse data store. PROC HPF provides a quick and automatic way to generate forecasts for many sets of time series or transactional data. For more information, see "Generating Forecasts" on page 377.
Frequency	Creates an output table that contains frequency information. For more information, see "Frequency of Eye Color By Hair Color Crosstabulation" on page 385.
One-Way Frequency	Creates a one-way output table that contains frequency information about the relationship between two classification variables. For more information, see "One-Way Frequency of Eye Color By Region" on page 398.

Name	Description
Summary Statistics	Creates an output table that contains summary statistics. For more information, see "Creating Summary Statistics for a Table" on page 407.
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. For more information, see "Creating a Summary Tables Report from Table Data" on page 413.

Archived Folder

In order to support backward 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 is displayed and 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 2.5 *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. For information about the current version, see "About Fact Tables" on page 527.

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 2.6 *Change Folder Transformations*

Name	Description
Attunity CDC	Loads changed data only from Attunity and other selected databases. For more information, see "Working with Change Data Capture" on page 557.

Name	Description
DB2 CDC	Loads changed data only from DB2 databases. For more information, see "Working with Change Data Capture" on page 557.
General CDC	Loads changed data only from a wide range of databases. For more information, see "Working with Change Data Capture" on page 557.
Oracle CDC	Loads changed data only from Oracle databases. For more information, see "Working with Change Data Capture" on page 557.

Control Folder

The following table describes the transformations in the **Control** folder in the Transformations tree.

Table 2.7 *Control Folder Transformations*

Name	Description
Conditional End	Marks the end of a conditional process in a job. For more information, see "Running Conditional Processes" on page 757.
Conditional Start	Marks the beginning of a conditional process in a job. For more information, see "Running Conditional Processes" on page 757.
Loop	Marks the beginning of the iterative processing sequence in an iterative job. For more information, see "Creating and Running an Iterative Job" on page 506.
Loop End	Marks the end of the iterative processing sequence in an iterative job. For more information, see "Creating and Running an Iterative Job" on page 506.
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. For more information, see "Perform Actions Based on the Status of a Transformation" on page 217.

Data Folder

The following table describes the transformations in the **Data Transforms** folder in the Transformations tree.

Table 2.8 *Data Folder Transformations*

Name	Description
Append	Creates a single target table by combining data from several source tables. For more information, see "Creating a Table That Appends Two or More Source Tables" on page 679.

Name	Description
Business Rules	Enables you to use the business rule flow packages that are created in SAS Business Rules Manager in the context of a SAS Data Integration Studio job. You can import business rule flows, specify flow versions, map source table columns to required input columns, and set business rule options. The Business Rules transformation enables you to map your source data and output data into and out of the rules package. Then, the SAS Data Integration Studio job applies the rules to your data as it is run. When you run a job that includes a rules package, statistics are collected. Statistics include the number of rules that were triggered, and the number of invalid and valid data record values. You can use this information to further refine your data as it flows through your transformation logic. For more information, see "Using a Business Rule Flow in a Job" on page 675.
Compare Tables	Enables you to detect changes between two tables such as an update table and a master table and generate a variety of output for matched and unmatched records. For more information, see "Comparing Tables" on page 538.
Data Transfer	Moves data directly from one machine to another. Direct data transfer is more efficient than the default transfer mechanism. For more information, see "Moving Data Directly from One Machine to Another Machine" on page 728.
Data Validation	Cleanses data before it is added to a data warehouse or data mart. For more information, see "Validating Product Data" on page 689.
Enterprise Decision Management	Maps physical data from an Enterprise Decision Management flow package to decision flows. The output tables attached to the transformation produce decision-making results from the mapped input data. For more information, see "Generating Enterprise Decision Management Output" on page 751.
Key Effective Date	Enables change tracking in intersection tables. For more information, see "Tracking Changes in Source Datetime Values" on page 554.
Lookup	Loads a target with columns taken from a source and from several lookup tables. For more information, see "Loading a Fact Table Using Dimension Table Lookup" on page 545.
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. For more information, see "Integrating a SAS Enterprise Miner Model with Existing SAS Data" on page 700.
Rank	Ranks one or more numeric column variables in the source and stores the ranks in the target. For more information, see "Create a Table That Ranks the Contents of a Source" on page 721.
SCD Type 1 Loader	Enables you to load a dimension table using type 1 updates. Type 1 updates insert new rows, update existing rows, and generate surrogate key values in a dimension table without maintaining a history of data changes. Each business key is represented by a single row in the dimension table. For more information, see "Loading a Dimension Table with Type 1 Updates" on page 528.

Name	Description
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. For more information, see "Loading a Dimension Table with Type 1 and 2 Updates" on page 535.
Sort	Reads data from a source, sorts it, and writes the sorted data to a target. For more information, see "Creating a Table That Contains the Sorted Contents of a Source" on page 438.
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. For more information, see "Create Two Tables That Are Subsets of a Source" on page 724.
Standardize	Creates an output table that contains data standardized to a particular number. For more information, see "Creating Standardized Statistics from Table Data" on page 732.
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. For more information, see "Loading a Table and Adding a Surrogate Primary Key" on page 551.
Transpose	Creates an output table that contains transposed data. For more information, see "Creating Transposed Data from Table Data" on page 736.
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. For more information, see "Adding a User Written Code Transformation to a Job" on page 274.

Data Quality Folder

The following table describes the transformations in the **Data Quality** folder in the Transformations tree. In general, you can use Apply Lookup Standardization, Create Match Code, and Standardize with Definition for data cleansing operations. You can use DataFlux Batch Job and DataFlux Data Service to perform tasks that are a specialty of DataFlux software, such as profiling, monitoring, or address verification.

Name	Description
Apply Lookup Standardization	Enables you to select and apply DataFlux schemes that standardize the format, casing, and spelling of character columns in a source table. For more information, see "Standardizing Values with a Standardization Scheme" on page 341.

Name	Description
Create Match Code	Enables you to analyze source data and generate match codes based on common information shared by clusters of records. Comparing match codes instead of actual data enables you to identify records that are in fact the same entity, despite minor variations in the data. For more information, see "Using Match Codes to Improve Record Matching" on page 347.
DataFlux Batch Job	Enables you to select and execute a DataFlux job that is stored on a DataFlux Data Management Server. You can execute DataFlux Data Management Studio data jobs, process jobs, and profiles. You can also execute Architect jobs that were created with DataFlux® dfPower® Studio. For more information, see "Using a DataFlux Job or Profile in a SAS Data Integration Studio Job" on page 355.
DataFlux Data Service	Enables you to select and execute a data job that has been configured as a real-time service and deployed to a DataFlux Data Management Server. For more information, see "Using a DataFlux Data Service in a SAS Data Integration Studio Job" on page 351.
Standardize with Definition	Enables you to select and apply DataFlux standardization definitions to elements within a text string. For example, you might want to change all instances of "Mister" to "Mr." but only when "Mister" is used as a salutation. For more information, see "Standardizing Values with a Definition" on page 346.

Hadoop Folder

Hadoop is an open-source technology for large data volume storage and processing. Hadoop provides scalability through the union of the Hadoop Distributed File System (HDFS), its high bandwidth and clustered storage system, and Map Reduce, its fault-tolerant, distributed processing algorithm.

The following table describes the transformations in the **Hadoop** folder in the Transformations tree.

Name	Description
Hadoop Container	Enables you to use one transformation to perform a series of steps in one connection to the Hadoop cluster. The steps could include transfers to and from Hadoop, Map Reduce processing, and Pig Latin processing. For more information, see "Creating a Hadoop Container Job" on page 594.
Hadoop File Reader	Reads a specified file from a Hadoop Cluster.
Hadoop File Writer	Writes a specified file to a Hadoop Cluster.
Hive	Enables you to submit your own HiveQL code in the context of a job. For more information, see "Creating a Hive Job" on page 591.

Name	Description
Map Reduce	Enables you to submit your own Map Reduce code in the context of a job. You must create your own Map Reduce program in Java and save it to a JAR file. You then specify this JAR file in the Map Reduce transformation, along with some relevant arguments. Your Hadoop installation usually includes an example Map Reduce program. For an example of Map Reduce processing in a Hadoop container job, see "Creating a Hadoop Container Job" on page 594.
Pig	Enables you to submit your own Pig Latin code in the context of a job. For more information, see "Creating a Pig Job" on page 586.
Transfer From Hadoop	Transfer a specified file from a Hadoop cluster. For an example of how this transformation can be used, see "Creating a Hadoop Container Job" on page 594
Transfer To Hadoop	Transfer a specified file to a Hadoop cluster.

For more information about these transformations, see "Working with Hadoop and SAS LASR Analytic Server" on page 584.

High-Performance Analytics Folder

The Transformations tree in SAS Data Integration Studio includes a **High-Performance Analytics** folder. You can use these transformations to load and unload tables on a Hadoop cluster or a SAS® LASR™ Analytic Server. These transformations are typically used to support a SAS Analytics solution that includes both SAS Data Integration Studio and SAS LASR Analytic Server.

Name	Description
SAS Data in HDFS Loader	Loads a table to the file system (HDFS) on a Hadoop cluster. The source can be a SAS data set or a table in any DBMS supported by SAS. The target is a table in a SAS Data in HDFS Library.
SAS Data in HDFS Unloader	Unloads a table from HDFS. The input is a table in a SAS Data in HDFS Library.
SAS LASR Analytic Server Loader	Loads a table to memory on a SAS LASR Analytic Server. The source can be a SAS data set, a table in any DBMS supported by SAS, or a table in a SAS Data in HDFS Library. The target is an in-memory table in a SAS LASR Analytic Server Library.
SAS LASR Analytic Server Unloader	Unloads a table from memory on a SAS LASR Analytic Server. The input is an in-memory table in a SAS LASR Analytic Server Library.

For more information about these transformations, see "Working with Hadoop and SAS LASR Analytic Server" on page 584.

Output Folder

The following table describes the transformations in the **Output** folder in the Transformations tree.

Table 2.9 *Output Folder Transformations*

Name	Description
List Data	Creates an HTML report that contains selected columns from a source table. For more information, see "Creating Reports from Table Data" on page 715.

Publish Folder

The following table describes the transformations in the **Publish** folder in the Transformations tree.

Table 2.10 *Publish Folder Transformations*

Name	Description
Publish to Archive	Creates an HTML report and an archive of the report. For more information, see "Creating a Publish to Archive Report from Table Data" on page 682.
Publish to Email	Creates an HTML report and e-mails it to a designated address. For more information, see "Creating a Publish to Email Report from Table Data" on page 693.
Publish to Queue	Creates an HTML report and publishes it to a queue using MQSeries. For more information, see "Creating a Publish to Queue Report from Table Data" on page 706.

SPD Server Dynamic Cluster Folder

The following table describes the transformations in the **SPD Server Dynamic Cluster** folder in the Transformations tree.

Table 2.11 *SPD Server Dynamic Cluster Folder Transformations*

Name	Description
Create or Add to a Cluster	Creates or updates an SPD Server cluster table. For more information, see "Creating an SPD Server Cluster Table" on page 578.
List Cluster Contents	Lists the contents of an SPD Server cluster table. For more information, see "Maintaining an SPD Server Cluster" on page 579.

Name	Description
Remove Cluster Definition	Deletes an SPD Server cluster table. For more information, see "Maintaining an SPD Server Cluster" on page 579.

SQL Folder

The following table describes the transformations in the **SQL** folder in the Transformations tree. For more information, see "Working with SQL Join Transformations" on page 443 and "Working with Other SQL Transformations" on page 489.

Table 2.12 *SQL Folder Transformations*

Name	Description
Create Table	Provides a simple SQL interface for creating tables.
Delete	Generates a PROC SQL statement that deletes user-selected rows in a single target table. Supports delete, truncate, or delete with a WHERE clause. Also supports implicit and explicit pass-through.
Execute	Enables you to specify custom SQL code to be executed and provides SQL templates for supported databases.
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. For more information, see "Extracting Data from a Source Table" on page 712.
Insert Rows	Provides a simple SQL interface for inserting rows into a target table. For more information, see "Inserting Rows into a Target Table" on page 491.
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. For more information, see "Creating a Simple SQL Query" on page 457.
Merge	Inserts new rows and updates existing rows using the SQL Merge DML command. The command was officially introduced in the SQL:2008 standard.
Set Operators	Enables you to use set operators to combine the results of table-based queries. For more information, see "Using the SQL Set Operators Transformation" on page 495.
Update	Updates user-selected columns in a single target table. The target columns can be updated by case, constant, expression, or subquery. Handles correlated subqueries.

Note: Some functions in the Delete, Execute, Insert Rows, Merge, and Update transformations might work only when the table comes from a database management system that provides an implementation of an SQL command for which a SAS/ACCESS interface is available. One example is sort. You can use SAS tables and tables from database management systems that do not implement the SQL command, but these command-specific functions might not work.

Ungrouped Folder

The **Ungrouped** 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.

Working with Stored Processes

Overview

You can create two types of stored processes in SAS Data Integration Studio:

- Version 1.0 stored processes, which are the IOM Direct Interface Stored Processes that were introduced in SAS 8.
- Version 2.0 stored processes, which are the SAS Stored Processes that were introduced in SAS 9.

The following table compares the compatibility and features available in the two versions of stored processes.

Table 2.13 *Stored Process Feature Comparison*

Version 1.0	Version 2.0
Compatible with server versions prior to SAS 9.3 and SAS 9.3 or later servers.	Compatible with SAS 9.3 or later servers only.
Associated with a specific logical server, which can be a SAS Stored Process Server or a SAS Workspace Server.	Associated with an application server context, and can be run by either a SAS Stored Process Server or a SAS Workspace Server. You can choose whether to restrict the server type or let the client application make the server selection.
Stores source code on the application server.	Stores source code either on the application server or in metadata.
Allows execution on the specified application server only.	Allows execution on other application servers or on the specified application server only.
Requires the *ProcessBody; comment if they are running on a workspace server.	Does not require the *ProcessBody; comment, regardless of which server is used.

Version 1.0	Version 2.0
Must use the stored process server to produce streaming output.	Uses either the stored process server or the workspace server to produce streaming output.
Data sources and targets can be generic streams or XML streams.	Data sources and targets can be generic streams, XML streams, or data tables.

You can perform the following tasks with stored processes:

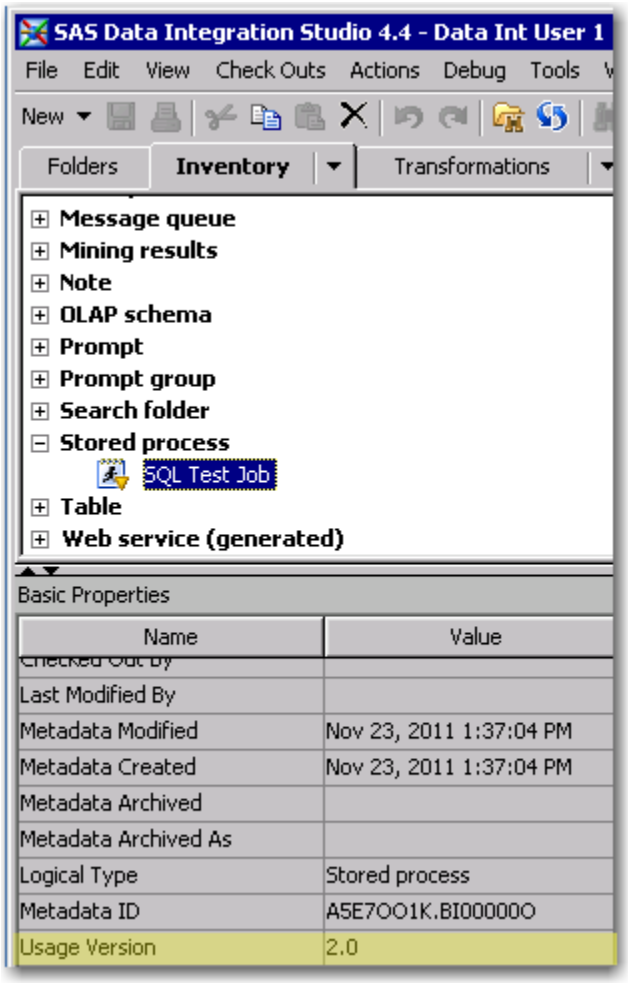
- “View the Version Number for a Stored Process” on page 44
- “Deploy a Job as a Version 1.0 or Version 2.0 Stored Process” on page 45
- “Create a Version 2.0 Stored Process” on page 45
- “Convert a Stored Process from One Version to Another” on page 46

View the Version Number for a Stored Process

To view the version number for an existing stored process, perform the following steps:

1. From the desktop, verify that the **View** ⇨ **Basic Properties** option is selected.
2. Navigate to a folder that contains stored processes.
3. Select a stored process. You can then view the version number in the Basic Properties pane, as shown in the following figure.

Display 2.3 *Basic Properties for a Stored Process*



Deploy a Job as a Version 1.0 or Version 2.0 Stored Process

You can deploy an existing job as a version 1.0 or version 2.0 stored process. For information, see the stored process topics in "Deploying Jobs as Stored Processes" on page 238.

Create a Version 2.0 Stored Process

To create a new version 2.0 stored process that is not based on a SAS Data Integration Studio job, right-click a folder in the Folders tree and select **Stored Process** from the **New** menu. You can also select **Stored Process** in the **New** item on the toolbar. Either method displays the New Stored Process wizard.

For detailed information about creating a stored process, navigate to the Execution page of the wizard. Then, click **Manage** to display the Manage Source Code Repositories window. Finally, click **Help**. Open the **Stored Process Management** folder to review the available topics.

Convert a Stored Process from One Version to Another

You can convert a stored process from one version to another. For example, you might deploy a job as a version 1.0 stored process, but later you might want to take advantage of the version 2.0 features. In that case, you can deploy the job as a version 1.0 stored process. Then, you can upgrade that stored process to version 2.0 and access the new features.

To convert a version 1.0 stored process to version 2.0, right-click the stored process and select **Upgrade**. You can verify that the version number in the **Usage Version** field in the Basic Properties pane has been changed to 2.0. You can open the Properties window of the upgraded stored process and enable the 2.0 features on the **Execution** tab.

You might also want to convert a version 2.0 stored process to a version 1.0 stored process in order to run it on an older server (a server with a version prior to SAS 9.3). To convert a version 2.0 stored process, select the stored process. Open the Properties window to verify that no features that are unique to version 2.0 are being used. Then, right-click the stored process and select **Make Compatible**. If the stored process runs on a SAS Workspace server, make sure that the *ProcessBody; comment is included in the source code. You can verify that the version number in the **Usage Version** field in the Basic Properties pane has been changed to 1.0.

Working with Web Services

You can use a web service client to execute SAS Data Integration Studio jobs. For more information, see the web service topics in "Deploying Jobs" on page 226. You can use SAS Data Integration Studio jobs to call third-party web services. For more information, see the SOAP and REST topics in Appendix 3, "Miscellaneous Transformations," on page 673.

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 20
- "Use the Global Options Window" on page 47

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

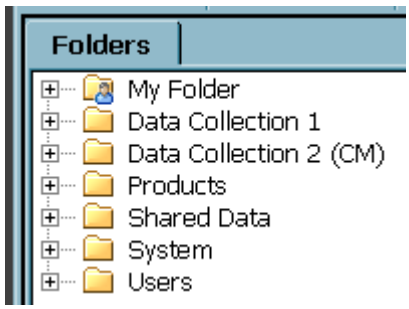
Problem

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 2.4 *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 2.5 *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 49
- "Create a Connection Profile for an Administrator under Change Management" on page 49
- "Add New Metadata" on page 49
- "Check In Metadata" on page 50
- "Check Out Metadata" on page 50
- "Delete Metadata" on page 51
- "Undo Checkouts" on page 51

- "Clear All Metadata from Your Project" on page 51
- "Clear All Metadata from a Project That You Do Not Own" on page 51

See also "Usage Notes for Change Management" on page 52.

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 that 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, and 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 check-out 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 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, and then select **Checkouts** ⇨ **Clear** from the menu bar.

Clear All Metadata from a Project That You Do Not Own

Some problems 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 49.
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 types 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.

Users who are working under change management should not run the Import Metadata Wizard with the **Compare import metadata to repository** option selected. The import and comparison can fail when metadata is imported to a folder that is under change management. For more information, see "Solution" on page 70.

Search Metadata

Problem

You want to create complicated searches of the metadata of the current repository that you have specified in your user profile. You also want the ability of save search criteria for reuse.

Solution

You can use the Search window that you can access from the **Tools** menu. The search function enables you to search for objects by name, which includes the ability to search for patterns. You can subset a search to a specific folder, search by type, by last change date, or by other user-defined criteria. You can also save searches to a folder and bring them up later when needed. For example, you can use the saved search feature to maintain a recently changed object list.

The Search window enables you to perform the following tasks:

- "Specify Basic Search Criteria" on page 53
- "Select Object Types" on page 53

- "Specify a Date Range" on page 54
- "Create Advanced Search Filters" on page 54
- "Run the Search" on page 54
- "Save Search Criteria" on page 55
- "Reuse a Saved Search" on page 55

Tasks

Specify Basic Search Criteria

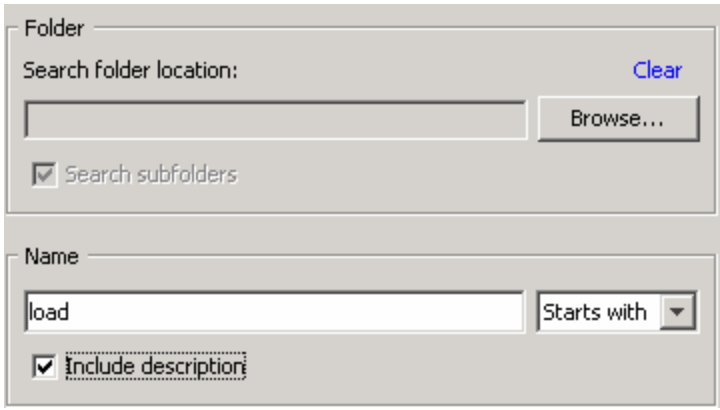
You can specify basic search criteria in the **Folder** and **Name** sections of the Search window.

Perform the following steps to specify basic search criteria:

1. Determine whether you need to specify a folder location or a name. For this example, try specifying a name but leaving the **Search folder location** blank.
2. Enter text that you want to find into the **Name** field. Enter *load* and select **Starts with** in the drop-down list. Finally, select the **Include description** check box. So far, you are searching for objects that begin with the text *load*. You are also searching in description columns.

   The basic search criteria are shown in the following display:

   Display 2.6 *Basic Search Criteria*



Select Object Types

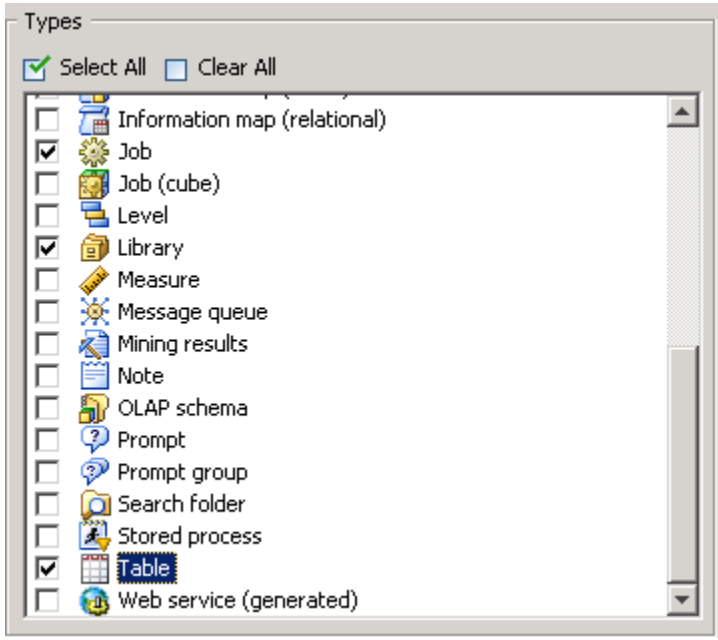
You can use the **Types** section of the Search window specify the types of objects that are included in the search. By default, all of the types are select. However, you can easily create a more selective list.

Perform the following steps to select object types:

1. Click **Clear All** to deselect all of the object types.
2. Select the object types that you want to include in the search, such as **Job**, **Library**, and **Table**.

The following display shows the type criteria for the sample search:

Display 2.7 *Type Criteria*

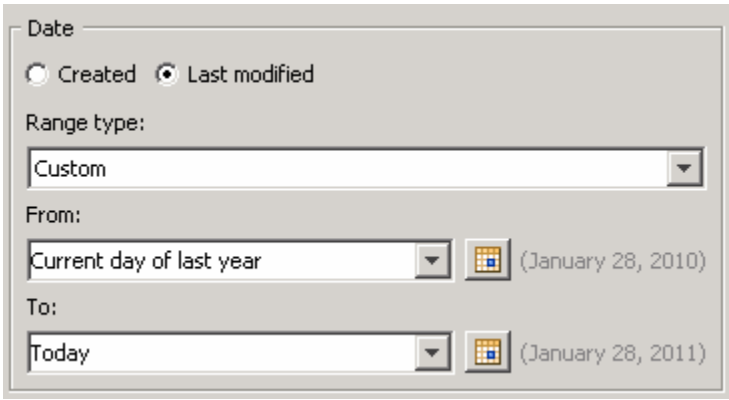


Specify a Date Range

You can use the fields in the **Date** section of the window.

The date range for the sample search is shown in the following display:

Display 2.8 *Date Criteria*



Create Advanced Search Filters

Click **Advanced** to further restrict the search by specifying keywords that the object must have or by specifying a responsible party the object must have. A responsible party is specified by a person's name and the person's role for the object.

Run the Search

Click **Search** to run the search after all the criteria have been entered.

The following display shows a portion of the results from the sample search:

Display 2.9 *Search Results*

Name	Type	Folder Lo...	Description	Date Modified
Load _41_VENTAS	Job	/heike/SI...	Load R3 Hierar...	Oct 26, 2010 1:45:...
Load dimension 0D...	Job	/heike/SI...		Sep 30, 2010 9:42:...
Load dimension 0D...	Job	/heike/SI...		Dec 3, 2010 2:41:0...
Load dimension 0D...	Job	/heike/SI...		Sep 30, 2010 9:42:...
Load dimension 0D...	Job	/heike/SI...		Dec 3, 2010 2:41:0...
Load dimension 0D...	Job	/heike/SI...		Sep 30, 2010 9:42:...
Load dimension 0D...	Job	/heike/SI...		Dec 3, 2010 2:41:0...
Load dimension 0D...	Job	/heike/SI...		Sep 30, 2010 9:42:...
Load dimension 0D...	Job	/heike/SI...		Dec 3, 2010 2:41:0...
Load dimension 0D...	Job	/heike/SI...		Sep 30, 2010 9:42:...
Load dimension 0D...	Job	/heike/SI...		Dec 3, 2010 2:41:0...
Load dimension 0D...	Job	/heike/SI...		Sep 30, 2010 9:42:...
Load dimension 0D...	Job	/heike/SI...		Dec 3, 2010 2:41:0...

Save Search Criteria

Click **Save** to save the criteria for the current search. You can specify the name and location of the saved search.

Reuse a Saved Search

Right-click a saved search, and then click **Open** in the pop-up menu to reuse it and the criteria that it contains. Note that a selected search runs immediately when you open it. Some searches can take a long time to execute.

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 56
- "Create a Note and Attach It to a Metadata Object" on page 56
- "Create a Document and Attach It to a Metadata Object" on page 56
- "Attach One or More Registered Notes or Documents to a Metadata Object" on page 57
- "Associate a Quick Note, a Note, or a Document with a Column" on page 57

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 58
- "View Notes in the SAS Data Integration Studio Tree View" on page 58
- "View Documents in the SAS Data Integration Studio Tree View" on page 58

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

Metadata Import and Export in SAS Data Integration Studio **60**

Working with SAS Package Metadata **60**
About Importing and Exporting SAS Package Metadata 60
Objects That Can Be Imported and Exported in SAS Package Format 61

Preparing to Import or Export SAS Package Metadata **61**
General Preparation 61
Preparing to Export and Import Jobs with Data Quality Transformations 61

Exporting SAS Package Metadata **62**
Problem 62
Solution 62
Tasks 62

Importing SAS Package Metadata **63**
Problem 63
Solution 63
Tasks 64
Usage Notes 64

Copying and Pasting Metadata Objects **65**
Problem 65
Solution 65
Tasks 65
Usage Notes 65

Working with SAS Metadata Bridges **66**
About SAS Metadata Bridges 66
Objects That Can be Imported or Exported with a SAS Metadata Bridge 66

Usage Notes for Importing or Exporting with a SAS Metadata Bridge **66**

Preparing to Import or Export with a SAS Metadata Bridge **67**

Importing New Metadata with a SAS Metadata Bridge **68**
Problem 68
Solution 68
Tasks 68

Importing Updated Metadata with a SAS Metadata Bridge **70**
Problem 70
Solution 70
Tasks 70

Exporting Metadata with a SAS Metadata Bridge **75**
Problem 75

Solution 75
Tasks 76

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*.

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 625.

Preparing to Import or Export SAS Package Metadata

General Preparation

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*.

Preparing to Export and Import Jobs with Data Quality Transformations

If you export and import jobs that contain DataFlux Batch Job transformations or DataFlux Data Service transformations, you will be prompted to select the DataFlux Data Management Server for the target environment. Be sure to do so, or the DataFlux transformations might execute on the old server.

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 61.

Perform the following tasks:

- "Document the Metadata That Will Be Exported (optional) " on page 62
- "Export Selected Metadata" on page 62

Tasks

Document the Metadata That Will Be Exported (optional)

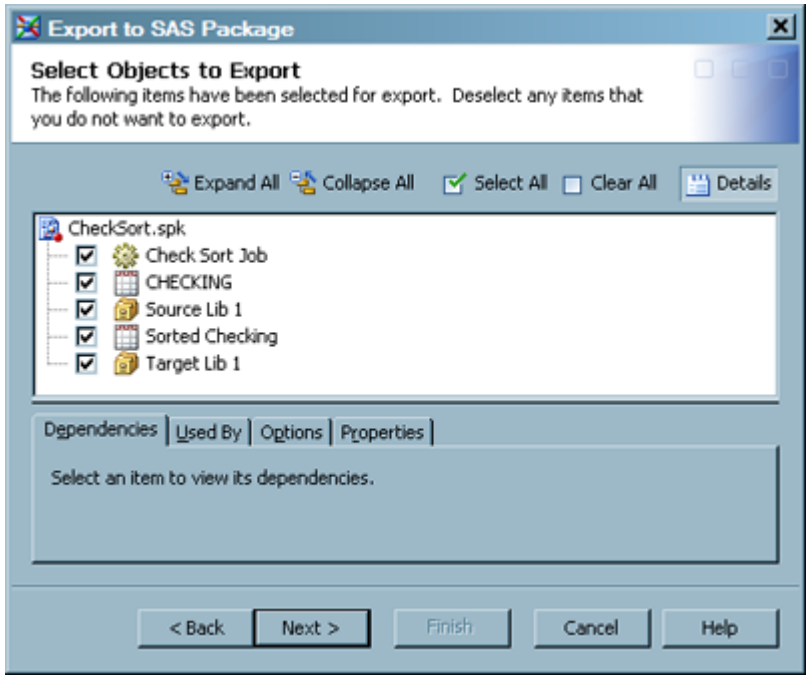
Metadata export and import tasks are easier to manage if you create a document that describes the metadata to be exported, the metadata that should be imported, and the main metadata associations that must be reestablished in the target environment. Otherwise, you might have to guess about these issues 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 61.

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 is displayed. 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 that 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 is displayed, 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**.

Usage Notes

A Generated transformation is a custom transformation that you create with the Transformation Generator wizard.

The following rules govern the import of a Generated transformation:

- If the name of the imported transformation is unique in the target metadata repository, or at least unique within the target folder, then use that name for the target transformation.
- If the target transformation has the same ID as the source transformation, then the source transformation will overwrite the target transformation.

If these rules are not followed, then the import fails.

For more information about generated transformations, see "Working with User-Written Code" on page 271.

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, and then select **Paste Special** from the pop-up menu.

Usage Notes

A Generated transformation is a custom transformation that you create with the Transformation Generator wizard. If you copy and paste a Generated transformation, a new ID that is unique across all active metadata repositories is applied. Otherwise, the normal rules for copy and paste apply. For more information about generated transformations, see "Working with User-Written Code" on page 271.

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.
- Users who are working under change management should not run the Import Metadata Wizard with the **Compare import metadata to repository** option

selected. The import and comparison can fail when metadata is imported to a folder that is under change management. For more information, see "Solution" on page 70.

- 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. It 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 bridge. . The bridges appropriate for your site were probably installed along with other SAS software. For a list of the available bridges, see the Metadata Bridges page: **`http://support.sas.com/software/bridges/`**.

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.

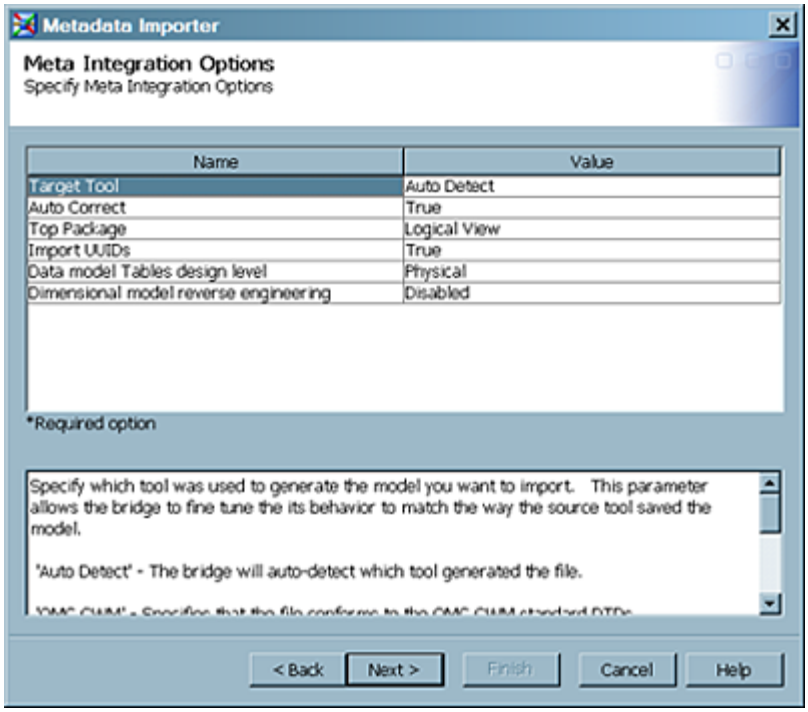
Perform the following steps to import metadata for one or more tables that have never been registered on the current metadata server:

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*



   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. 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.

Note: Users who are working under change management should not run the Import Metadata Wizard with the **Compare import metadata to repository** option selected. The import and comparison can fail when metadata is imported to a folder that is under change management.

If you want to use the **Compare import metadata to repository** option to import metadata to a folder that is under change management, an administrator with write privileges to the change-managed folder must perform the steps that are described in "Import the Metadata to be Compared" on page 70. After the metadata has been imported by an administrator, users who are working under change management can view differences and apply changes.

Perform the following tasks:

- "Import the Metadata to be Compared" on page 70
- "Compare the Imported Metadata to the Existing Metadata" on page 73
- "Applying Changes to Tables with Foreign Keys" on page 74
- "Restoring Metadata for Foreign Keys" on page 75
- "Deleting an Invalid Change Analysis Result" on page 75

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:

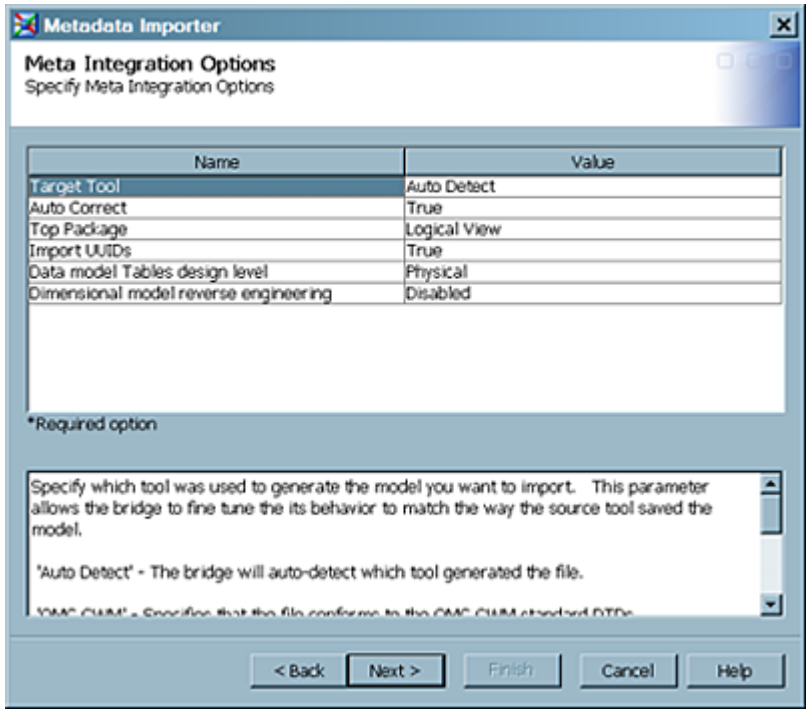
1. Right-click the folder in the Folders tree that stores the imported metadata. Then, select **Import** ⇨ **Metadata** to access the Select an import format page of the Metadata Import Wizard. This page lists the formats that are licensed for your site.

   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*



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.

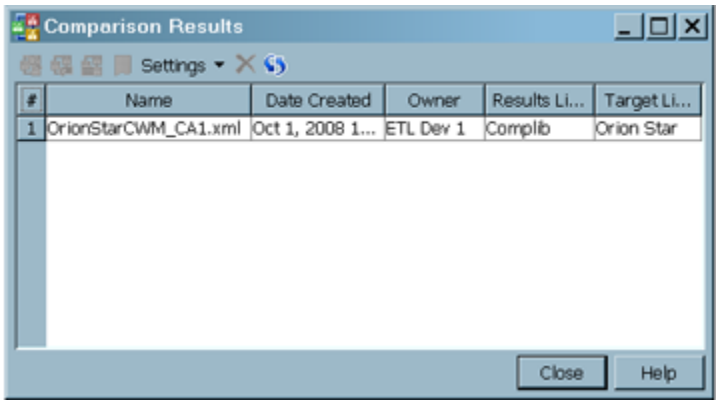
Also, be aware that if you are working under change management, the imported tables might not appear in the Checkouts tree until you refresh the tree. Right-click the Checkouts tree and select **Refresh**.

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*

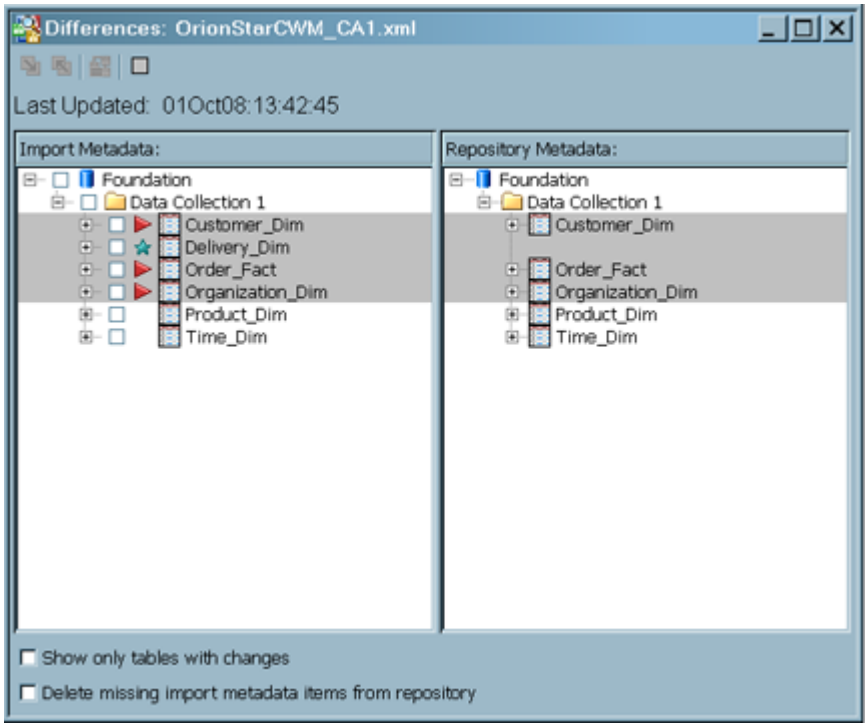
   

   The Comparison Results window enables you to select the results of a compare import metadata to repository operation. There is one record for each successful comparison operation. Select the desired comparison record. Then, click the **View differences found** icon in the toolbar to access the Differences 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, and 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. 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 60.

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 with a SAS Metadata Bridge" on page 67..

Perform the following tasks:

- "Document the Metadata That Will Be Exported (optional) " on page 76
- "Export Selected Metadata " on page 76

Tasks

Document the Metadata That Will Be Exported (optional)

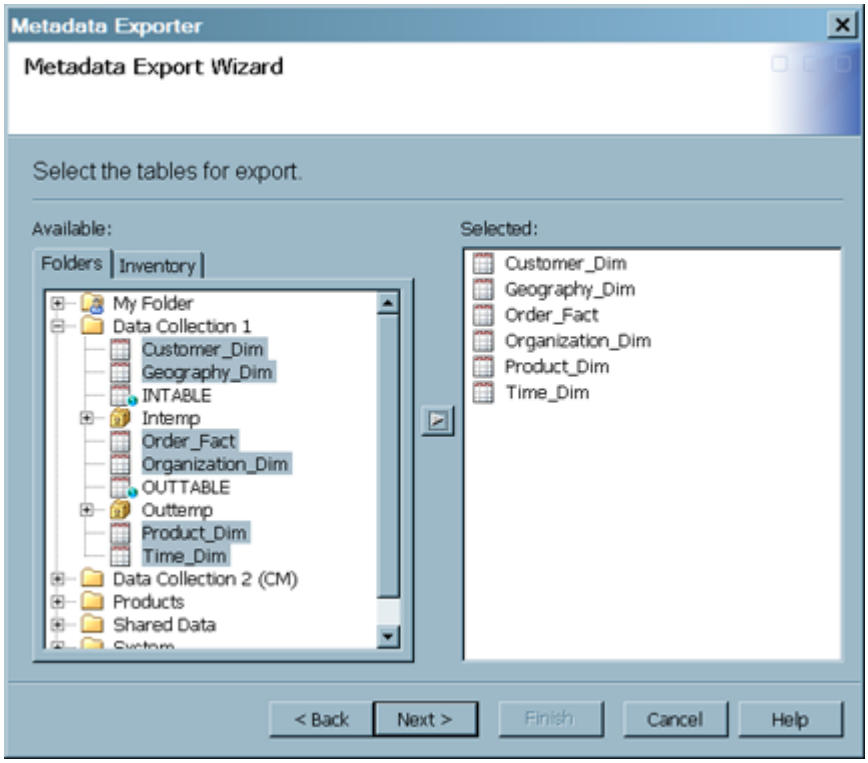
Metadata export and import tasks are easier to manage if you create a document that describes the metadata to be exported, the metadata that should be imported, and the main metadata associations that must be reestablished in the target environment. Otherwise, you might have to guess about these issues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

About Tables **80**

Registering Existing Tables with the Register Tables Wizard **81**
Problem 81
Solution 81
Tasks 81

Registering New Tables with the New Table Wizard **82**
Problem 82
Solution 82
Tasks 83

Viewing or Updating Table Metadata **84**
Problem 84
Solution 85

Using a Physical Table to Update Table Metadata **85**
Problem 85
Solution 85
Tasks 86
Usage Note 86

Specifying Options for Tables **86**
Problem 86
Solution 86
Tasks 87

Supporting Case and Special Characters in Table and Column Names **88**
Overview 88
About Case and Special Characters in SAS Names 88
About Case and Special Characters in DBMS Names 89
Set Default Name Options for New Tables 91
Set Name Options in the Register Tables Wizard 91
Set Name Options for Registered Tables 92

Maintaining Column Metadata **92**
Problem 92
Solution 92
Tasks 92

Standardizing Columns **98**
Problem 98
Solution 98
Tasks 99

Maintaining Keys **104**

Problem . . . 104
Solution . . . 105
Tasks . . . 105

Maintaining Indexes . . . 109
Problem . . . 109
Solution . . . 109
Tasks . . . 110

Browsing Table Data . . . 111
Problem . . . 111
Solution . . . 111
Tasks . . . 111

Editing SAS Table Data . . . 114
Problem . . . 114
Solution . . . 114
Tasks . . . 114

Using the View Data Window to Create a SAS Table . . . 117
Problem . . . 117
Solution . . . 117
Tasks . . . 118

Specifying Browse and Edit Options for Tables and External Files . . . 118
Problem . . . 118
Solution . . . 118

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 "Registering Existing Tables with the Register Tables Wizard" on page 81 and "Registering New Tables with the New Table Wizard" on page 82.
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 "Creating a Process Flow for a Job" on page 148.
View the data or metadata for a registered table.	For more information, see "Browsing Table Data" on page 111 and "Viewing or Updating Table Metadata" on page 84.

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.

See also "Usage Notes for Register Tables Wizards and the New Table Wizard" on page 653.

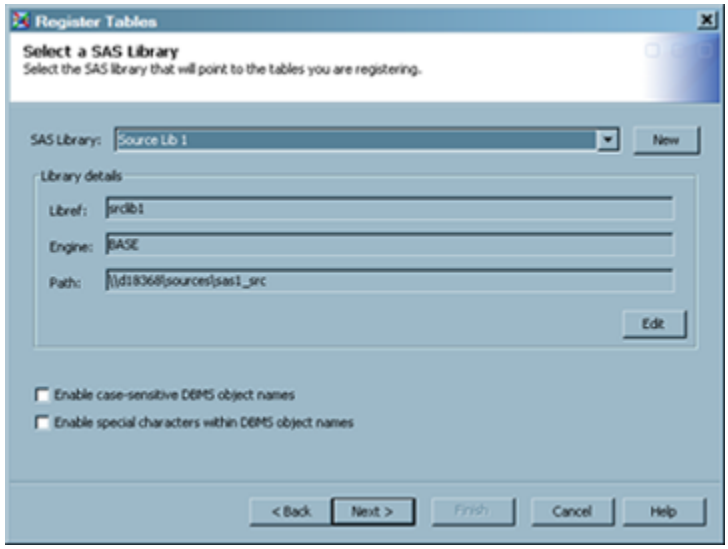
Tasks

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.)

See also "Usage Notes for Register Tables Wizards and the New Table Wizard" on page 653.

Tasks

Register a New Table with the New Table Wizard

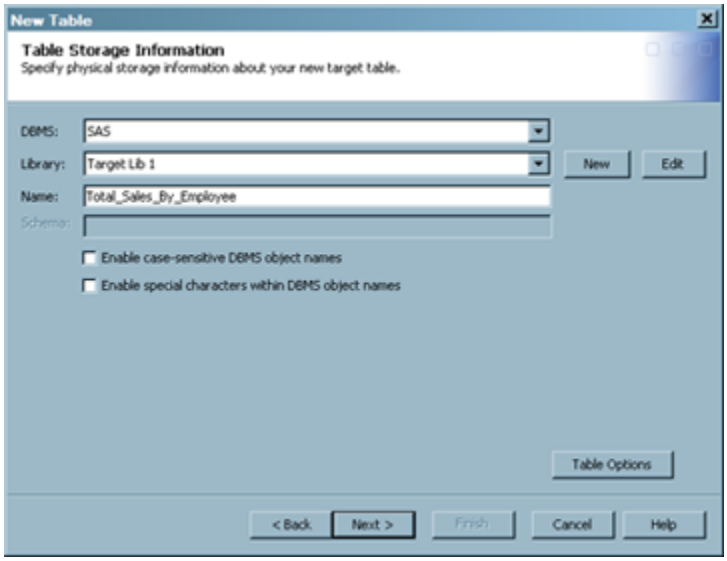
Perform the following steps to register a table that does not exist:

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*



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 the includes the table.

You can use the impact analysis and reverse impact tools in SAS Data Integration Studio to estimate the impact of these updates on your existing jobs.

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 were successfully updated and which 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.

Usage Note

The update table metadata feature cannot be used on a table until you save the job. This feature cannot be used with Hadoop tables, or on a table whose physical name includes a macro variable, such as &mstatus.OUT. For more information, see "Update Table Metadata Cannot Be Used for Some Tables" on page 651.

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 and local options for tables.

Tasks

Set Global Options for Tables

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.

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 **Enable case-sensitive DBMS object names** check box is also automatically selected.

The global settings apply to any new table metadata object, unless the settings are overridden by a local setting. For more information about DBMS object names, see "Supporting Case and Special Characters in Table and Column Names" on page 88.

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. To display most table options, display the properties window for a table and select the **Options** tab. The options available will vary according to the data format of the tables (SAS or DBMS).

You can specify other table options, such as DBMS name options, on the **Physical Storage** tab of the properties window for a table. See the help for the **Physical Storage** tab for a description of these options.

You can specify table options for the inputs and outputs of most transformations on the **Table Options** tab of the properties window for the transformation. The options available will vary according to the data format of the tables (SAS or DBMS) and whether the table is an input or an output.

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 88
- "About Case and Special Characters in DBMS Names" on page 89
- "Set Default Name Options for New Tables" on page 91
- "Set Name Options in the Register Tables Wizard" on page 91
- "Set Name Options for Registered Tables" on page 92

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 92 or "Set Default Name Options for New Tables" on page 91. 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 90 or "Enable Name Options for an Existing Database Library" on page 90. 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 90.

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 92.

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 92
- "Modify Metadata for a Column" on page 93
- "Add and Maintain Notes and Documents for a Column" on page 95
- "Perform Additional Operations on Column Metadata" on page 95

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 93.
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 4.3 *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: 1. Double-click the current name to make it editable. 2. Enter a new name of 32 characters or fewer. 3. Press the **Enter** key.

Attribute	Description	Instructions
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: 1. Double-click in the Description field. 2. Edit the description, using 200 characters or fewer. 3. Press the **Enter** key.
Type	The type can be either numeric or character.	Perform the following steps to enter the data type: 1. Double-click the current value to display the drop-down list arrow. 2. Click the arrow to make a list of valid choices appear. 3. Select a value from the list.
Length	This is the length of the column.	Perform the following steps to enter the column length: 1. Double-click the current length. 2. Enter a new length. A numeric column can be from 3 to 8 bytes long (2 to 8 in the z/OS operating environment). A character column can be 32,767 characters long. 3. Press the **Enter** key.
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: 1. Double-click the current value to display the drop-down list arrow. 2. Click the arrow to make a list of valid choices appear and then select a value from the list, or type in a new value and press **Enter**.
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.

Attribute	Description	Instructions
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 95.

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 notes or documents are associated with a column, 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, whereas 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 that you can perform on metadata in the **Columns** tab.

Table 4.4 *Additional Operations on Column Metadata*

Task	Action
Delete Metadata for a Column	Perform the following steps to delete the metadata for a column in the current table: 1. Select a column. 2. Click **Delete**. Note: When you modify or delete the metadata for a column in a table and that table is used in a SAS Data Integration Studio job, you might also have to make the same modifications to other tables in the job. For example, if you change the data type of a column and that table is used as a source in a job, then you need to change the data type of that column in the target table and in the temporary work tables in the transformations in that job. Changes to column metadata in SAS Data Integration Studio do not appear in the physical table automatically. You must select the **Replace in the Load Style** field and the **Entire table in the Replace** field on the **Load Technique** tab of the Table Loader transformation that loads the current table. 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 **Analyze**. Note that you can also obtain information about reverse impact analysis on another tab in the same window.

Task	Action
Import Metadata for a Column	Perform the following steps to import column metadata that has been added to the metadata server that is specified in your current connection profile: 1. Click **Import columns** to access the Import Columns window. 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. 3. Select the right arrow to move the selected columns into the Selected field. 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. 5. Click **OK** to import the columns into the table. Be aware of the following implications if you add or import metadata for a column: • You might need to propagate that column metadata through the job or jobs that include the current table. • Changes to column metadata in SAS Data Integration Studio do not appear in the physical table automatically. You must select the **Replace in the Load Style** field and the **Entire table in the Replace** field in the **Load Technique** tab of the Table Loader transformation that loads the current table.
Maintain Indexes	Indexes are registered automatically when using Register tables to register metadata about existing tables. Indexes are imported correctly when import/export is used. Update table metadata also updates indexes. See "Maintaining Indexes" on page 109.
Maintain Keys	Primary, foreign, and unique keys are registered automatically when using Register tables to register metadata about existing tables. Keys are imported correctly when import/export is used. Update table metadata also updates them, although currently it does not handle foreign key updates. 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 "Maintaining Keys" on page 104.
Propagate Column Metadata from One Table to Other Tables in a Job	See "Managing the Scope of Column Changes in Jobs" on page 189.

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 enable 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 whether 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 **Columns** tab, select the desired column, and then click the Properties icon in the toolbar. In the properties window, click the **Extended Attributes** tab. Use this tab to view or update extended attributes.

Standardizing Columns

Problem

You want to standardize the metadata for table columns that have the same name and that are used for the same purpose. For example, two columns named Total Sales should perhaps have the same data type and column length. Standardizing metadata can be especially useful for the target tables in SAS Data Integration Studio jobs. After you perform the standardization process, the columns in the existing table are updated the next time you run the job.

Solution

You can use the Column Standardization Tool wizard to standardize the column metadata and evaluate the effects through the use of impact analysis. The column standardization function is provided as a plug-in to SAS Management Console and SAS Data Integration Studio. The wizard helps you to update table column metadata between tables so that they match. You can use this wizard to standardize column lengths between two or more tables, formats, and other attributes that you would like to match

between the tables. Finally, you can use this feature to generate a report about column differences or log updates for audit purposes.

Perform the following tasks:

- "Select Libraries and Column Attributes" on page 99
- "Standardize Non-Standard Columns" on page 100
- "Review the Standardization Summary" on page 102
- "Review the Column Standardization Report" on page 102
- "Complete the Standardization" on page 103

Tasks

Select Libraries and Column Attributes

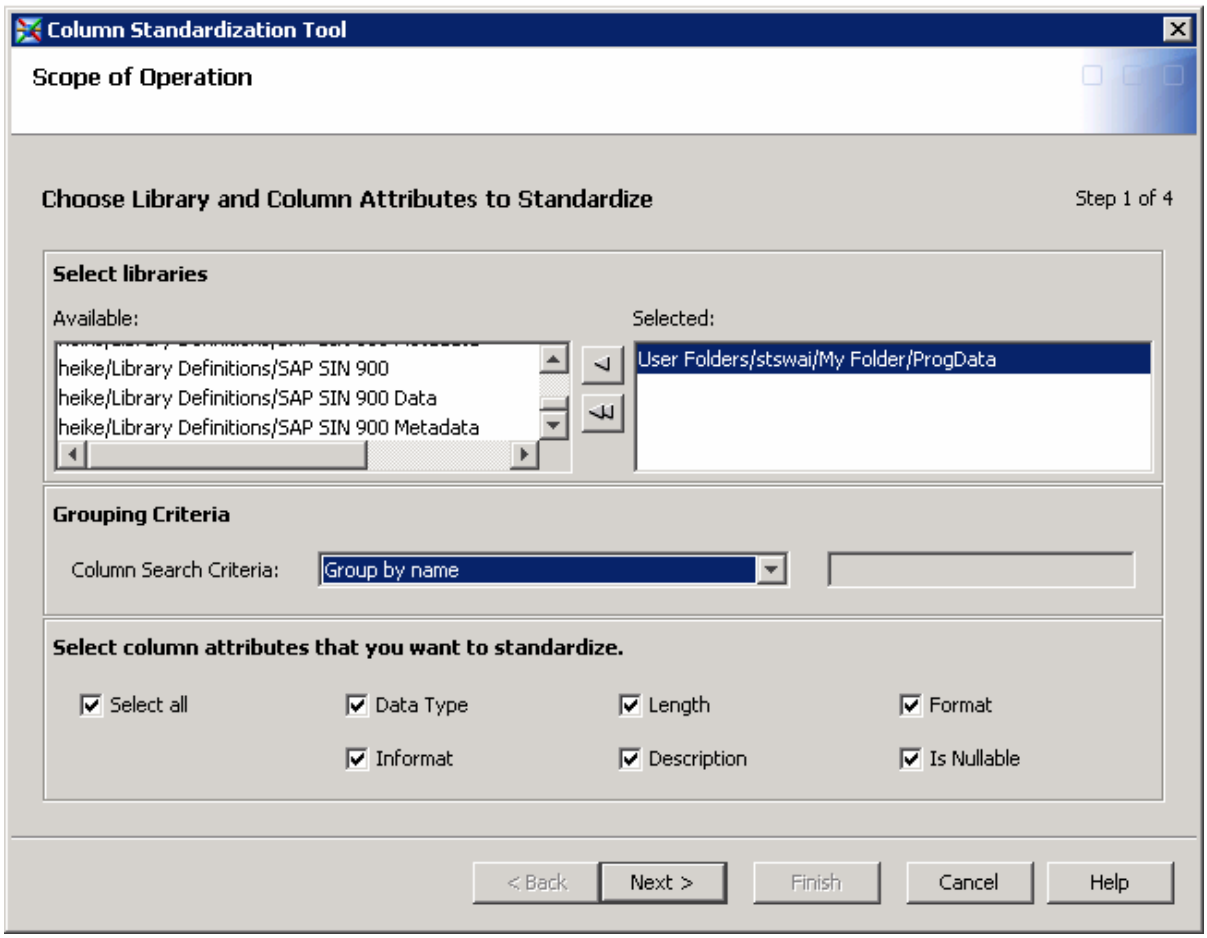
Use the Scope of Operation page to choose one or more libraries and a set of attributes to standardize.

Perform the following steps:

1. Select the libraries that you want to process for standardization and move them to the **Selected** field. For example, you can select the **ProgData** library.
2. Specify a grouping criterion such as **Group by name** in the **Column Search Criteria** field.
3. Specify the set of attributes that you want to standardize. Note that you can select **Select all** to select all of the attributes at once.

The libraries and attributes selected for a sample column standardization run are shown in the following display:

Display 4.3 *Scope of Operation*



4. Click **Next** to access the Non-standard Columns page.

Standardize Non-Standard Columns

Use the Non-standard Columns page to select the columns that you want to standardize and enter standard values.

Perform the following steps:

1. Select a column in the **Column Groups** field, which is displayed because the **Group by name** criteria was selected in the **Column Search Criteria** field. For example, you can select the **EmpID** column.

   Note: You can use the drop-down menu in the **Sort By** field. For example, you can select **By disparity** display the columns with the most disparities at the beginning of the columns list. You can also sort columns by name. Finally, you can sort by the number of tables in which the columns are used.

2. Select a row in the **Columns** table. Each row contains the data for the column in one of the tables included in the libraries that you have selected for standardization. (You should select a row that closely approximates the values that you would like to standardize, such as the row containing the EmpID column in the FLIGHTATTENDANTS table.)

3. If SAS Management Console is installed, click **Impact Analysis** to see how the selected column and table combination is used in jobs. Then you can review the jobs to ensure your planned standardizations will not affect the jobs adversely. For information about impact analysis, see "Impact Analysis and Data Lineage" on page 311.
4. Double-click the selected row to populate its values into the **Standard values** row.
5. Review the fields that you want to standardize. Edit the values in the **Standard values** row as needed.

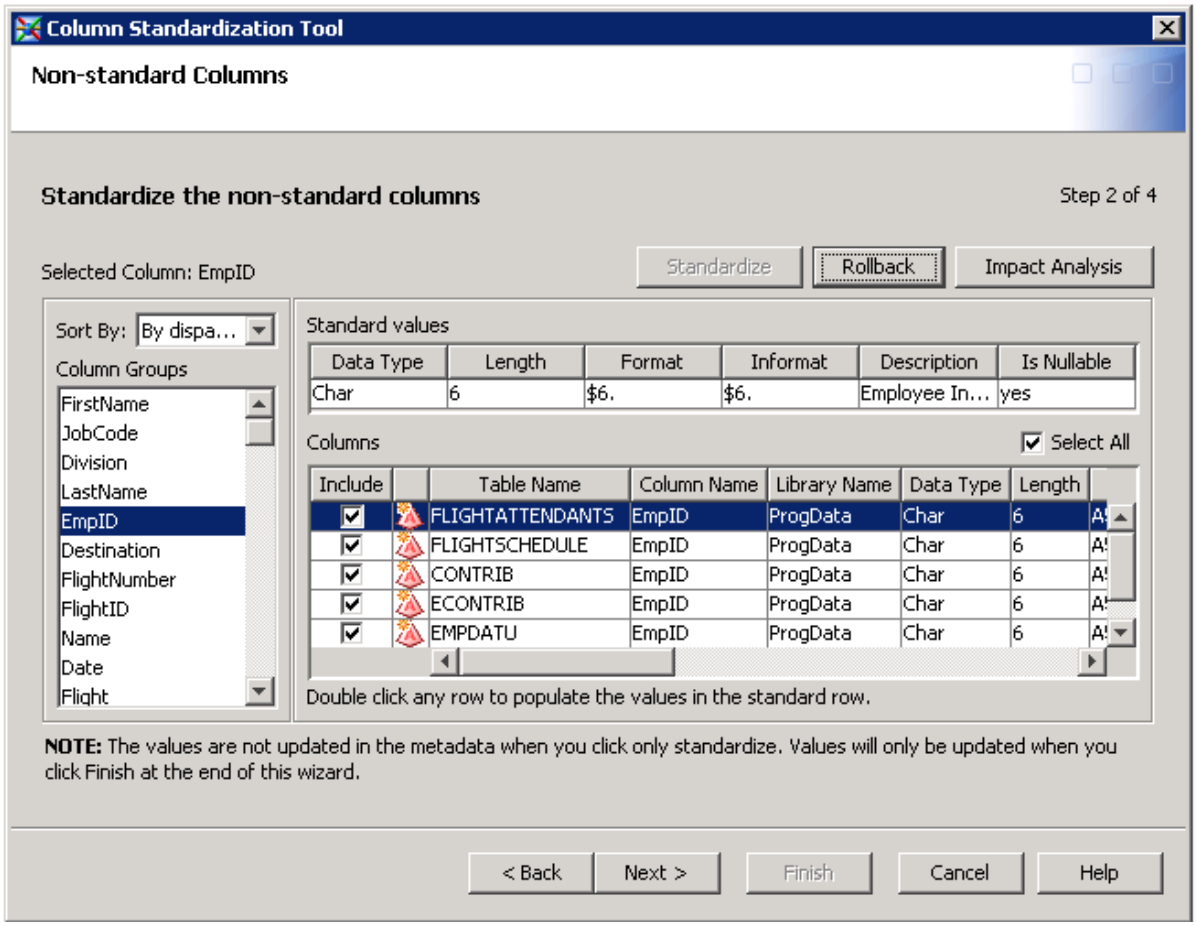
   The following standardizations were made for this example:

   - Length: 6 (was 4 for some tables)
   - Format: $6. (was missing for some tables)
   - Informat: $6. (was missing for some tables)
   - Description: Employee Identification Number (was missing for some tables)

   These values will be uniform across all of the tables in the selected libraries after the standardization is applied.
6. Click **Standardize** to apply the standardization. Note that the metadata will be changed only at the end of the wizard.
7. Review the results of the standardization in the **Columns** table.

   These results are shown in the following display:

Display 4.4 *Non-Standard Columns*



Note that you can click **Rollback** to reverse the standardization of the selected column.

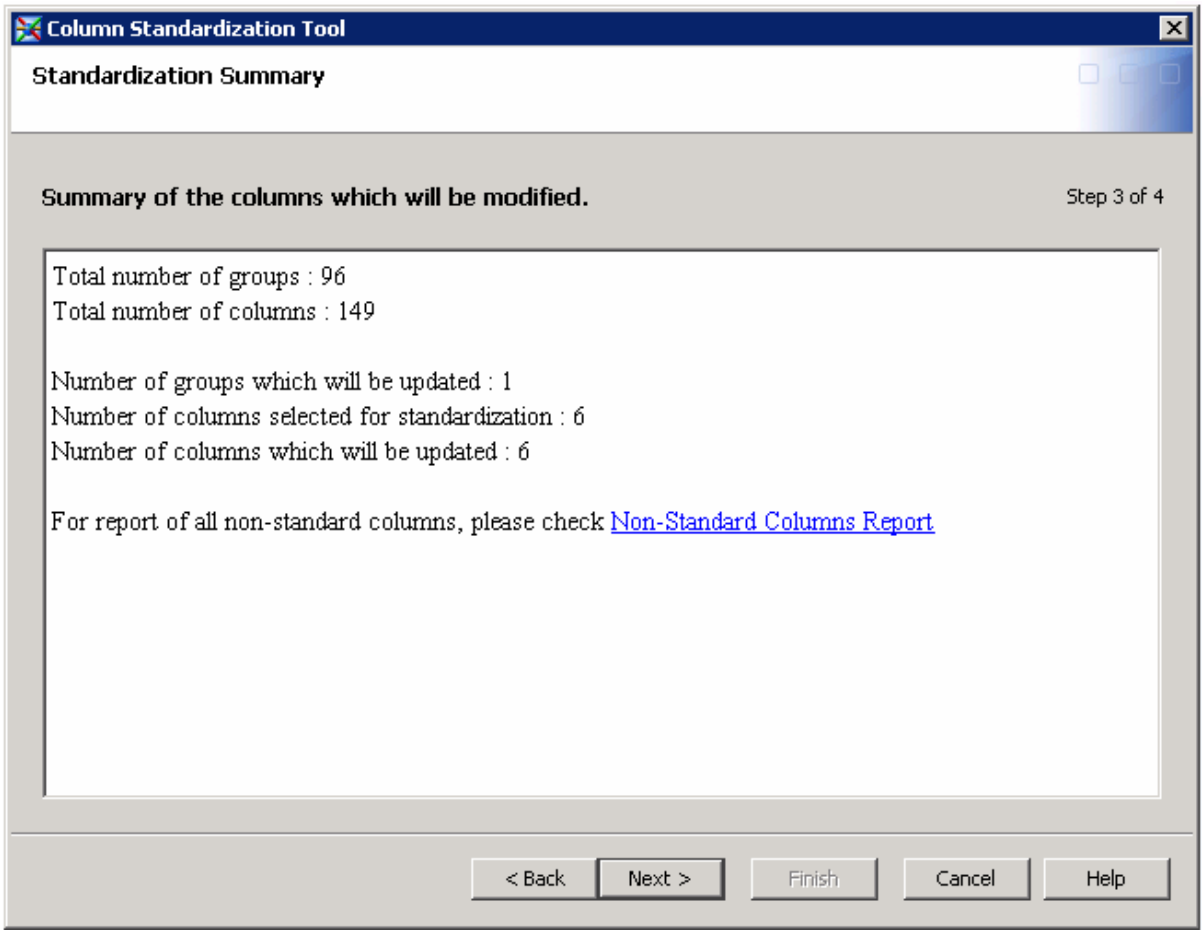
8. Repeat the standardization process for the other columns in the **Column Groups** field.
9. Click **Next** to access the Standardization Summary page.

Review the Standardization Summary

Use the Standardization Summary page to display a summary of the columns that will be modified.

The following display shows the summary for the sample standardization:

Display 4.5 *Standardization Summary*



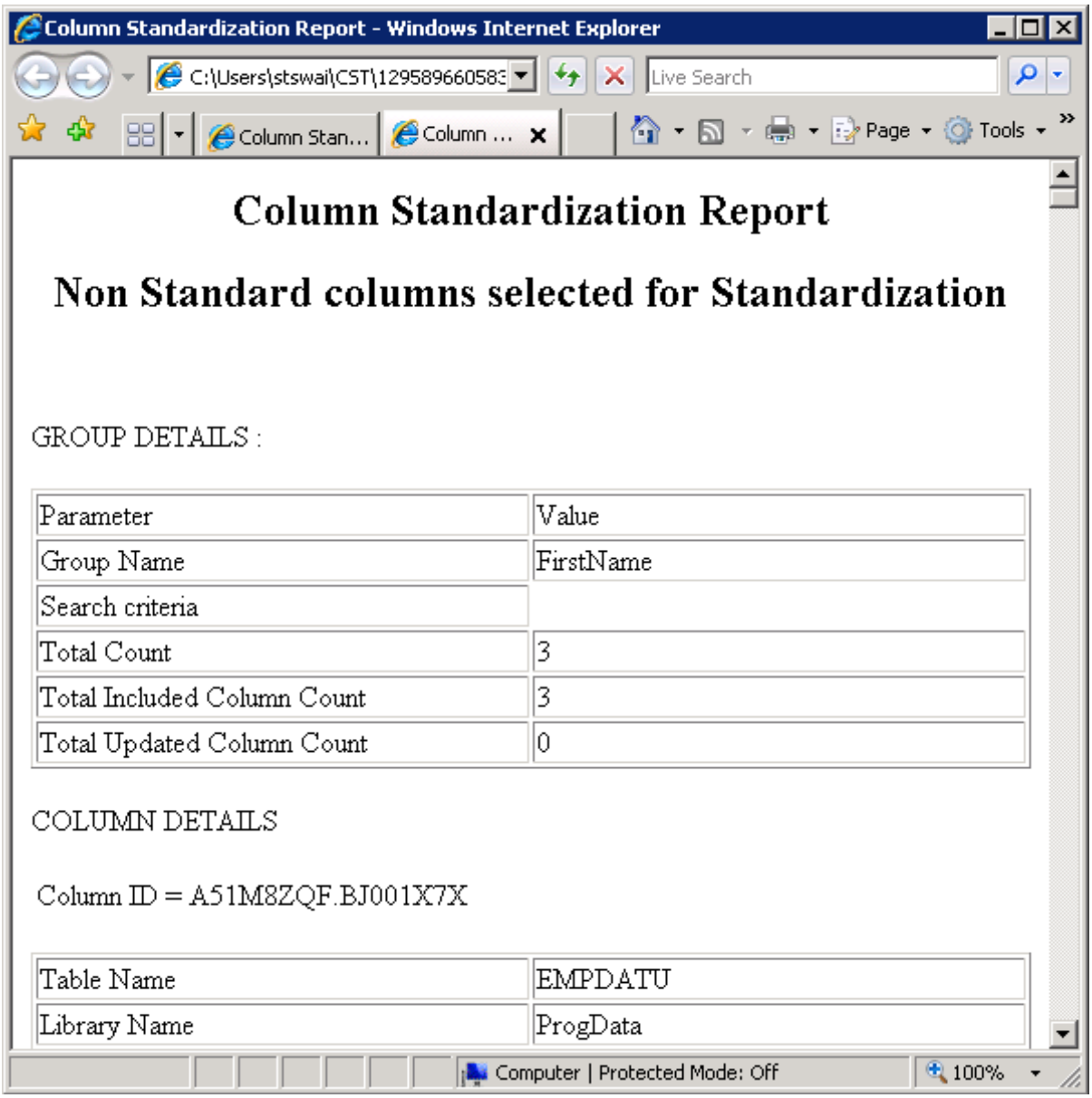
Click **Non-Standard Columns Report** to see a detailed report of the changes. Note that this report is optional. It contains a list of all of the nonstandard columns found in the metadata search. This search is performed using the search criteria user specified in first tab of the wizard.

Review the Column Standardization Report

Use the Column Standardization Report to review a detailed listing of the changes included in the standardization process.

The following display shows the report for the sample standardization.

Display 4.6 *Column Standardization Report*



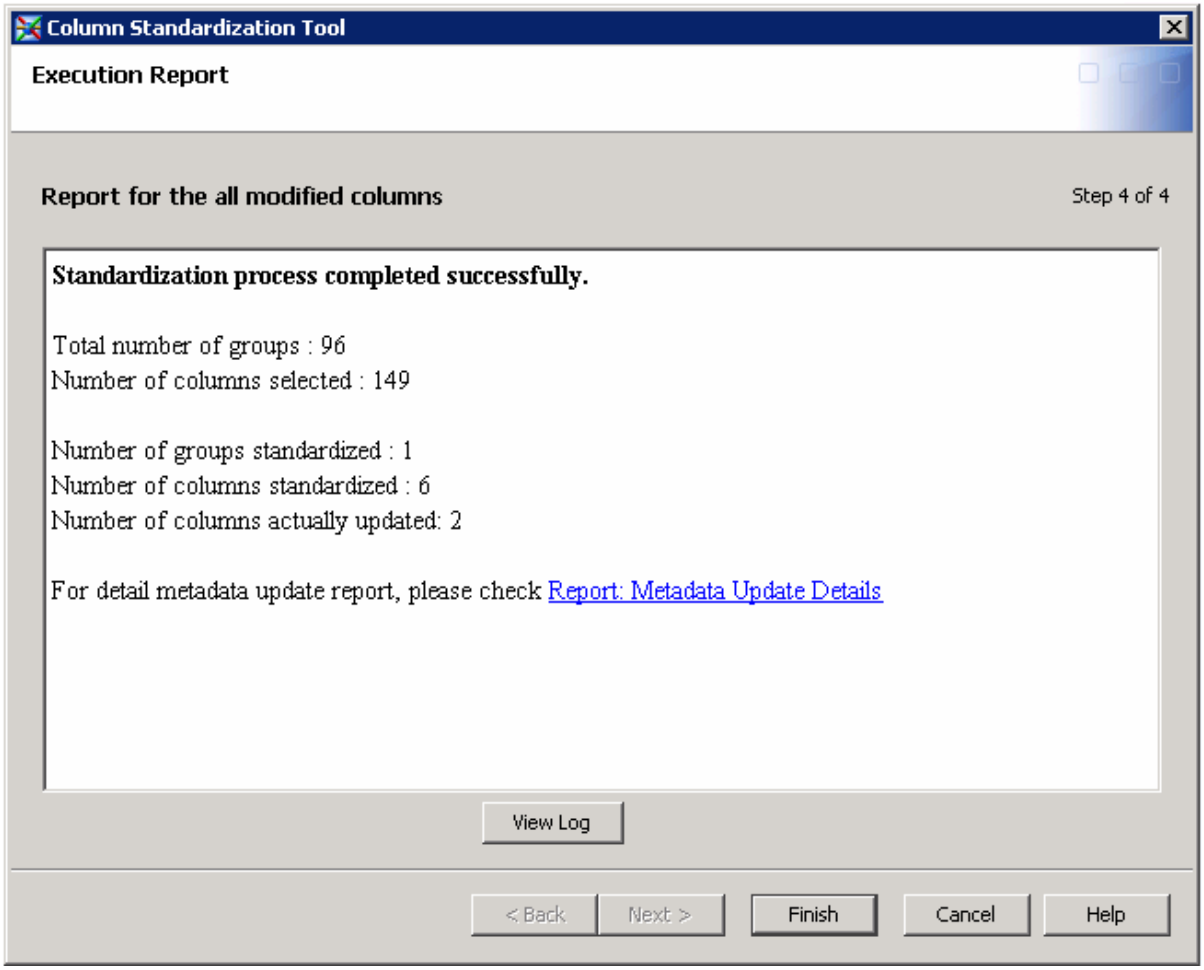
After you have reviewed the report, click **Next** to complete the standardization process and display the Execution Report page. Note that the metadata is updated at this point.

Complete the Standardization

Use the Execution Report page to confirm that the standardizations were successfully executed.

The following screen shows the Execution Report page for the sample standardization:

Display 4.7 *Execution Report*



Note that you can click **Report: Metadata Update Details** to display the report. This report contains a list of the columns involved in the actual standardization process. The Non-Standard Columns Report and the Report: Metadata Update Details are located in the following location: **`<User location>\CST\<Folder with timestamp>`**.

You can also review a log of the standardization process. Finally, click **Finish** to close the Column Standardization Tool wizard.

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 105. Then perform the following tasks as needed:

- "View Keys" on page 105
- "Add a Primary Key or a Unique Key" on page 107
- "Add a Foreign Key" on page 108
- "Update the Columns in a Key" on page 109
- "Delete or Rename a Key" on page 109

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.

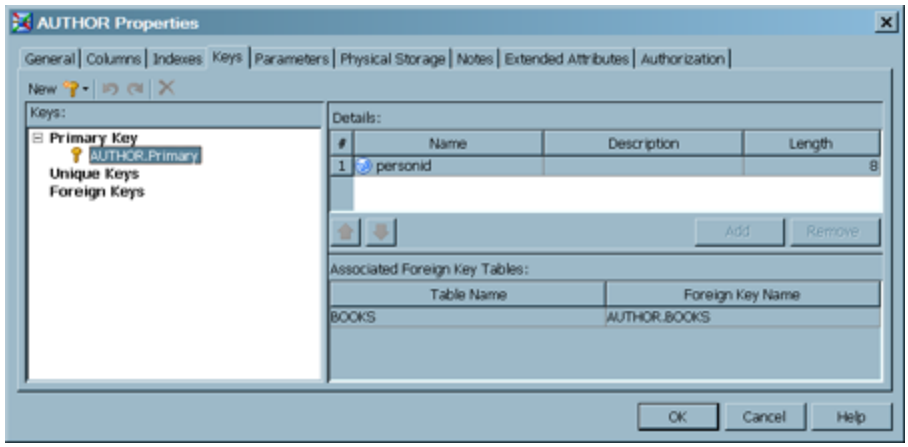
Primary keys and unique keys are often used in table joins. A foreign key is used to create and enforce a link between the data in two tables. A link is created between two tables such that the column or columns that hold a primary key value or a unique key value in one table are referenced by a column 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 4.8 *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.

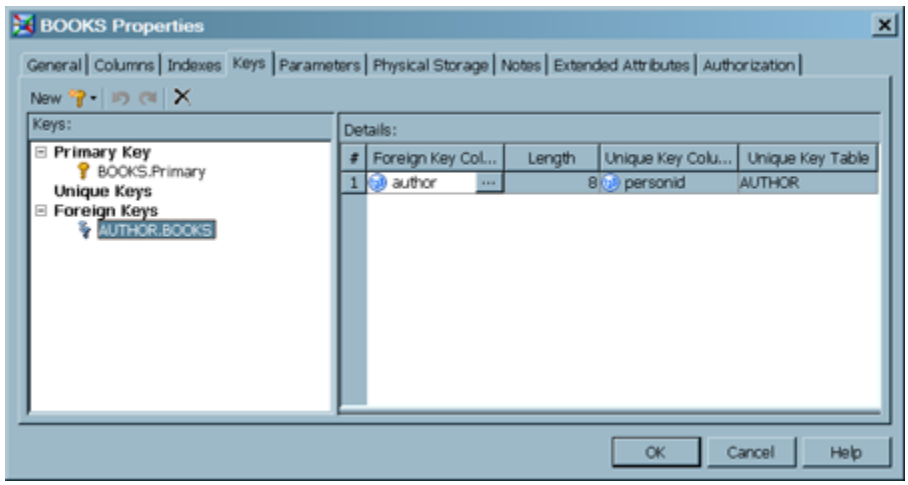
The default name for a unique key is **`currentTableName`**.*UniqueKey***`N`**, where **`currentTableName`** is the name of the current table, *UniqueKey* is a literal string, and **`N`** is an iteration number added to the end.

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 4.9 *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:

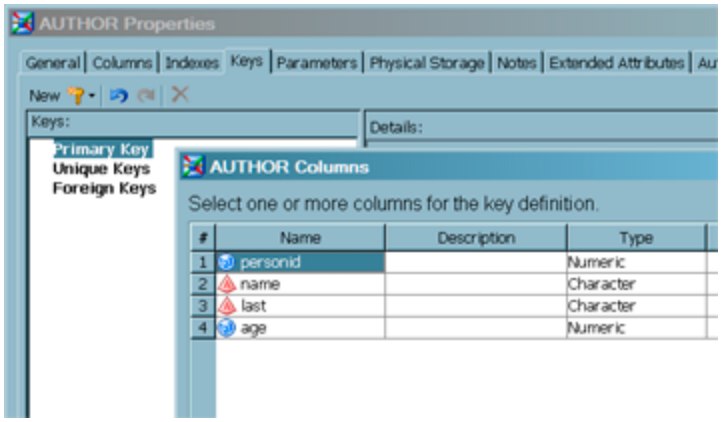
- **Foreign Key Column** displays the column or combination of columns in the current table that references the corresponding column or combination of columns in another table. In the preceding display, the foreign key column is named **author**, which is the name of a column in the BOOKS table.
- **Length** 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 4.10 *Selecting a Column for a Primary Key*



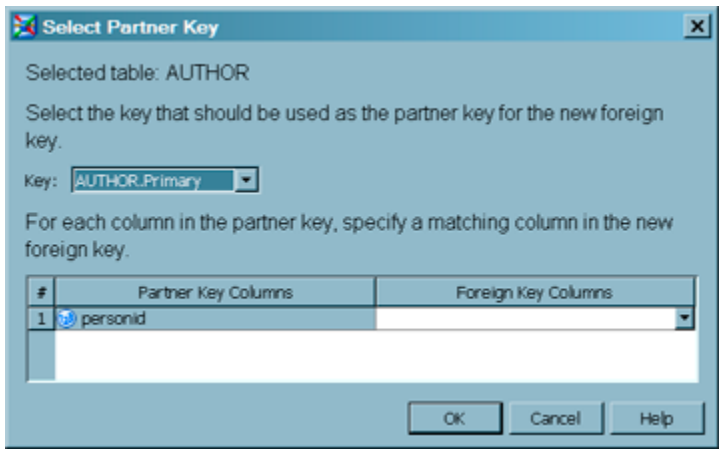
4. Click **OK** to save the selected columns in the metadata for a key. The new key is displayed in the Details pane.
5. Click **OK** to save the key.

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:

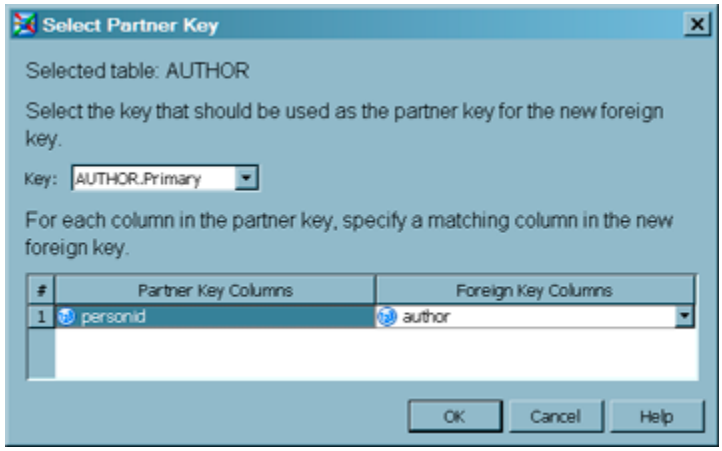
1. Access the **Keys** tab on the properties window for the table that requires a foreign key. For example, if you want to create a foreign key in the BOOKS table that references the primary key column in the AUTHORS table, then open the properties window for the BOOKS table.
2. Right-click **Foreign 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 4.11 *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 4.12 *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 110
- “Delete an Index or a Column” on page 110
- “Rearrange the Columns in an Index” on page 111

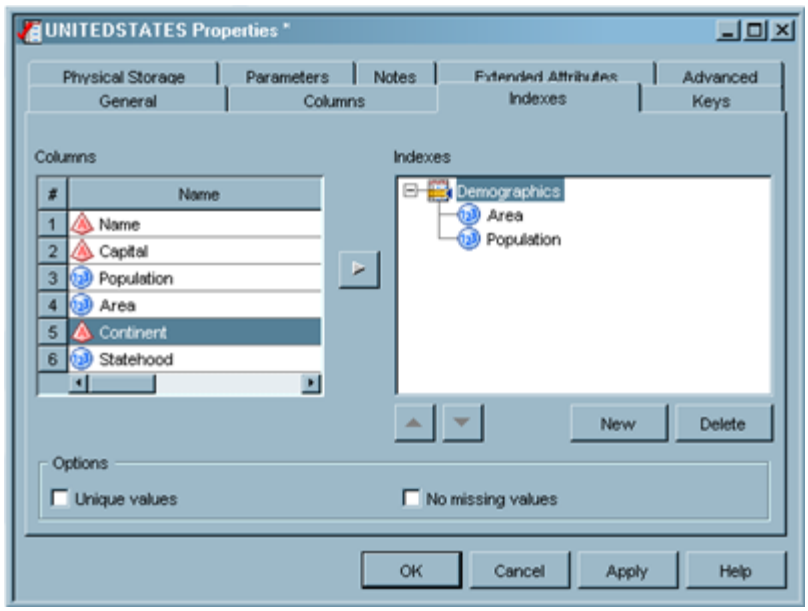
Tasks

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 4.13 *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.

See also "Usage Notes for the View Data Window" on page 658.

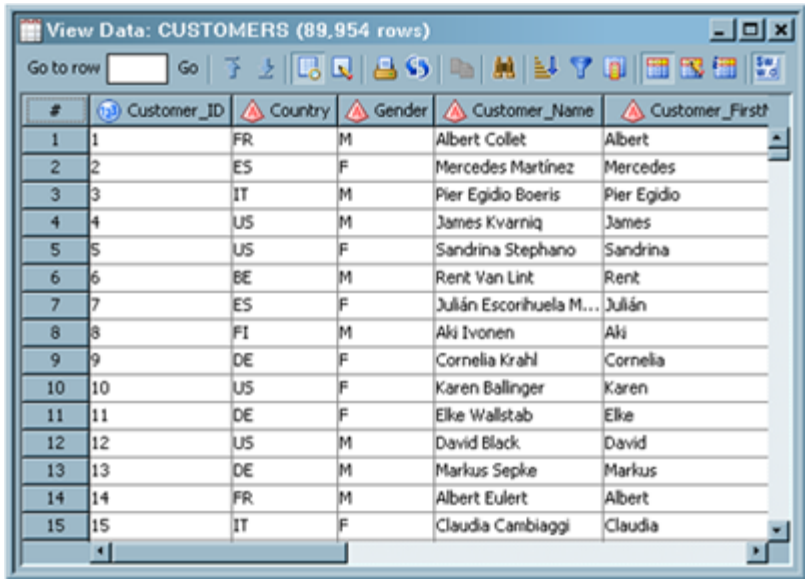
Tasks

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 4.14 *View Data Window in Browse Mode*



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

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

Browse Functions

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

Display 4.15 *View Data Browse Toolbar*



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

Table 4.5 *Browse Functions in the View Data Window*

Task	Action
Navigate within the data	Perform the following steps: • Enter a row number in the **Go to row** field and click **Go to row** to specify the number of the first row that is displayed in the table. • Click **Go to first row** to navigate to the first row of data in the View Data window. • Click **Go to last row** to navigate to the last row of data in the View Data window.

Task	Action
Select a View Data window mode	Perform the following steps: • Click **Switch to browse mode** to switch to the browse mode. • Click **Switch to edit mode** to switch to the edit mode. Note that the **Switch to browse mode** and **Switch to edit mode** buttons are displayed only for SAS tables.
Perform utility functions	Perform the following steps: • Click **Print** to print the View Data window. • Click **Refresh** to refresh the data in the View Data window.
Copy one or more rows of data into the copy buffer	Perform the following steps: • Highlight one or more rows of data. Then, click **Copy** to copy the selected text into the copy buffer.
Manipulate the data that is displayed in View Data window	Perform the following steps: • Click **Show search pane**. Then, use the search toolbar to search for string occurrences in the data set that is currently displayed in the View Data window. • Click **Launch sort screen**. Then, use the **Sort By Columns** tab in the Query Options window to specify a sort condition on multiple columns. The sort is performed on the data set that is currently displayed in the View Data window. • Click **Filter**. Then, use the **Filter** 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. • Click **Subset columns**. Use the **Columns** 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 **Columns** field. The redrawn View Data window includes only the columns that you select here on the **Columns** tab.

Task	Action
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. • Click **Show column name in column header** to display physical column names in the column headings. • Click **Show column description in column header** to display optional descriptions in the column headings. • Click **Show column metadata name in column header** 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.
Determine whether metadata formats are applied	Perform the following steps: • Click **Apply metadata formats** to toggle between showing formatted and unformatted data in the View Data window.

To sort columns and perform related tasks, right-click on a column name and select an appropriate option from the pop-up menu. To 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 118.

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.

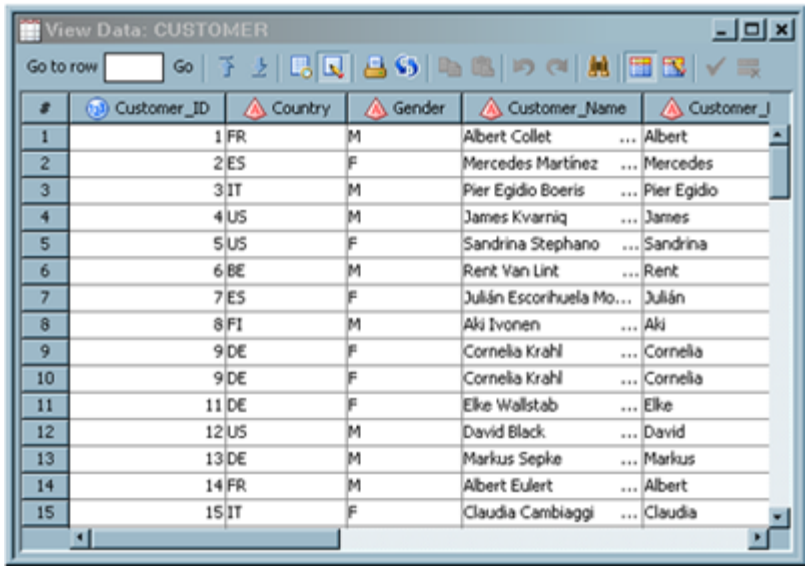
Tasks

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.
2. Enter the appropriate user ID and password, if you are prompted for them. The information in the table displays in the browse mode of the View Data window.
3. Click **Switch to edit mode** on the view data toolbar. The View Data window displays in edit mode, as shown in the following display.

Display 4.16 *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 Martínez	Mercedes
3	3	IT	M	Pier Egidio Boeris	Pier Egidio
4	4	US	M	James Kvarniq	James
5	5	US	F	Sandrina Stephano	Sandrina
6	6	BE	M	Rent Van Lint	Rent
7	7	ES	F	Julián Escorihuela Mo...	Julián
8	8	FI	M	Aki Ivonen	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.

4. 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.
5. Click **Undo last action** to reverse the change that you just made. (You can click **Redo last action** to return to the changed version of the cell.) Note that you can undo only the last operation because only a single level of undo is supported. 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.
6. Click a row number to select the row. Click **Copy** to copy the row into the buffer.
7. Click **Go to last row** to move to the last row in the table.
8. Click in the row marked by the New Row icon at the end of the View Data window. The New Row icon changes to the Editing Row icon. Click **Paste** to paste the copied data into the row.

   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.

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

Edit Tasks

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

Display 4.17 *View Data Edit Toolbar*



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

Table 4.6 *Edit Functions in the View Data Window*

Task	Action
Navigate within the data	Perform the following steps: • Enter a row number in the **Go to row** field and click **Go** to specify the number of the first row that is displayed in the table. • Click **Go to first row** to navigate to the first row of data in the View Data window. • Click **Go to last row** to navigate to the last row of data in the View Data window.
Select a View Data window mode	Perform the following steps: • Click **Switch to browse mode** to switch to the browse mode. • Click **Switch to edit mode** to switch to the edit mode. Note that the **Switch to browse mode** and **Switch to edit mode** buttons are displayed only for SAS tables.
Perform utility functions	Perform the following steps: • Click **Print** to print the View Data window. • Click **Refresh** to refresh the data in the View Data window.
Copy or paste data	Perform the following steps: • Highlight one or more rows of data. Then, click **Copy** to copy the selected text into the copy buffer. • Place the cursor in the row where you want to place the data. Then, click **Paste** to paste the data into the table. Note that you can also use **Paste Special** to paste more at once.

Task	Action
Undo or redo editing operations	Perform the following steps: • Click **Undo last action** to reverse the most recent editing operation. • Click **Redo last action** to restore the results of the most recent editing operation.
Search the data displayed in View Data window	Perform the following steps: • Click **Show search pane**. Then, use the search toolbar to search for string occurrences in the data set that is currently displayed in the View Data window.
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. • Click **Show column name in column header** to display physical column names in the column headings. • Click **Show column description in column header** to display displays optional descriptions in the column headings.
Commit or delete editing changes	Perform the following steps: • Click **Save edited row** to commit the changes that you have made to the currently edited row. • Click **Delete selected rows** to delete the changes that you have made to the currently edited row.

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

To 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.

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.

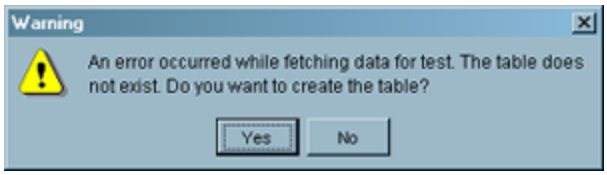
Tasks

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

Display 4.18 *Create Table Dialog Box*



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

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.

Chapter 5

Working with External Files

About External Files **120**

Registering a Delimited External File **120**
Problem 120
Solution 121
Tasks 121

Registering a Fixed-Width External File **124**
Problem 124
Solution 124
Tasks 124

Registering an External File with User-Written Code **128**
Problem 128
Solution 128
Tasks 128

Viewing or Updating External File Metadata **131**
Problem 131
Solution 131

Overriding the Code Generated by the External File Wizards **132**
Problem 132
Solution 132
Tasks 132

Specifying NLS Support for External Files **133**
Problem 133
Solution 133
Tasks 133

Accessing an External File with an FTP Server or an HTTP Server **133**
Problem 133
Solution 134
Tasks 134
Additional Information 134

Viewing Data in External Files **135**
Problem 135
Solution 135
Tasks 135

Registering a COBOL Data File That Uses a COBOL Copybook **136**
Problem 136
Solution 136
Tasks 136

Using an External File in the Process Flow for a Job 138
Problem 138
Solution 138
Tasks 138

Using a Format File to Register a Fixed-Width External File 141
Problem 141
Solution 141
Tasks 142

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 "Registering a Delimited External File" on page 120, "Registering a Fixed-Width External File" on page 124 , and "Registering an External File with User-Written Code" on page 128.
Specify a registered external file as a source or a target in a job.	For more information, see "Using an External File in the Process Flow for a Job" on page 138.
View the data or metadata for a registered external file.	For more information, see "Viewing Data in External Files" on page 135 and "Viewing or Updating External File Metadata" on page 131.

Registering a Delimited External File

Problem

You want to use a delimited external file in a SAS Data Integration Studio job.

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, where SAS Data Integration Studio can access it.

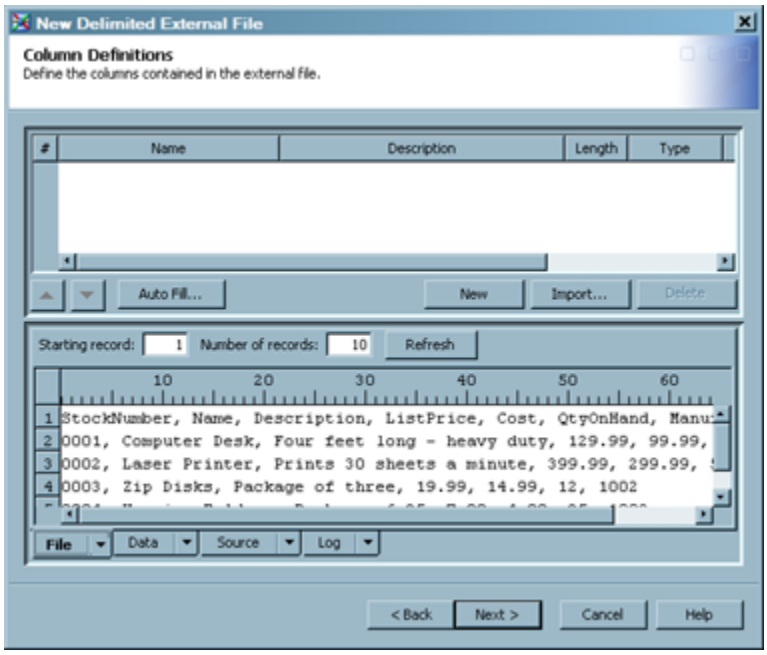
Tasks

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*

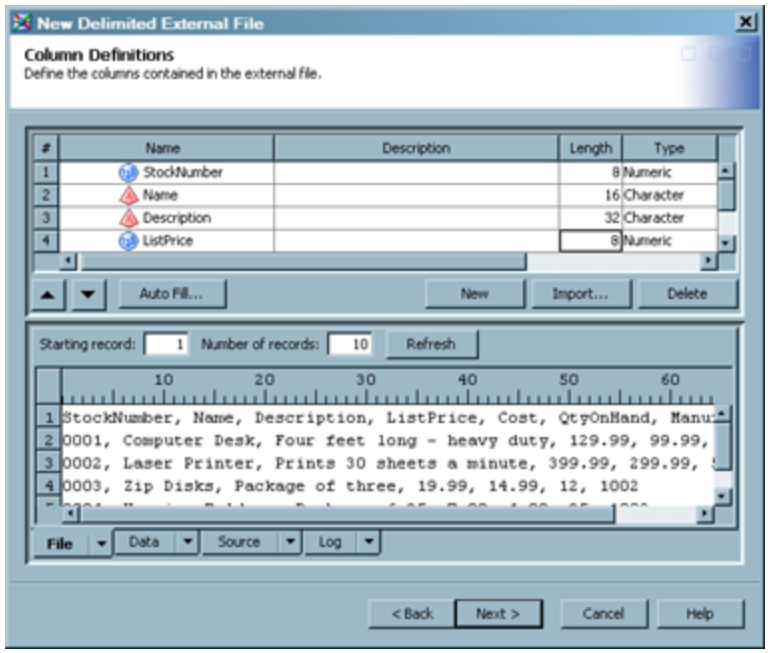


Note: If your external file contains fewer than 10 rows, a warning box is displayed. Click **OK** to dismiss the warning window.

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*



12. To verify that the metadata that 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.

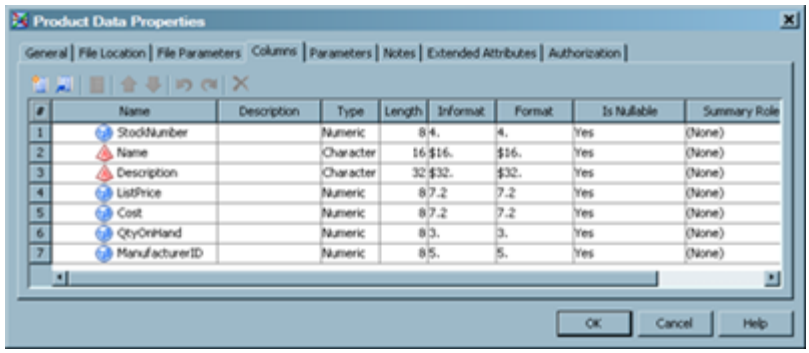
13. Click **Next** and then **Finish** to save the metadata and exit the delimited external file wizard.

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*



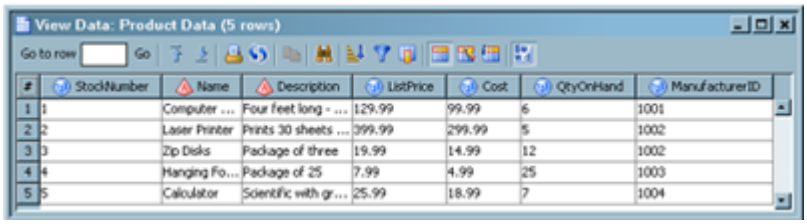
#	Name	Description	Type	Length	Informat	Format	Is Nullable	Summary Role
1	StockNumber		Numeric	8	4.	4.	Yes	(None)
2	Name		Character	16	$16.	$16.	Yes	(None)
3	Description		Character	32	$32.	$32.	Yes	(None)
4	ListPrice		Numeric	8	7.2	7.2	Yes	(None)
5	Cost		Numeric	8	7.2	7.2	Yes	(None)
6	QtyOnHand		Numeric	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	6	1001
2	2	Laser Printer	Prints 30 sheets ...	399.99	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

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 use a fixed-width external file in a SAS Data Integration Studio job.

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, where it can be accessed by SAS Data Integration Studio.

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.

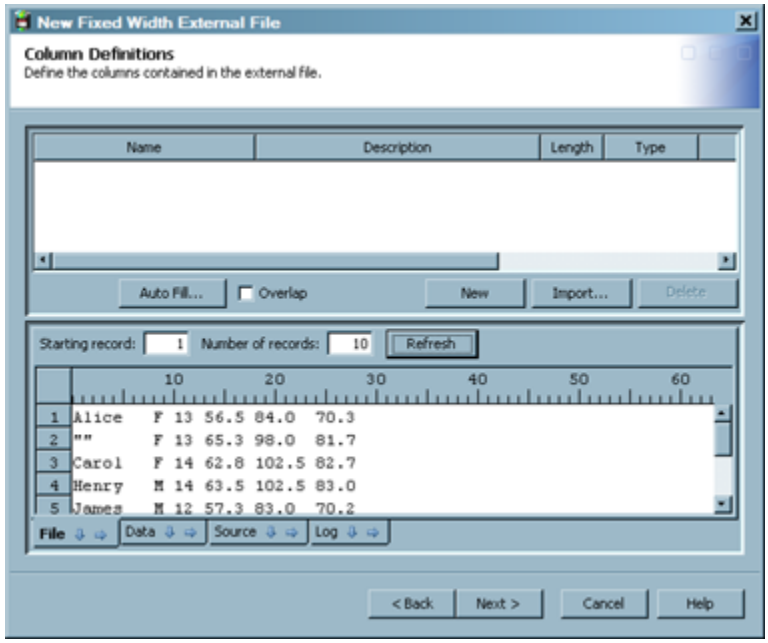
Tasks

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.

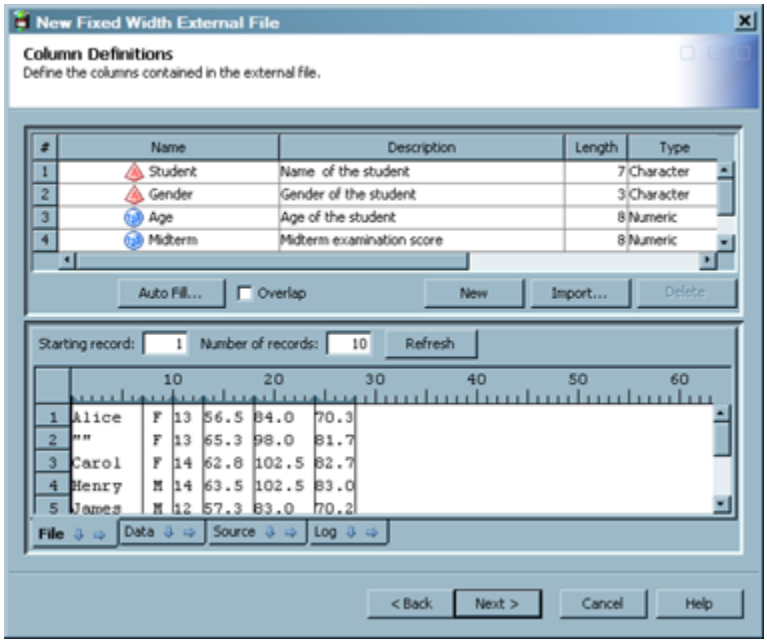
8. Click the appropriate tick marks in the ruler for the other columns in the external file. Break lines and metadata for these columns are set.

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*



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.

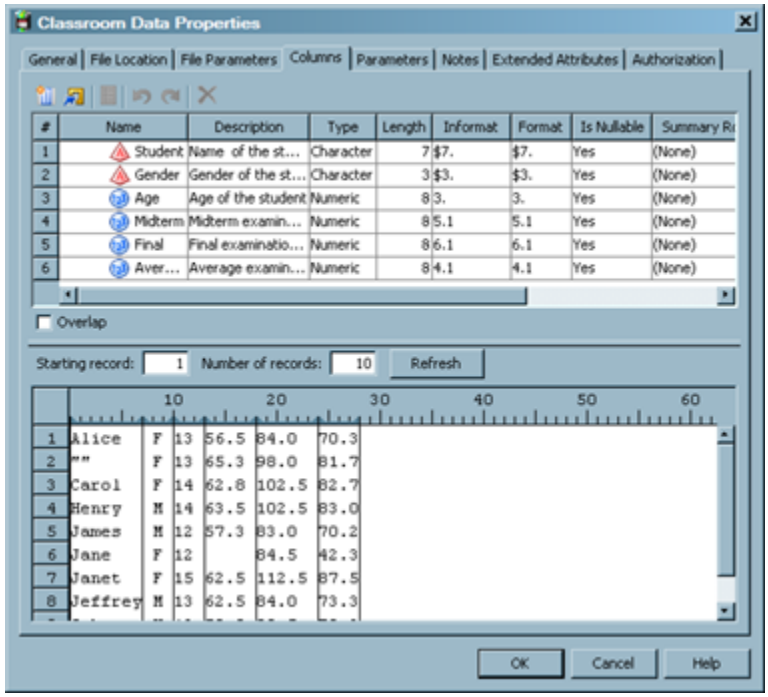
11. Click **Next** and **Finish** to save the metadata and exit the fixed-width external file wizard.

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*

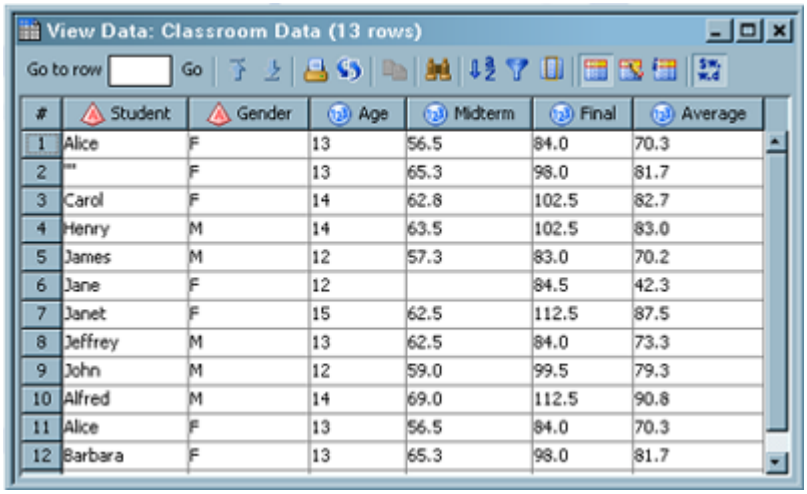


2. Click **OK** to save any changes and close the properties window.

View the Data

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

Display 5.8 *Sample External File Data in the View Data Window*



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;
```

```
run;
```

Note: The first two lines of this SAS code are entered to set the LIBNAME and output parameters that the SAS code needs to process the external file. After you have verified that the code ran successfully, delete 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

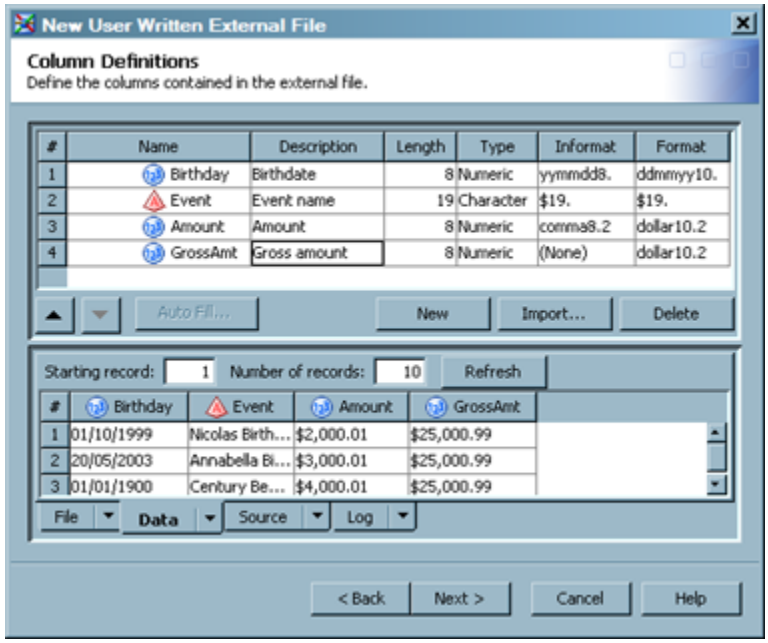
Perform the following steps to use one method to register an external file in the user-written wizard:

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 **`yymmdd8.`** informat, and the **`ddmmyy10.`** format. Click **New** to add a row to the columns component at the top of the Column Definitions window.
7. Review the data after you have defined all of the columns. To view this data, click the **Data** tab under the view pane at the bottom of the window. To view the code that 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*



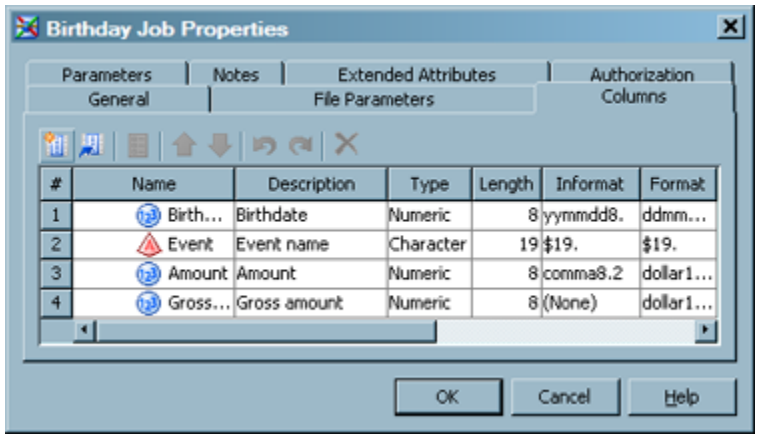
8. Click **Next** to access a summary page, and then click **Finish** to save the metadata and exit the user written external file wizard.

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:

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 5.10 *External File Columns Tab*

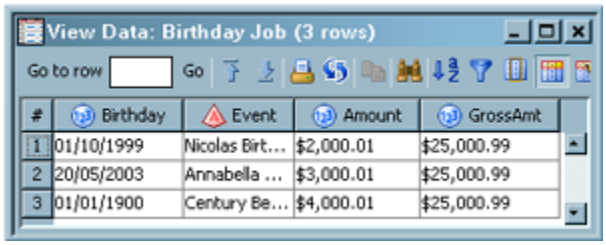


2. Click **OK** to save any changes and close the properties window.

View the Data

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

Display 5.11 *External File Data in the View Data Window*



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 the includes the external file.

You can use the impact analysis and reverse impact tools in SAS Data Integration Studio to estimate the impact of these updates on your existing jobs.

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 nonstandard 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 nonstandard access method, see "Accessing an External File with an FTP Server or an HTTP Server" on page 133.

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 133.)
- 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 120. 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 120.

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:wky.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:dwatest.tst.v913.cffd`**.
- The COB2SAS programs must reside in a PDS with certain characteristics. For more information about these characteristics, see **`http://support.sas.com/techsup/technote/ts536.html`**.
- The path to the **`r2cob1.sas`** program should specify the PDS and member name. Here is an example path, which would be specified in the **Full path for r2cob1.sas** field in the Advanced options window: **`mvs:dwatest.tst.v913.cob2sasp(r2cob1)`**.

Copy Column Metadata From the COBOL Format File

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:

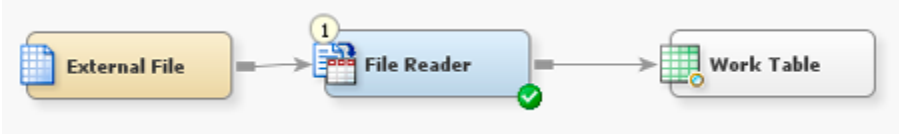
- "Read from an External File in a Job" on page 138
- "Write to an External File in a Job" on page 139
- "Run the Job and Verify the Results" on page 140

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 view. Most SAS Data Integration Studio transformations can read a SAS view, 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.

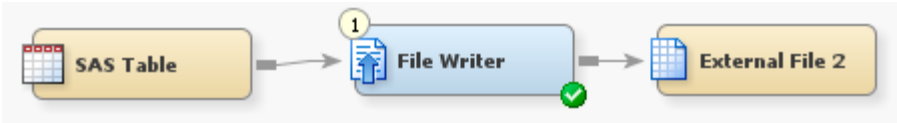
Perform the following steps to specify an external file as the input to the File Reader transformation.

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 120, "Registering a Fixed-Width External File" on page 124, and "Registering an External File with User-Written Code" on page 128.
2. Create an empty SAS Data Integration Studio job. For more information, see "Creating an Empty Job" on page 147.
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 5.13 *File Writer Process Flow*



The File Writer transformation reads information from a SAS or DBMS table and writes the output to an external file. The input to a File Writer transformation could be the output of a previous transformation in the current job, or it could be output from another job. In this way, the output of SAS Data Integration Studio jobs can be made available to third-party applications that support external files.

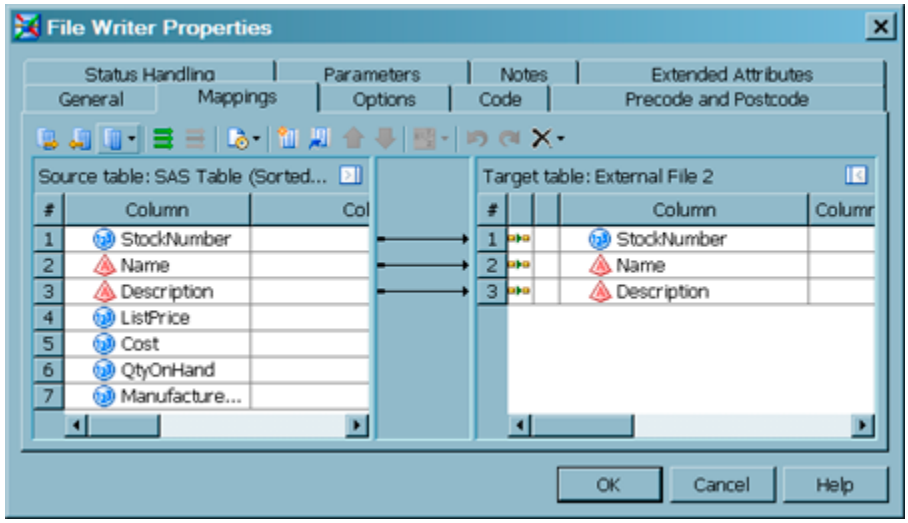
Assume that the SAS or DBMS table input to the File Writer transformation is already registered, and that the external file output is a new file, one that is created when the job that includes the File Writer executes for this first time. Perform the following steps to specify an external file as the output of the File Writer 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 120, "Registering a Fixed-Width External File" on page 124, and "Registering an External File with User-Written Code" on page 128.
2. Create an empty SAS Data Integration Studio job. For more information, see "Creating an Empty Job" on page 147.
3. Select and drag a File Writer 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 a SAS or DBMS input table from the tree view. Then, drop it before the File Writer transformation on the **Diagram** tab.

5. Drag the cursor from the input table to the input port of the File Writer transformation. This action connects the input to the transformation.
6. Select and drag the external file output from the tree view. Then, drop it after the File Writer transformation on the **Diagram** tab.
7. Drag the cursor from the output port of the File Writer transformation to the external file. This action connects the output to the transformation. At this point, the process flow should look similar to the preceding process flow diagram.

   The File Writer transformation attempts to automatically map columns between the input table and the output external file. You might want to verify that the mappings are correct.
8. (Optional) To verify the mappings in the File Writer transformation, right-click the transformation in the job and select **Properties** from the pop-up menu. The next display shows the **Mapping** tab for the File Writer transformation.

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 185.

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.

Using a Format File to Register a Fixed-Width External File

Problem

You want to use a fixed-width external file in a SAS Data Integration Studio job. You also want to minimize the amount of column metadata that you must manually specify for the external file.

Solution

Create an external format file that specifies the column metadata for the external data file. In SAS Data Integration Studio, run the fixed-width external file wizard and specify both the data file and the format file. The wizard uses the format file to register the column metadata for the data file. This reduces the need to manually specify column metadata for the data file.

An external format file describes the structure of the columns in an external data file. The format file must be a well-formed file that the SAS INFILE statement can read.

For example, the following portion of a format file for a fixed-width data file contains census data. The format file is in comma-separated-values (CSV) format.

```
Name,SASColumnType,BeginPosition,EndPosition,ReadFlag,Desc,SASFormat,SASInformat
RECTYPE,C,1,1,y,Record Type,$char.,$char.
SERIALNO,C,2,8,y,Serial #: Housing Unit ID,$char.,$char.
SAMPLE,C,9,9,y,Sample Identifier,,
DIVISION,C,10,10,y,Division code,,
STATE,C,11,12,y,State Code,,
PUMA,C,13,17,y,Public Use Microdata Area (State Dpndnt),,
AREATYPE,C,18,19,y,Area Type Rev. for PUMS Equavalency fl,,
MSAPMSA,C,20,23,y,MSA/PMSA,,
PSA,C,24,26,y,PLANNING SRVC AREA (ELDERLY SAMPLE ONLY),,
SUBSAMPL,C,27,28,y,SUBSAMPLE NUMBER (USE TO PULL EXTRACTS),,
HOUSWGT,N,29,32,y,Housing Weight,,
PERSONS,N,33,34,y,Number of person records this house,,
...
```

The values in the first row are SAS column attributes. The values of subsequent rows specify metadata for the columns in the external file, in this case a fixed-width file that contains census data. Here is a description of the SAS column attributes in the first row.

Name
: A logical identifier for the object, in this case a column name, such as RECTYPE and SERIALNO.

SASColumnType
: This represents the SAS type (character or numeric) for this column. The value can be either 'C' or 'N'.

BeginPosition
: The position within a record where the column begins. This is used for external tables and record-oriented tables.

EndPosition
: The position within the record where the column ends. This is used for external tables and record-oriented tables.

ReadFlag
: Indicates whether to read the column. If set to N, the column is ignored when the data is read in.

Desc
: Brief description of the object, in this case a column.

For a full description of SAS column attributes, see the topics for the Column type and the Logical Column type in the *SAS Metadata Model: Reference*. The version of this book for SAS 9.3 applies to both SAS 9.3 releases and SAS 9.4 releases. This book can be accessed from the "Documentation by Title" section of the SAS Product Documentation page: **`http://support.sas.com/documentation/`**.

Note: If your external format file does not specify SAS informats for all column variables, you need to specify these manually in SAS Data Integration Studio. If appropriate informats are not provided for all columns, then incorrect results can be encountered when the external file is used in a job or when its data is viewed.

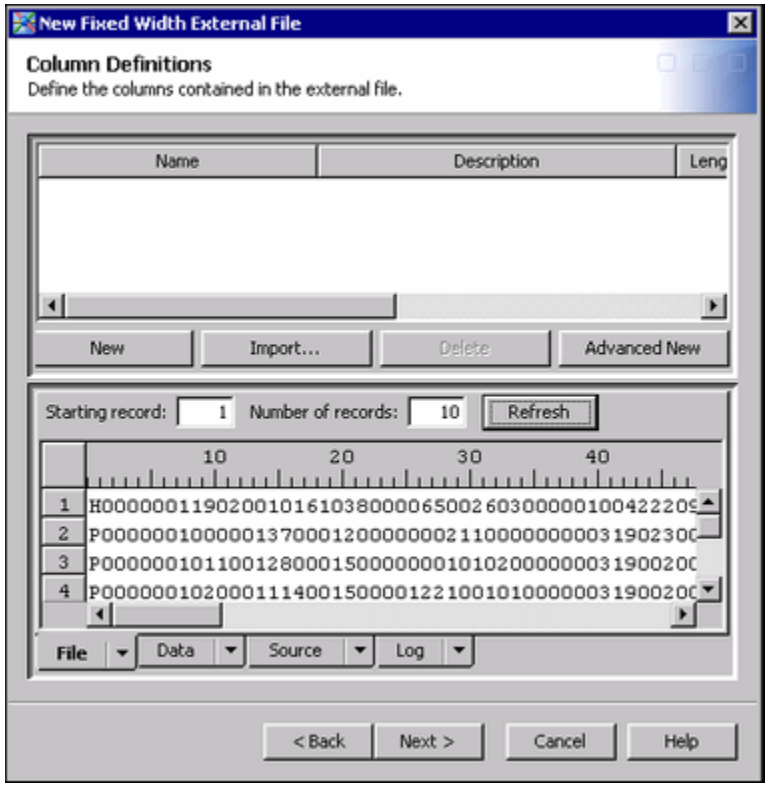
Tasks

Run the Fixed-Width External File Wizard

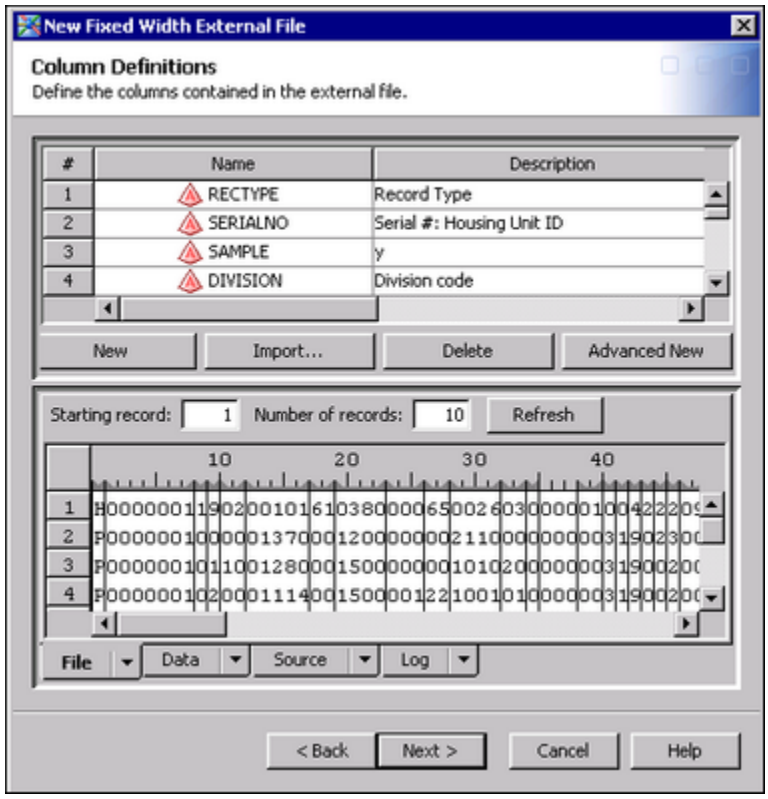
Perform the following steps to use one method to register an external file in the fixed-width external file wizard:

1. Create an external format file that specifies the column metadata for the external data file. For more information about this file, see the "Solution" section above.
2. 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.
3. If you are prompted, enter the user ID and password for the default SAS Application Server that is used to access the external file.
4. Specify the physical path to the external file in the **File name** field. Click **Next** to access the Parameters page.
5. 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.
6. Accept the default for the **Logical record length**, and click the **Next** button to access the Column Definitions page.
7. 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.

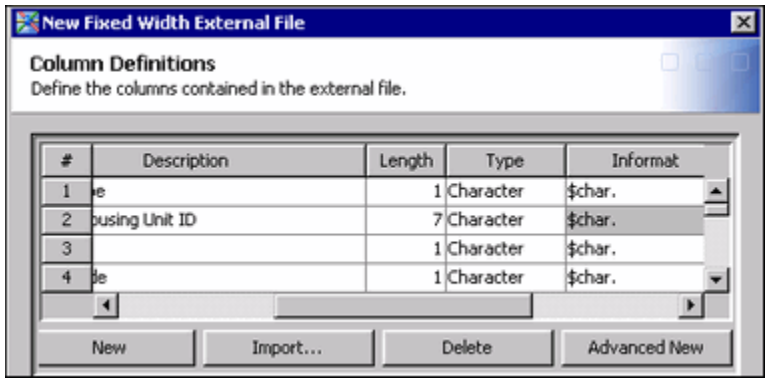
Display 5.15 *Sample Fixed-Width Data on the File Tab*



8. Click **Import**. The Import Column Definitions dialog box is displayed.
9. Select the **Get the column definitions from a format file** radio button.
10. Specify the path to the external format file that you created in Step 1.
11. Click **OK**. The column metadata in the external format file is applied to the current data file, as shown in the next display.



12. If your external format file did not specify SAS informats for all column variables, specify those now. Access the **Informats** column for each data column and select an appropriate SAS informat, as shown in the next display.



13. If you want to see what the data in the external file looks like with the column metadata applied, click **Data** tab, then click **Refresh**. Your data will be formatted with the current metadata. If the data looks correctly formatted, go to the next step. If the data does not look correctly formatted, then use the controls on the Column Definitions tab to correct the metadata.

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

View the External File Metadata

Follow the steps that are described in "View the External File Metadata" on page 126.

View the Data

Follow the steps that are described in "View the Data" on page 127.

Chapter 6

Creating Jobs

About Jobs **146**
Jobs with Generated Source Code 146
Jobs with User-Supplied Source Code 147
Run Jobs 147
Manage Submitted Jobs 147

Creating an Empty Job **147**
Problem 147
Solution 147
Tasks 147

Creating a Process Flow for a Job **148**
Problem 148
Solution 148
Tasks 148

Creating a Job That Contains Jobs **150**
Problem 150
Solution 150
Tasks 150

Working with Default Temporary Output Tables **150**
Problem 150
Solution 150
Tasks 151

Specifying Options for Jobs **155**

Documenting Process Flow Diagrams **155**
Problem 155
Solution 155
Tasks 155

Accessing Local and Remote Data **156**
Data Access Overview 156
Access Data in the Context of a Job 156
Access Data Interactively 157
Use a Data Transfer Transformation 158

Viewing or Updating Job Metadata **158**
Problem 158
Solution 158
Tasks 159

Displaying the SAS Code for a Job **160**
Problem 160

Solution 160
Tasks 160

Common Code Generated for a Job 160
Overview 160
LIBNAME Statements 161
SYSLAST Macro Statements 161
Remote Connection Statements 162
Macro Variables for Status Handling 162
User Credentials in Generated Code 163

About Jobs

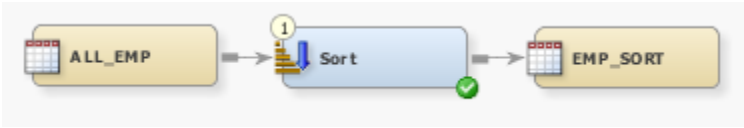
Jobs with Generated Source Code

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 150.

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 148.

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 271.

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 166.
- deploy the job for scheduling. For information, see "Deploying Jobs for Scheduling" on page 227.
- deploy the job as a SAS stored process. For information, see "Deploying Jobs as Stored Processes" on page 238.
- deploy a stored process as a Web service. For information, see "Deploying a Stored Process as a Web Service" on page 251.

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 166.

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 menu bar. 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.

Note: A one-minute screencast (video demonstration) of this task is available at `http://support.sas.com/documentation/onlinedoc/etls/`.

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 147.
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 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 150.)

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

Display 6.2 *Sample Process Flow*



Note: Note the source table is named ALL_EMP and target table is named EMPLOYEES_SORTED. You can also see that icon overlays have been added to the tables to indicate the type of data that they contain. In this case, both tables contain SAS data and feature that icon (). These icon overlays will be shown in all of the process flows that are displayed in future editions of the *SAS Data Integration Studio User's Guide*.

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 "Specifying Options for Jobs" on page 155.

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.

Note: A one-minute screencast (video demonstration) of this task is available at `http://support.sas.com/documentation/onlinedoc/etls/`.

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 in which 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. Be 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 151

- "Use the Default Temporary Output Table As an Input to Another Transformation" on page 151
- "Replace the Default Temporary Output Table with a Permanent Target Table" on page 152
- "Use the Temporary Output Table As an Input to a Table Loader " on page 154

Tasks

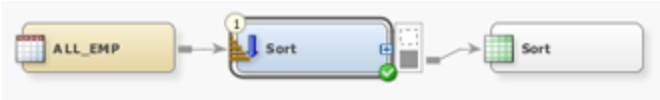
Use the Default Temporary Output Table As the Final Output

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 cannot 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

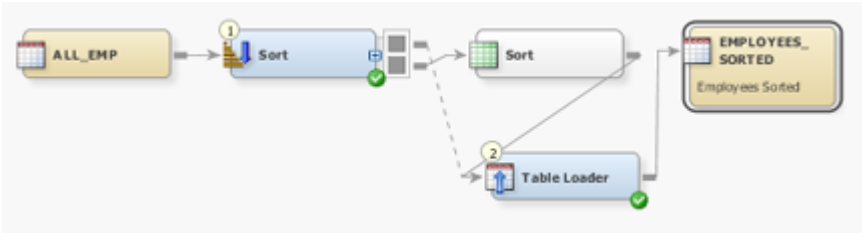
Note that some of these actions are also possible with the SCD Type 2 Loader transformation.

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.

Specifying Options for Jobs

You can enable global options that apply to new jobs by selecting **Tools** ⇨ **Options** from the menu bar. Click the **General** tab and the **Code Generation** tab to set global job options.

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.

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 156
- "Access Data Interactively" on page 157
- "Use a Data Transfer Transformation" on page 158

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 how 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 236. 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 160.

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. Verify 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.

The following display shows the icon that is displayed on the affected transformation when implicit data transfer is used:

Display 6.9 *Implicit Data Transfer Icon*



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.

Note: A one-minute screencast (video demonstration) of this task is available at `http://support.sas.com/documentation/onlinedoc/etls/`.

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 84 or "Viewing or Updating External File Metadata" on page 131.
   - To update the metadata for transformations in the job, open the properties window for the transformation and update the appropriate tabs.
   - To add a transformation to a process flow diagram, select the transformation and drop it in the Job Editor window. For information, see "Adding a Transformation to an Existing Job" on page 180.

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 166.

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 271.)

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 22.

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 161
- "SYSLAST Macro Statements" on page 161
- "Remote Connection Statements" on page 162

- "Macro Variables for Status Handling" on page 162
- "User Credentials in Generated Code" on page 163

The generated code includes the user name and password of the person who created the job. You can set options for the code that SAS Data Integration Studio generates for jobs and transformations. For details, see "Specifying Options for Jobs" on page 155.

LIBNAME Statements

When SAS Data Integration Studio generates code for a job, a library is considered local or remote in relation to the SAS Application Server that executes the job. If the library is stored on one of the machines that is specified in the metadata for the SAS Application Server that executes the job, it is local. Otherwise, it is remote.

SAS Data Integration Studio generates the appropriate LIBNAME statements for local and remote libraries.

The following syntax is generated for a local library:

```
libname libref  <"lib-specification"> <connectionOptions> <libraryOptions>
  <schema=databaseSchema> <user=userID> <password=password>;
```

The following syntax is generated for a remote library:

```
options comamid=connection_type;
%let remote_session_id=host_name <host_port>;
signon remote_session_id <user=userID password=password>;
rsubmit remote_session_id;
libname libref <engine> <"lib-specification"> <connectionOptions>
<libraryOptions> <password=password>;
endrsubmit;
```

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 6.10 *Process Flow with a Custom Error Handling Transformation*



In this example, the Custom Error Handing transformation contains user-written code that handles errors from the Extract transformation, and the error-handling code does not produce output that should be become the input to the target table, ALL_MALE_EMP. Instead, the output from the Extract transformation should become the input to ALL_MALE_EMP. 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 Handing transformation.

Remote Connection Statements

Each transformation within a job can specify its own execution host. When SAS Data Integration Studio generates code for a job, a host is considered local or remote in relation to the SAS Application Server that executes the job. If the host is one of the machines that is specified in the metadata for the SAS Application Server that executes the job, it is local. Otherwise, it is remote.

A remote connection statement is generated if a remote machine has been specified as the execution host for a transformation within a job, as shown in the following sample statement:

```
options comamid=connection_type;
%let remote_session_id=host_name <HOST_PORT>;
SIGNON remote_session_id <USER=userID password=password>;rsubmit
remote_session_id;
  ... SAS code ...
endrsubmit;
```

Macro Variables for Status Handling

When SAS Data Integration Studio generates the code for a job, the code includes a number of macro variables that can be used to monitor the status of jobs. For details, see “About Status Handling for Jobs and Transformations” on page 209.

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 227.

Chapter 7

Managing Jobs

About Managing Jobs **166**

Submitting a Job for Immediate Execution **166**
Problem 166
Solution 167
Tasks 167

Meeting Prerequisites for Collecting Job Statistics **169**

Reviewing a Successful Job **170**
Problem 170
Solution 170
Tasks 170

Diagnosing and Correcting an Unsuccessful Job **175**
Problem 175
Solution 175
Tasks 175

Adding a Transformation to an Existing Job **180**
Problem 180
Solution 180
Tasks 180

Understanding the Job Has Changed Warning **182**

Understanding the Crossed Versions in a Job Warning **183**

Displaying Run-Time Statistics in SAS Job Monitor **184**
Overview 184
Prerequisites for Monitoring Jobs in SAS Job Monitor 184

Displaying Run-Time Statistics in SAS Web Report Studio or the SAS Stored Process Server **184**

Maintaining Column Mappings **185**
Problem 185
Solution 185
Tasks 186

Managing the Scope of Column Changes in Jobs **189**
Problem 189
Solution 189
Tasks 190

Managing Connections in Job Editor Windows **193**
Problem 193

Solution . . . 193
Tasks . . . 193

Viewing the Code for a Transformation . . . 195
Problem . . . 195
Solution . . . 195
Tasks . . . 196

Specifying Options for Transformations . . . 196
Problem . . . 196
Solution . . . 196

Redirecting Temporary Output Tables . . . 196
Problem . . . 196
Solution . . . 196
Tasks . . . 197

Pushing ELT Job Code Down to a Database . . . 198
Problem . . . 198
Solution . . . 198

Using a Web Client to Orchestrate Jobs . . . 199
Problem . . . 199
Solution . . . 199

About Managing Jobs

Once you have a 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 166
- “Meeting Prerequisites for Collecting Job Statistics” on page 169
- “Reviewing a Successful Job” on page 170
- “Diagnosing and Correcting an Unsuccessful Job” on page 175
- “Maintaining Column Mappings” on page 185
- “Managing the Scope of Column Changes in Jobs” on page 189
- “Managing Connections in Job Editor Windows” on page 193
- “Redirecting Temporary Output Tables” on page 196
- “Pushing ELT Job Code Down to a Database” on page 198

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 167
- "Submit Selected Transformations in a Job" on page 167
- "Submit a Segment of a Job" on page 169
- "Submit a Job One Step at a Time" on page 169
- "Submit a Job to a Grid" on page 169

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 27. If you use the pushdown feature, the relational databases in the job are processed on the appropriate database server. For more information, see "Pushing ELT Job Code Down to a Database" on page 198.

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.

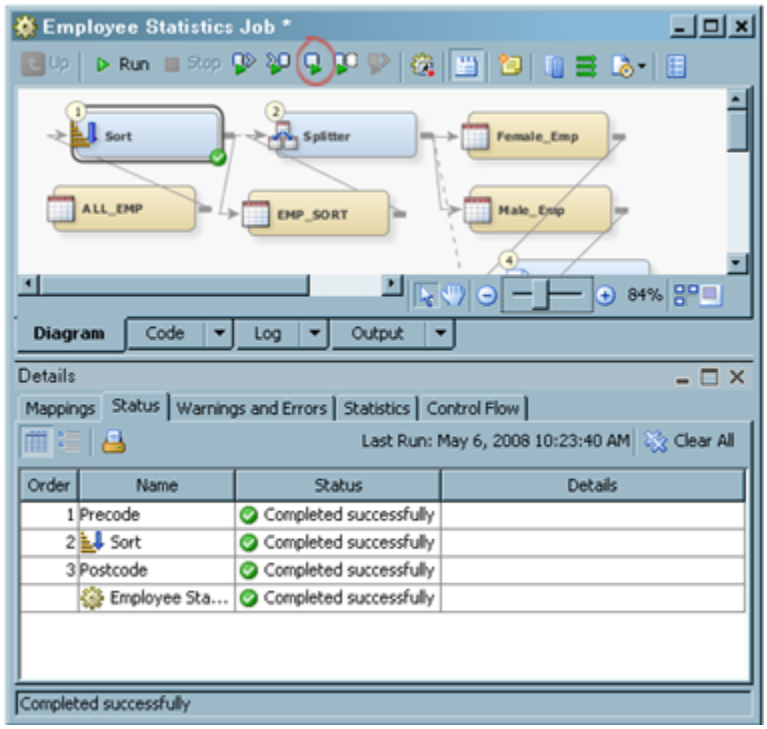
Note: A one-minute screencast (video demonstration) of this task is available at `http://support.sas.com/documentation/onlinedoc/etls/`.

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. Right-click one of the selected transformations, and then select **Run Selected Transformations** from the context menu. 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*

View Data: EMP_SORT (19 rows)

Go to row Go

#	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. You can right-click the transformation and select **Run From Selected Transformation**, or **Run To Selected Transformation**, or **Run Selected Transformation** from the context menu. Alternatively, you can select a transformation and then click **Run From Selected Transformation**, or **Run To Selected Transformation**, or **Run Selected Transformation** from 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

You can track performance statistics for a job that is run interactively. You can use SAS Web Report Studio or the SAS Stored Process Server to display pre-built reports for multiple jobs that were executed on a batch server. If your site has licensed SAS® Environment Manager and SAS® Job Monitor, then you can use a web browser to display run-time statistics for jobs. To collect job statistics, the following prerequisites must be met:

- The logging facility must be enabled on the server that executes the job. ARM statistics are enabled by default for SAS Workspace Servers, but not for SAS Batch servers. If you want to use the pre-built reports, the SAS Data Integration Studio job statistics package must also be installed and configured on the server. For more information, administrators should see the "Administering Logging for SAS Servers" chapter in the *SAS Intelligence Platform System Administration Guide*.
- The collect run-time statistics option must be enabled for the job. This option is enabled by default. If the option has been disabled, and you want to enable it, open the job in the Job Editor window. Then, right-click the canvas and select **Collect Runtime Statistics** and **Collect Table Statistics**. Note that you can also select **Collect Diagnostics**.

Note: You can collect run-time statistics for all new jobs by selecting **Tools** ⇨ **Options** ⇨ **Job Editor**. Then, select the check boxes for **Collect Runtime Statistics** and **Collect Table Statistics**. You can also use the **Maximum numbers of warnings**

and errors field to control the amount of diagnostic information collected for each step.

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:

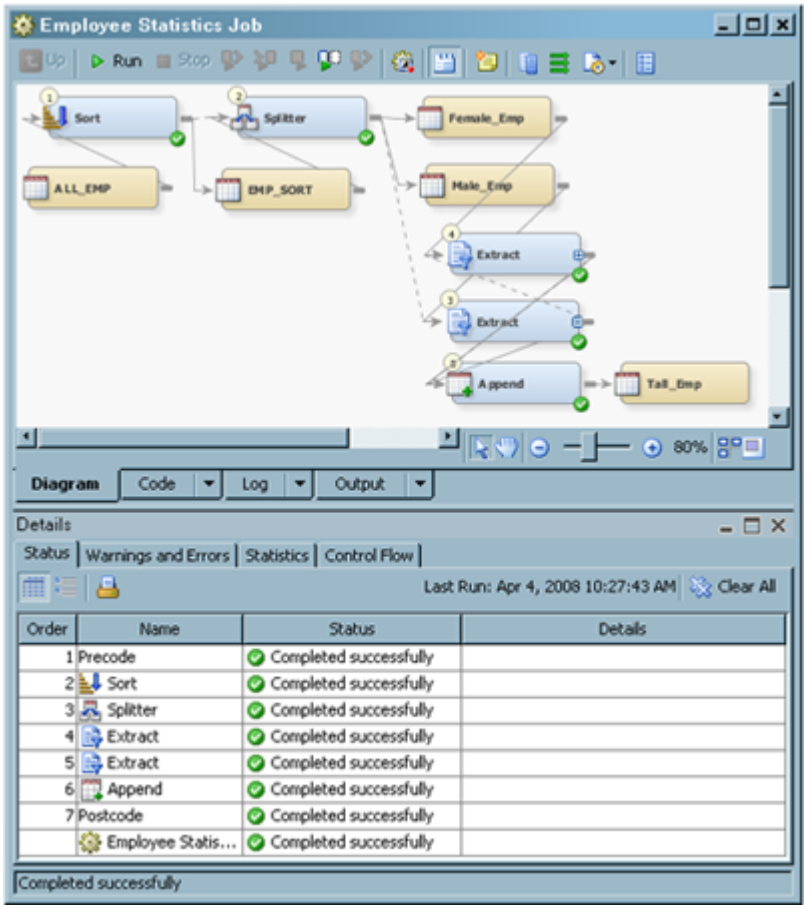
- "Check the Status Tab" on page 170
- "Examine the Statistics Tab" on page 171
- "Examine the Control Flow Tab" on page 174
- "Review the Job Output" on page 174

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 run-time 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:

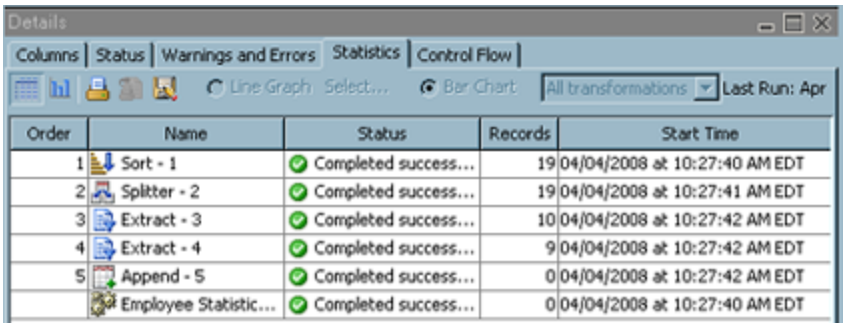
- a green check for a status of complete
- a yellow triangle for a warning
- red X for an error

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*



The statistics table includes the following columns:

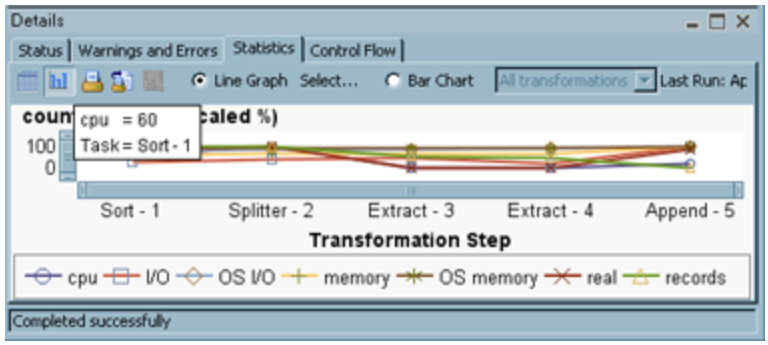
- Order
- Name
- Status
- Records
- Start Time
- End Time
- Duration
- CPU Time
- Current Memory
- System Memory
- Current I/O
- System I/O
- Server
- Threads

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.

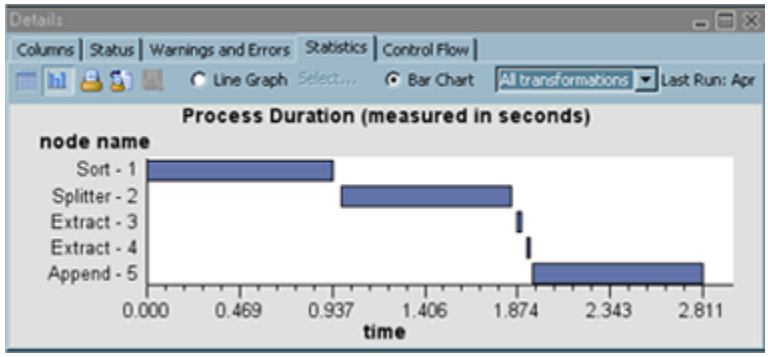
Display 7.5 *Sample Line Graph*



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 positioning the mouse pointer over its bar.

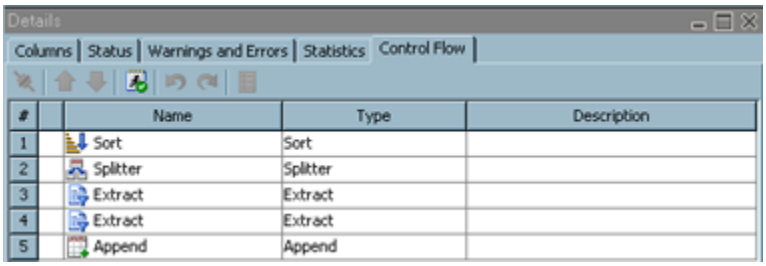
If you do not see the output that you expect on the **Statistics** tab, then you can perform the following troubleshooting tasks:

- When you execute jobs interactively and have run-time 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 run-time 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	

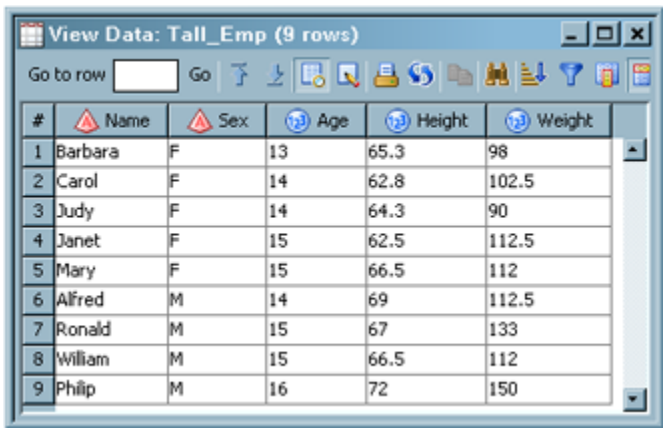
You can click **Validates the control flow** to make sure that the flow is valid. You can also drag a row to a higher or lower position in the table by clicking on the row number and moving the row either up or down. This action moves the transformation included in the row to a different position in 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 does not 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.

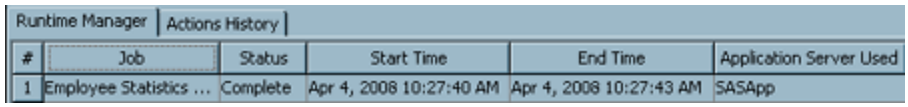
Display 7.8 *Sample View Data Window*



#	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*



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:

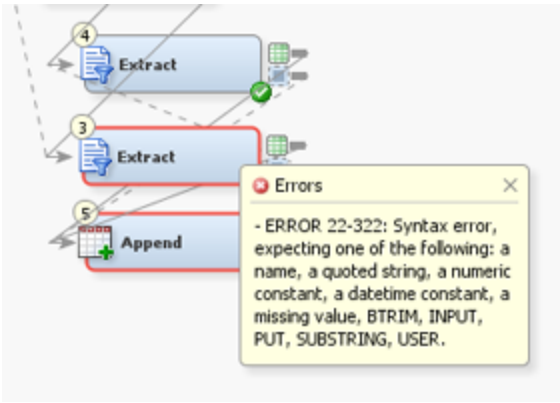
- "Examine the Diagram Tab" on page 175
- "Check the Status Tab" on page 176
- "Read the Warnings and Errors Tab" on page 177
- "Examine the Problem in the Log Tab" on page 178
- "Fix the Problem" on page 178
- "Run the Job and Check the Results" on page 179

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*

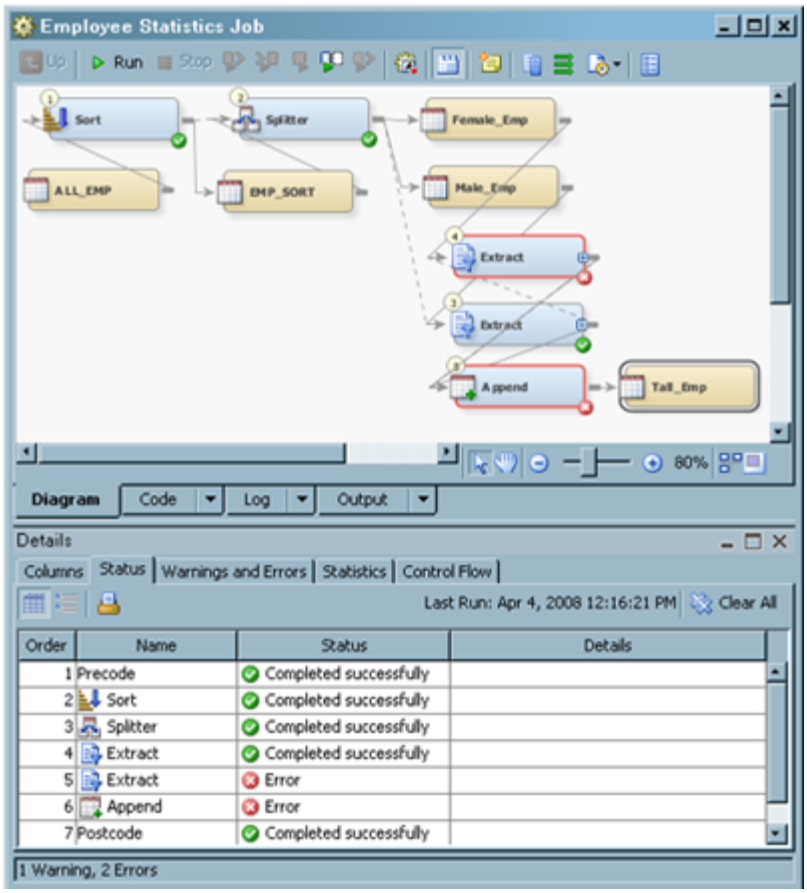


Note: When there are many warning or error messages, only the first few messages are shown in the sticky note due to performance reasons. You can set a limit on the number of messages at the following location: **Tools** ⇨ **Options** ⇨ **Job Editor** ⇨ **Maximum number of warnings and errors to display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

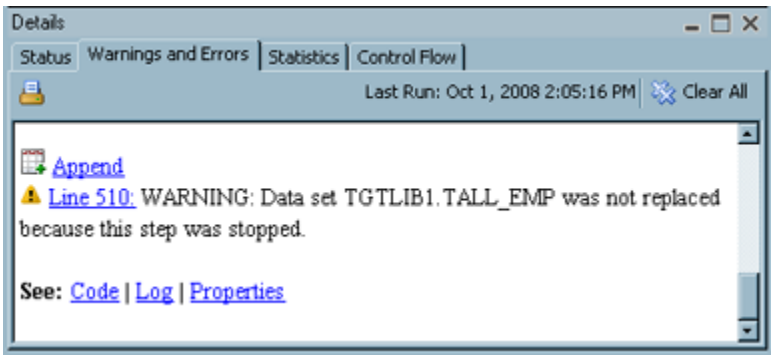
Display 7.11 *Unsuccessful Sample Job*



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

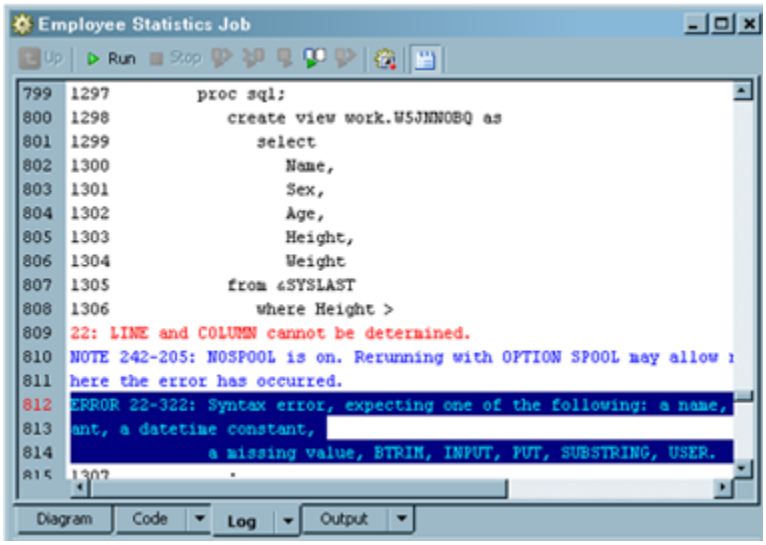
- **Log**: displays the error on the **Log** tab
- **Properties**: displays the properties window for the transformation

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*

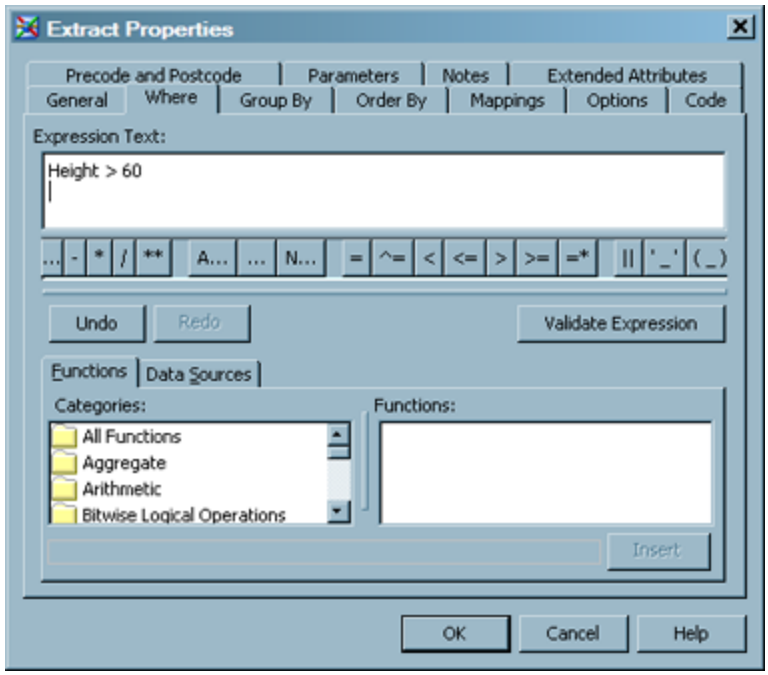


The error corresponds to the code, which is missing a value for **`where Height >`**.

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*

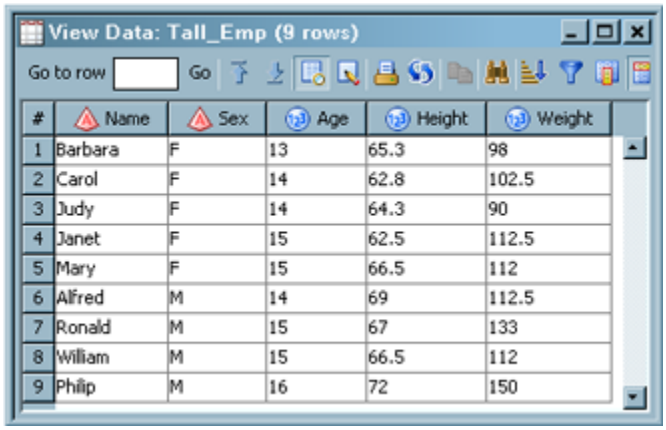


You can fix the sample job by correcting the text in the **Expression Text** field and saving the values in the properties window. After the correction, the expression text reads `Height > 60`.

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*



#	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

Adding a Transformation to an Existing Job

Problem

You want to add a transformation to an existing process flow diagram in a SAS Data Integration Studio job. This transformation adds new functionality to the job. However, you need to add the transformation without disturbing the existing mapping and propagation settings of the current components of the job.

Solution

You can follow a standard process for adding transformations to jobs. This process includes the following tasks:

- "Prepare the Job" on page 180
- "Add the Transformation" on page 180
- "Configure and Run the Updated Job" on page 181

Tasks

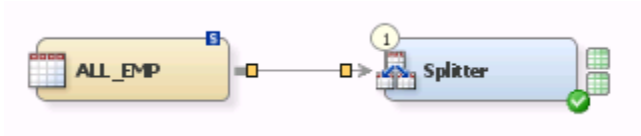
Prepare the Job

Perform the following tasks before you add a transformation to the **Diagram** tab for an existing job:

1. Disable the **Automatically Propagate Job** item. Access this item by clicking the **Settings** button in the toolbar. This action prevents the automatic propagation feature from changing all columns in all transformations. You can restore the propagation settings to the job as a part of the configuration task that is covered at the end of this topic.
2. Delete the arrow between the objects that you need to separate with the added transformation.

   The following display shows the connecting arrow between the source table and the Splitter transformation selected for deletion:

   Display 7.16 *Selected Arrow Connection*



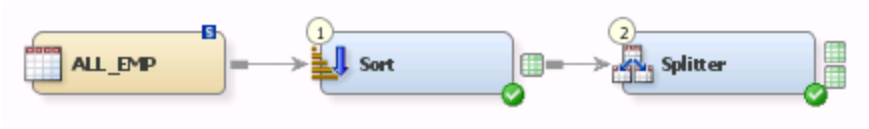
This job uses the Splitter transformation to generate separate lists of male and female employees from a table that contains employee data.

Add the Transformation

Now you can add a transformation to the **Diagram** tab and connect it to the objects that surround it in the job.

The following display shows a job that was updated with a Sort transformation between a source table and a Splitter transformation:

Display 7.17 *Job with Added Transformation*



The Sort transformation is used to sort the data by weight before it is processed by the Splitter transformation.

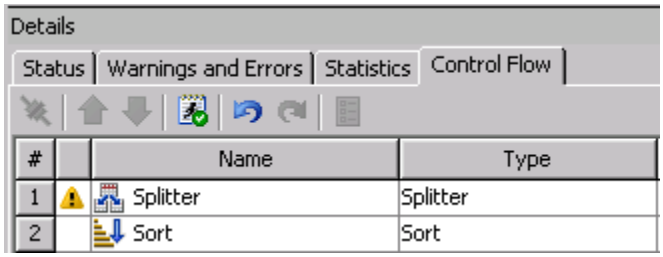
Configure and Run the Updated Job

Perform the following configuration steps before you run the updated job:

1. Click **Control Flow** in the **Details** pane to place the transformations in the proper order.

   The following display shows the uncorrected transformation order:

   Display 7.18 *Uncorrected Transformation Order*

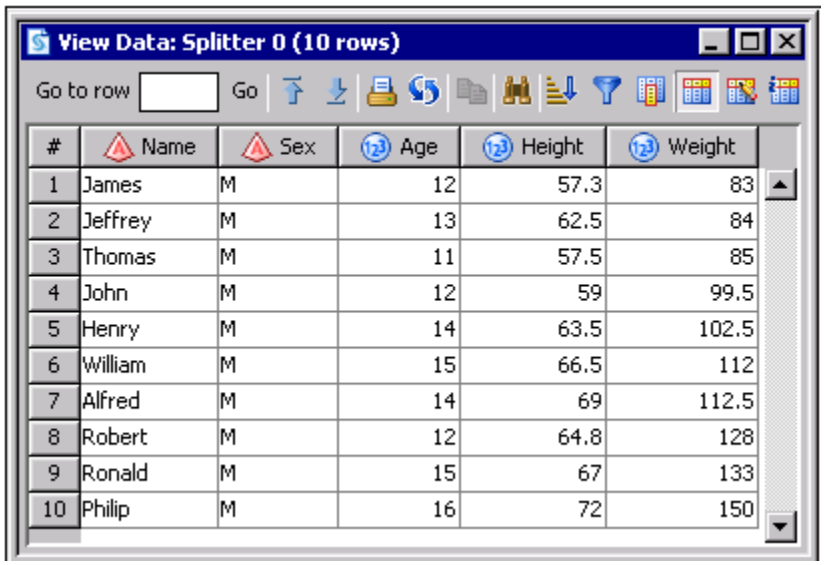


2. Reorder the transformations so that the newly added Sort transformation comes before the Splitter transformation. This action makes the control flow order match the order that is displayed in the job flow on the **Diagram** tab.
3. Open the properties window of the Sort transformation. Then, specify the sort criteria on the **Sort By Columns** tab. For example, you can specify an ascending sort on the **Weight** column.
4. Right-click the Sort transformation and select the **Propagate Columns** item in the pop-up menu. Set the propagation flow to **From Selected Transformation's Sources** and **To Targets**.
5. Open the properties window of the Splitter transformation that was in the original job. Then, check the settings on the **Mapping** tab. If necessary, click **Map all columns** to map between the added Sort transformation and the original Splitter transformation.

Now you can run the job and check the output.

The following display shows the output of the job:

Display 7.19 *Job Output for Males, Sorted by Weight*



#	Name	Sex	Age	Height	Weight
1	James	M	12	57.3	83
2	Jeffrey	M	13	62.5	84
3	Thomas	M	11	57.5	85
4	John	M	12	59	99.5
5	Henry	M	14	63.5	102.5
6	William	M	15	66.5	112
7	Alfred	M	14	69	112.5
8	Robert	M	12	64.8	128
9	Ronald	M	15	67	133
10	Philip	M	16	72	150

As expected, the data from the source table has been sorted by weight and split into separate tables for male and female employees. The Sort transformation has been successfully added to the job flow.

Understanding the Job Has Changed Warning

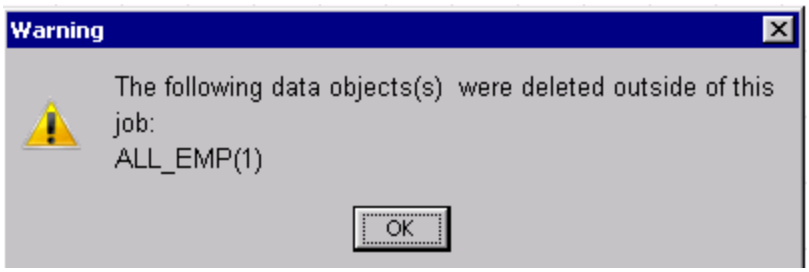
The jobs that you create in SAS Data Integration Studio are frequently run more than once or used by more than one user. Sometime the contents of a job can change, often in subtle ways, between one run and the next. Other times, one user can change a job without notifying the job's other users. For example, you can create and save a job that contains the following objects:

- a source table, such as a list of all the employees in a business
- the Splitter transformation
- two output tables, one for male employees and another for female employees

The source table in the example can then be deleted from the Inventory pane.

When the job is reopened, the warning window in the following display appears:

Display 7.20 *Job Has Changed Message*



This message warns you that the job has changed and briefly describes the change. When you click **OK**, the changed job is shown in the process flow diagram. In this example, the source table has been removed because its metadata was removed from SAS Data Integration Studio when you deleted it. At this point, you can take corrective action, such as adding an updated table as the source for the Splitter transformation.

The warning alerts you to the fact that the job has changed. Without the warning, you can see the changed job. However, you are given no indication of what has changed. The principal scenarios that generate the warning include the following:

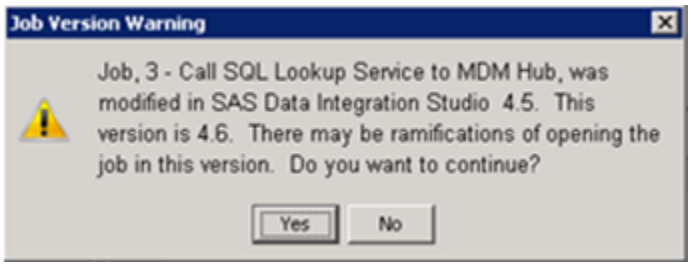
- changes to the generated transformations used in jobs, such as deleting an output table.
- deletion of the metadata for a physical table that is attached to a transformation in a job.
- absence of a generated transformation in a job when the job is loaded. This scenario can occur when you create a generated transform and export a job with it but do not include the generated transformation in the export. If the metadata server that you import the job to does not have the transformation, you see the job has changed warning when you open the job.
- absence of a custom Java transformation in a job when the job is loaded. This scenario can occur when you create a custom Java transformation and export a job with it but do not include the transformation in the export. If the metadata server that you import the job to does not have the transformation, you see the job has changed warning when you open the job.
- items that are out of synchronization with XML such as indexes in a tables list or a transformation in a list. This scenario can occur when a user modifies the job metadata outside of SAS Data Integration Studio clients.

Understanding the Crossed Versions in a Job Warning

The jobs that you create in SAS Data Integration Studio can sometimes be created in one version of SAS Data Integration Studio and run in another version. This cross-version scenario can occur after you have upgraded SAS Data Integration Studio or when you work in a mixed environment that contains more than one version.

If you are using an upgraded version of SAS Data Integration Studio and you open a job created in an earlier version, you see a warning similar to the following:

Display 7.21 *Crossed Versions in a Job Warning*



If all of your users are working in the upgraded version, you can safely ignore the warning. If some of your users need to continue working in the earlier version, you must decide whether to run the job or not.

For example, you might have some users working in the same job in versions 4.5 and 4.6. If the 4.6 user adds a new 4.6 transformation to a job that was created in 4.5 and saves it, the job depends on a transformation that was not available in version 4.5 and is saved as a version 4.6 job. Then the 4.5 user is unable to run the job in version 4.5.

Note that this warning can be disabled by clearing the **Display a warning on older job version** check box on the **Job Editor** tab of the Options window. This window is available from the **Tools** ⇨ **Options** menu. You can disable an error message that is displayed when you open a job that was created in a newer version than the application that you are using by clearing the **Display an error on newer job version** check box. This check box is located on the same tab in the Options window.

Displaying Run-Time Statistics in SAS Job Monitor

Overview

SAS Job Monitor is an optional component of SAS Environment Manager. SAS Environment Manager is a web-based monitoring solution for a SAS environment. SAS Job Monitor reads job logs at specified locations and displays run-time statistics from the logs. If your site has met the prerequisites in the following section, and you can access the SAS Environment Manager, then you can select **Analyze** ⇨ **SAS Job Monitor** to display run-time statistics for SAS Data Integration Studio data jobs.

Prerequisites for Monitoring Jobs in SAS Job Monitor

Your site must license SAS Environment Manager and SAS Job Monitor.

Both Run-time Statistics and Table Statistics must be turned on for those SAS Data Integration Studio jobs that you want to monitor in SAS Job Monitor. For more information about this task, see “Meeting Prerequisites for Collecting Job Statistics” on page 169.

In SAS Job Monitor, an administrator must use the **Tools** ⇨ **Configure Server** option to register the location of the job logs for SAS Data Integration Studio. For more information about this task, see the topic "Configure Batch Jobs for Monitoring” in the Help for SAS Job Monitor.

Displaying Run-Time Statistics in SAS Web Report Studio or the SAS Stored Process Server

You can use SAS Web Report Studio or the SAS Stored Process Server to display pre-built reports for multiple jobs that were executed on a batch server. The information for these reports is captured in server logs on at run time, using SAS Application Resource Monitoring (ARM) capabilities. ARM correlates the job with the hardware that it is being run on, so that memory use and I/O can be captured and tagged to a specific job. Performance records are combined with error messages, warnings, table names, and

other information to allow for complete, drillable reporting on historical job performance and problems.

For example, you can use cube-based reports in SAS Web Report Studio to track outlier executions of a job down to the specific, offending job step. You can use summary and detailed reports to quickly diagnose problems without having to traverse multiple log files by hand. Detailed reports of job-steps support stringent historical auditing of data sources and targets.

See "Meeting Prerequisites for Collecting Job Statistics" on page 169 for information about configuring these reports.

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:

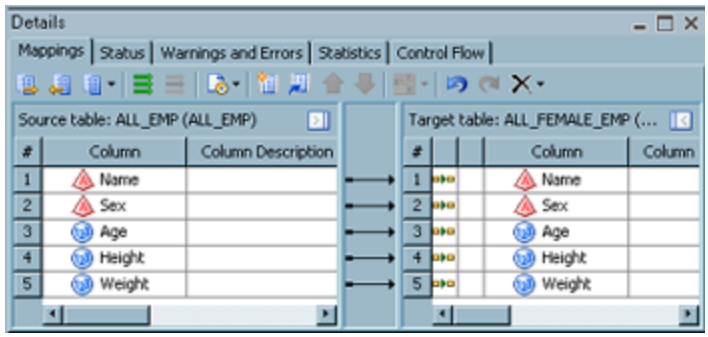
- "Create Automatic Column Mappings" on page 186
- "Create One-to-One Column Mappings" on page 187
- "Create Derived Column Mappings" on page 187
- "Delete Column Mappings" on page 188
- "Use the Options for Mappings" on page 189
- "Customize Mapping Rules" on page 189

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 7.22 *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 **Expression** field	You can create any type of expression by entering the expression directly into an **Expression** field. The expression can be a constant or an expression that uses the values of one or more source columns. For example, you can create a sample expression that writes today's date to a Date column in a target table. Perform the following steps: 1. Double-click in the field in which you want to enter the expression. A cursor displays in the field. (The button disappears.) 2. Enter your expression into the field. For example, to write today's date to every row in a column, you can enter the expression &SYSDATE.

Technique	Description
Create expressions that use no source columns	Some transformations such as Extract, Lookup, and SCD Type 2 Loader provide an Expression column in the target table. You can perform the following steps to enter an expression into this column that does not use source columns: 1. Right-click in an Expression column. Then, click Advanced in the pop-up menu to access the Expression window. 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.
Create expressions that use a single source column	Assume that you want to define the value of a DiscountedPrice column in the target by using the Price source column in an expression. This is possible if the discount is a constant, such as 6 percent. That is, you might want to define an expression as `Price * .94`. You could perform the following steps: 1. Select the Price source column and the DiscountedPrice target column. 2. Right-click either selected variable, and select Expression from the pop-up menu. Then, select Advanced to access the Expression window. 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.
Create expressions that use two or more source columns	You can create a derived mapping that uses two or more source columns. Perform the following steps: 1. Select the source columns and target column to be used in the mapping. For example, you 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. 2. Review the warning that displays because two source columns are mapped to a single target column. 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. 4. Create the expression, which is *Price - (Price * (Discount / 100))* 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.

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.

Customize Mapping Rules

All mappings other than user-defined mappings are created by using rules from a rules file. When you initially start SAS Data Integration Studio, if a mappings rule file does not exist, a file is created in your home folder, such as C:\User\user_name\AppData\SAS\SASDataIntegrationStudio\<version>. The mapping rules are used to determine whether two columns should be mapped automatically when you select a mapping option such as **Map All**. Three rules are provided by default:

- mappings based on Source.Name=Target.Name (case insensitive), Source.Length= Target.Length, Source.Type=Target.Type
- mappings based on an auto conversion Numeric to Character columns when Source.Name=Target.Name (case insensitive)
- mappings based on an auto conversion Character to Numeric columns when Source.Name=Target.Name (case insensitive)

You can customize the rules in the mappings rule file, where you can either add your own rules or edit the default rules. For example, you might define a mapping rule for all column names that begin with the letters WP.

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 190
- "Managing Manual Propagation" on page 191

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	**Automatically propagate columns** in the **Automatic Settings** group box on the **Job Editor** tab in the Options window. (Click **Options** in the **Tools** 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: • From beginning to end • From end to beginning
Job	**Automatically Propagate Job** in the drop-down menu that displays when you click **Settings** in the toolbar on the **Diagram** 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: • From Beginning to End • From End to Beginning
Process flow	**Propagate Columns** in the pop-up menu on the **Diagram** 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: • To Beginning • To End
Transformation	**Include Transformation in Propagation** in the drop-down menu that displays when you click **Settings** in the toolbar on the **Mappings** tab. This option controls automatic propagation of column changes in the selected transformation.	Not applicable
Transformation	**Include Selected Columns in Propagation** in the drop-down menu that displays when you click **Settings** in the toolbar on the **Mappings** 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	**Propagate Job** in the toolbar in the **Diagram** 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	**Propagate Columns** in the pop-up menu in the **Diagram** tab in the Job Editor window	Propagates column changes in the process flow in a specified direction.	Use the following directions: • To Beginning • To End
Transformation	**Propagate from sources to targets** in the toolbar in the **Mappings** tab	Propagates column changes in the process flow from source tables to target tables.	From source tables to target tables.
Transformation	**Propagate from targets to sources** in the toolbar in the **Mappings** tab	Propagates column changes in the process flow from target tables to source tables.	From target tables to source tables.
Transformation	**Propagate** in pop-up menus in the **Source table** field and the **Target table** field	Specifies a path and a direction for propagating column changes. See the table that follows for details.	
Transformation	**Propagate columns** in the toolbar on the **Mappings** 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 **Propagate** option in pop-up menus in the **Source table** field and the **Target table** field	
To Targets	• From Sources • From Beginning • From End
From Targets	• To Sources • To Beginning • To End
Selected Target Columns	• To Sources • To Beginning • To End
Update Selected Target Columns	• To Sources • To Beginning • To End
For the **Propagate columns** in the toolbar on the **Mappings** tab	
To Targets	• From Sources • From Beginning • From End
To Sources	• From Targets • From Beginning • From End
From Targets	• To Sources • To Beginning • To End
From Sources	• To Targets • To Beginning • To End
Selected Target Columns	• To Sources • To Beginning • To End

Path	Direction
Selected Sources Columns	• To Targets • To Beginning • To End
Update Selected Target Columns	• To Sources • To Beginning • To End
Update Selected Sources Columns	• To Targets • To Beginning • To End

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:

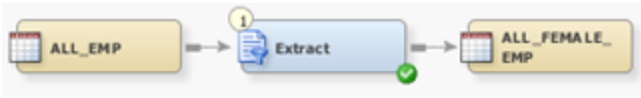
- "Review the Connections for the Object" on page 193
- "Change the Inputs and Outputs for the Object" on page 194

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.

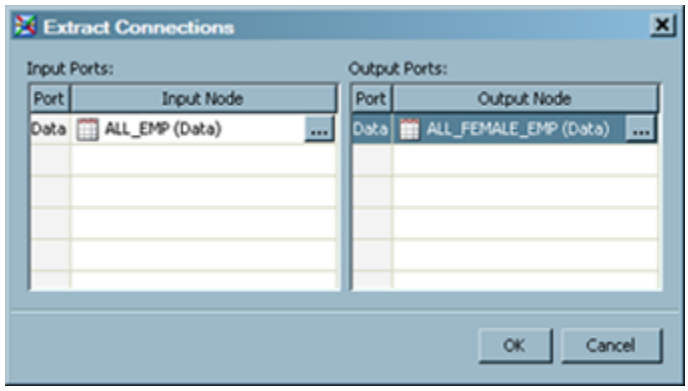
Display 7.23 *Initial Process Flow*



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 7.24 *Connections Window*



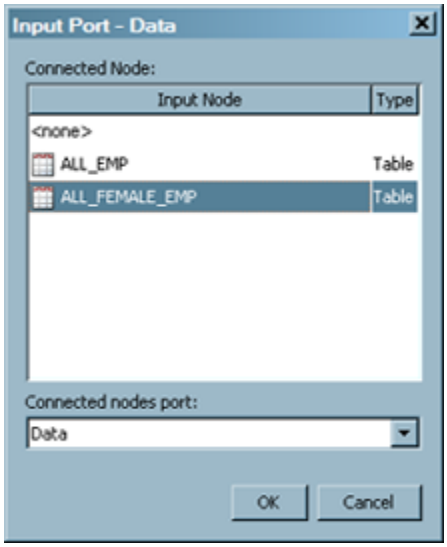
2. Review the inputs and outputs for the object. Note that the ALL_EMP table is listed as an input node in the **Input Ports** field. In addition, the ALL_FEMALE_EMP is listed as an output node in the **Output Ports** field. Both fields also include a **Selector** button. This button is displayed only 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.

1. 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.

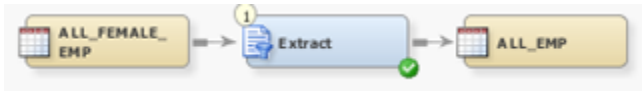
Display 7.25 *Input Selector Window*



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.

2. Click **OK** to save the change to the input node for the object.
3. Use selector windows to change any other objects that you need to update. Then, save the changes.
4. 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 7.26 *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:

1. Open an existing SAS Data Integration Studio job.
2. 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.
3. Open the **Code** tab, and review the code for the transformation.
4. 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.
5. Close the View Step Code window and the properties window for the transformation.

Specifying Options for Transformations

Problem

You want to specify options for a transformation, or you want to specify table options for the transformation inputs or outputs.

Solution

Use the **Options** tab in the properties window for a transformation to specify various options that can affect the behavior of the transformation. For example, you can collect diagnostic messages for some transformations. The options available will vary according to the transformation.

Use the **Table Options** tab to specify table options for the inputs and outputs of most transformations. The options available will vary according to the data format of the tables (SAS or DBMS) and whether the table is an input or an output.

Redirecting Temporary Output Tables

Problem

You want to redirect the output of your temporary tables to an alternative location.

Solution

Transformations in a job typically create temporary work tables as they execute. The default location for these temporary tables is the SAS WORK library. You can now

easily redirect these temporary tables to an alternative location, including a DBMS. Redirecting this output provides the following benefits:

- improved performance. For example, processing data in a DBMS requires no data transfer. For more information, see "Reviewing Temporary Output Tables" on page 306.
- support for the restarting jobs from checkpoints feature. For more information, see "Specify Libraries for a Checkpoint" on page 203.
- support for the pushdown of work to a third-party database. For more information, see "Pushing ELT Job Code Down to a Database" on page 198.

You can redirect the output of your temporary tables within the following scopes: all new jobs, a single job, and a single transformation. Perform the following tasks:

- "Redirect Temporary Output Tables in All New Jobs" on page 197
- "Redirect Temporary Output Tables in a Single Job" on page 197
- "Redirect Temporary Output Tables Attached to a Single Transformation" on page 197

Tasks

Redirect Temporary Output Tables in All New Jobs

Perform the following steps to redirect the output of your temporary tables to an alternative location for all new jobs.

1. Open the **Code Generation** tab in the Options window. You can access the Options window at **Tools** ⇨ **Options** in the menu bar.
2. Click **Browse for a library**, which is adjacent to the **Alternative library for temporary tables** field, to select an existing library.
3. Click **OK** to close the **Options** window.

Redirect Temporary Output Tables in a Single Job

Perform the following steps to redirect the output of your temporary tables to an alternative location for a single job.

1. Open the **Options** tab in the properties window for the job.
2. Click **Browse** adjacent to the **Alternate library for temporary tables** fields to select a library from the **Folders** tab.

   Note: You can set the **Clean up alternate temporary library after successful run** option to **Yes** to delete temporary tables after the deployed job runs successfully. If you set this option to **No**, you should periodically delete the temporary tables manually to conserve disk space.
3. Click **OK** to close the properties window.

Redirect Temporary Output Tables Attached to a Single Transformation

Perform the following steps to redirect the output of your temporary tables to an alternative location for a temporary output table attached to a transformation.

1. Click the **Physical Storage** tab in the properties window for the temporary output table.

2. Click **Redirect to a registered library** in the drop-down menu in the **Location** field.
3. Click **Select a library** in the **Library** field and select the appropriate library. Click **OK** to close the Select a library window. You can also click **New** to access the New Library wizard and register a new library.
4. Click **OK** to close the properties window.

Pushing ELT Job Code Down to a Database

Problem

You want to submit some of the code in a SAS Data Integration Studio job to a relational database server. You need to extract the data, load it in a native database, and transform it in that database. Then, you can run transformations on the data in relational database tables directly in the relational database.

Solution

You can use the pushdown feature to specify that the relational database code in the job is processed in the relational database server. This feature enables you to verify that your job contains tables and transformations that support pushdown. It also enables you to validate your job for pushdown and confirm that pushdown processing occurs when you submit the job.

When both the inputs and outputs of the Extract, SQL Join, Teradata Table Loader, and Table Loader transformations are stored in the same relational database, the code for these transformations can be pushed down to a database server for execution. This option increases performance by shifting data transformation to the most appropriate processing resource.

Note: The use of the Table Loader transformation in a pushdown job requires the following settings:

- **Load style**: select either **Append to Existing** or **Replace**
- **New Rows**: select **Insert (SQL)**

Database processing is validated whenever a job is run. If a job can be run on the database server, it is by default. You can also perform a check to determine whether it is possible to use database processing for a job. This check is strictly diagnostic. It validates only the possibility of database processing without running the actual job. To run this check, click **Check Database Processing** in the job toolbar.

Database processing can fail for a variety of reasons. The following causes are common:

- using SAS data set options
- requesting views instead of tables
- disabling the **Use the optimized pass-through facility for SQL statements** option on a transformation

The following paper explains how to stage data inside the database and direct SAS to do its data integration work inside the database: “SAS® Data Integration Studio: Tips and Techniques for Implementing ELT.” You can access this paper at“SAS® Data

Integration Studio: Tips and Techniques for Implementing ELT.”. If you need user-defined functions, see “User-Defined Functions” on page 622.

Using a Web Client to Orchestrate Jobs

Problem

You want to use a web client to integrate SAS Data Integration Studio jobs into a larger process flow.

Solution

If your site has licensed SAS Visual Process Orchestration, then you can use a web browser to integrate SAS Data Integration Studio jobs into a larger process flow. SAS Visual Process Orchestration enables you to build orchestration jobs, which are process jobs that run other jobs. An orchestration job can integrate executable files from various systems into a single process flow. A single orchestration job can run one or more executable files, such as SAS Data Integration Studio jobs, DataFlux Data Management Studio jobs, SAS code files, third-party programs, scripts, and web services. For more information about SAS Visual Process Orchestration, see **http://support.sas.com/documentation/onlinedoc/po/**.

Chapter 8

Restarting Jobs From Checkpoints

About Restarting Jobs **201**

Prerequisites for Restarting Jobs **202**

Adding Checkpoints to a Job **202**
Problem 202
Solution 203
Tasks 203

Restarting a Job **204**
Problem 204
Solution 204
Tasks 204

About Restarting Jobs

The restart from a checkpoint feature provides a means for a failed job to be restarted at the last successful checkpoint taken before the failed step. By default, jobs are designed without implicit checkpoints. Instead, users must explicitly specify checkpoints at the appropriate steps. Checkpoints consist of code that is inserted before a selected transformation's step.

When a job is rerun after a failure, the last saved checkpoint becomes the restart point for the run. The restart feature enables you to restart a job at the beginning of a step (transformation) when a job previously failed at that step or a subsequent step.

The code for the steps preceding the checkpoint is skipped, and the state is restored from the save-state information preserved by the checkpoint code. Then, processing can pick up from the specified transformation. On a rerun, you can run from either the last saved checkpoint or the beginning of the job. You cannot rerun the job from any other checkpoint.

Note: Only the last successful checkpoint is saved when a job with multiple checkpoints is run. The saved-state information of the last successful checkpoint overwrites the information from earlier checkpoints.

The state can be restored because the following entities are restored to their values from the previous run:

- macro variable values that are saved at the checkpoint using **`set sashelp.vmacro(where=(scope ne 'AUTOMATIC'))`**. However, some macro variables are filtered out:

```
'CPRID','JOB_RC','TRANS_RC','ETLS_STARTTIME','ETLS_ENDTIME',
'SQLRC','ETLS_RESETRESTART','ETLS_STEPSTARTTIME','ETLSCPR_PENDINGID',
'ETLSCPR_RUNNINGID','ETLS_RUNNINGINTERACTIVE','_ARMEXEC','_PERFINIT',
'_ARMTXID','_PERFNEST','_ARMSHDL','_ARMAPID'
```

.

- library assignments

The following entities are not restored:

- SAS Global Options. Restoring global options might undo a setting set by an administrator in a configuration file for a rerun. If you add code in a job to set global options, the code should be put in a transformation marked to always run. To set a selected transformation to always run, click **Yes** in the **Run this transformation always when restarting** field on the **Options** tab of the properties window of the transformation.
- Macros in catalog WORK.SASMACR. Although saving and restoring these macros might be beneficial, there are write-permission problems with restoring macros in this catalog. Therefore, the restart from checkpoint feature will no longer be saving and attempting to restore SASMACR catalog entries. If you have a transformation in a job that declares a macro used in subsequent steps, you must flag the transformation as **Run always**.
- Connections to remote machines. If the step that contains the connection code for a job is skipped, the steps that depend on the connection fail.

The restart from a checkpoint feature is covered in the following topics:

- "Prerequisites for Restarting Jobs" on page 202
- "Adding Checkpoints to a Job" on page 202
- "Restarting a Job" on page 204

Prerequisites for Restarting Jobs

You must satisfy the following prerequisites to restart jobs from checkpoints in SAS Data Integration Studio jobs:

- Add the checkpoints to appropriate transformations in a job. For more information, see "Add a Checkpoint to a Transformation" on page 203
- Specify a save-state library for the job. For more information, see "Specify Libraries for a Checkpoint" on page 203.

Adding Checkpoints to a Job

Problem

You want to mark selected transformations as restart points in a job. If the job fails to complete successfully, you want to be able to rerun the job from the point where it failed.

Solution

You can set checkpoints for appropriate transformations in the job. If the job fails to run successfully, you will be able to restart it from either the last successful checkpoint or from the beginning of the job. You also must specify a library for the saved-state information for the job. Finally, you can specify an optional library for the work tables in the job. Perform the following tasks:

- "Add a Checkpoint to a Transformation" on page 203
- "Specify Libraries for a Checkpoint" on page 203

Tasks

Add a Checkpoint to a Transformation

Perform the following steps to add a checkpoint to a transformation:

1. Open a SAS Data Integration Studio job.
2. Right-click a transformation and click **Assign as Restart-Point** in the pop-up menu. The Restart-point Setup window is displayed.

Specify Libraries for a Checkpoint

Perform the following steps to specify one or both libraries for the checkpoint:

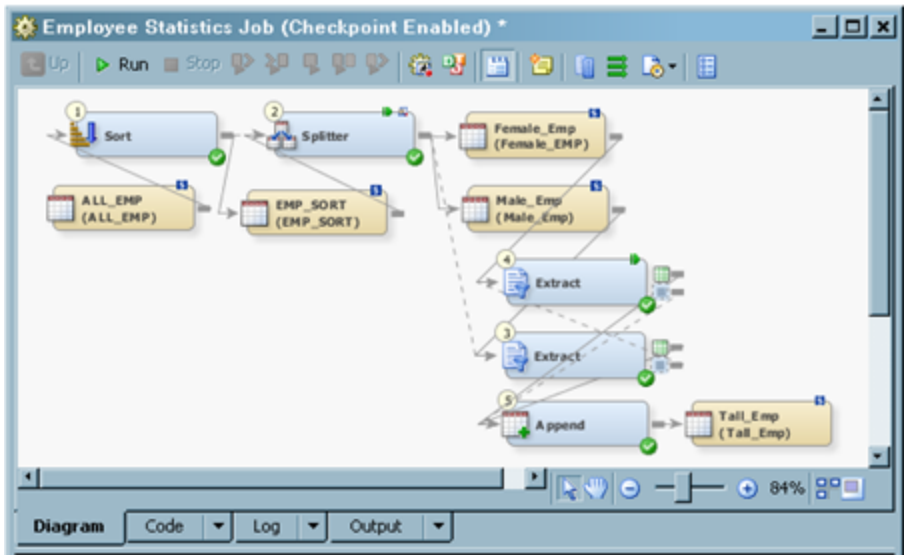
1. Specify a saved-state library in the **Restart-point state library** field. You can click **Select a library** to select an existing library or click **New** to register a new library. After you have specified a library, you can click **Properties** to access its properties window. Note that the saved-state library must be local to the server executing the job.
2. You can also specify an optional library to save the temporary tables in the job in **Alternative library for temporary tables (optional)** field. You need this library only when your job requires the SAS work tables that were created in previous steps when you restart it. You can either select an existing library or register a new library. For details about redirecting output, see "Redirecting Temporary Output Tables" on page 196.

   Note: As implemented, the save-state feature does not save the SAS WORK library during a checkpoint. You must determine whether any particular checkpoint-flagged step (or subsequent step) requires the SAS WORK tables created in preceding steps. If so, you must change the physical location of those temporary tables as part of the job design. If the temporary tables are left in SAS WORK, a rerun with a restart pending can result with "Table-not-found" errors. You can change the location on the **Physical Storage** tab of the properties window for the temporary table. You can also use the **Alternative library for temporary tables (optional)** field to specify the default temp library for the job to be something other than SAS WORK.
3. Click OK to close the Restart-point Setup window.

   Note: You can specify one or both of the libraries for the checkpoints in all new jobs. Use the **Restart-point state library** and **Alternate library for temporary tables** on the **Code Generation** tab of the Options window. You an access the Options window at **Tools** ⇨ **Options** in the menu bar.

4. Right-click any additional transformations that require checkpoints and click **Assign as Restart-Point** in the pop-up menu. The following display shows the **Diagram** tab for a sample job with checkpoints.

Display 8.1 *Sample Job With Checkpoints*



Note the checkpoint icon overlays ( ) in the upper-right corners of the Splitter transformation and the first Extract transformation.

Restarting a Job

Problem

You want to restart a SAS Data Integration job after it has failed to complete successfully.

Solution

You restart the job from the first checkpoint that follows the error in the job. For information about adding checkpoints to jobs, see "Adding Checkpoints to a Job" on page 202. Perform the following tasks:

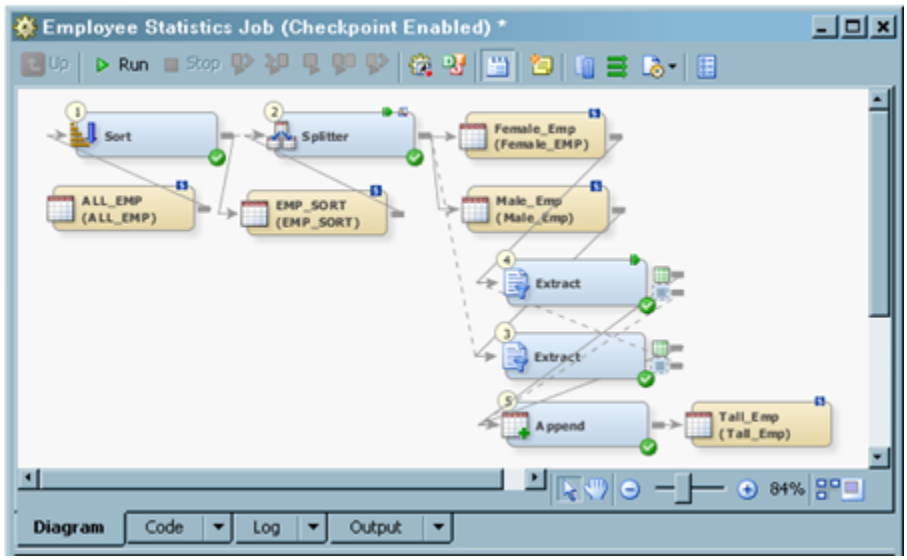
- "Run a Job That Includes Checkpoints" on page 204
- "Restart the Job From a Checkpoint" on page 206

Tasks

Run a Job That Includes Checkpoints

Perform the following steps to run a job that includes one or more checkpoints:

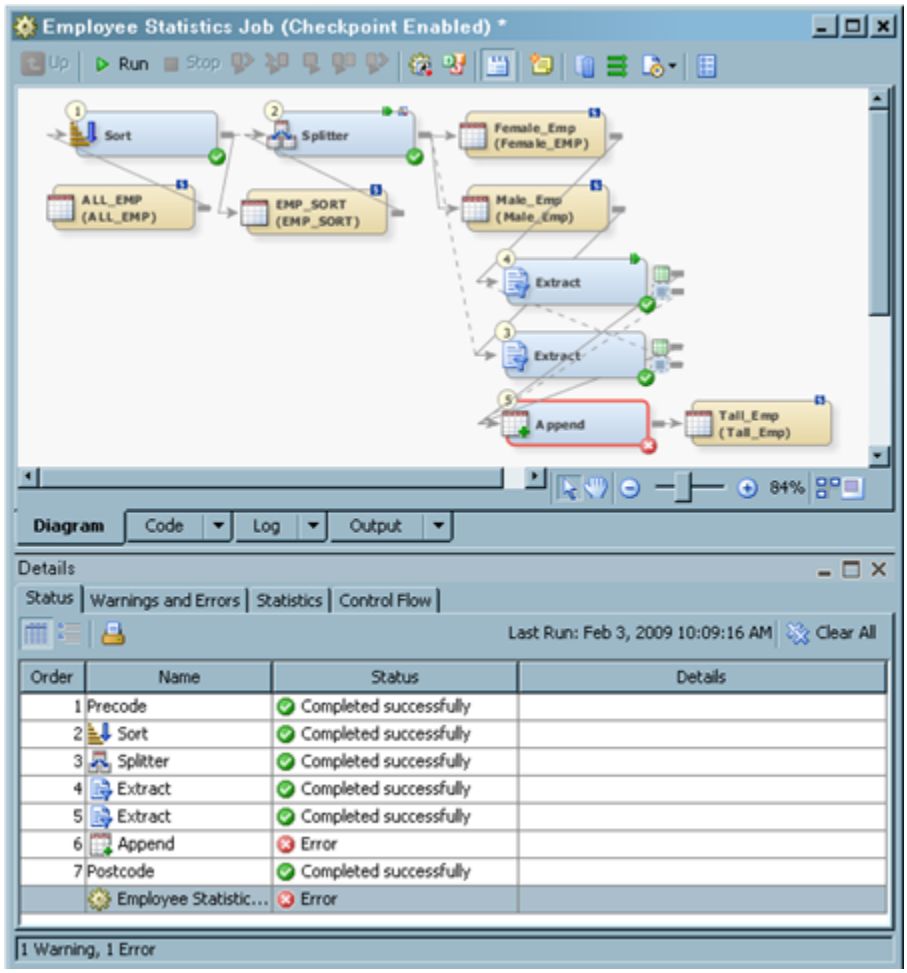
1. Open a job that the contains checkpoints. For example, the sample job shown in the following display contains two checkpoints that are attached to selected transformations.



Note (1) the checkpoint icons in the upper-right corners of the Splitter transformation and the first Extract transformation and (2) the message Checkpoint Enabled in the title bar of the Job Editor window. The first Extract transformation contains an error.

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 the results of the run.

Display 8.2 *Sample Job After First Run*



The error causes the job to fail at the Append transformation.

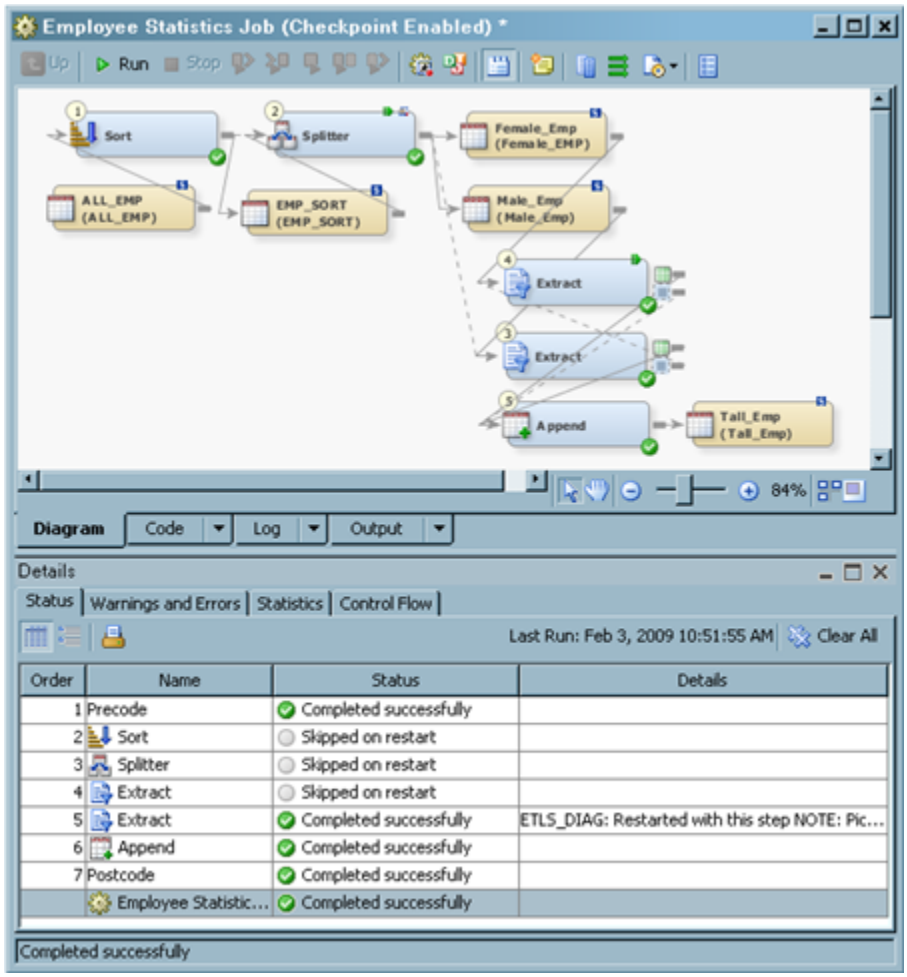
3. Correct the error that caused the job to fail. In this case, you can specify a minimum height for female employees (`Height > 60`) in the properties window for the first Extract transformation. Now you can use the checkpoints in the job to enable you to restart it at the appropriate place.

Restart the Job From a Checkpoint

Perform the following steps to restart the job at the checkpoint attached to the first Extract transformation:

1. Right-click on an empty area of the job, and click **Run** in the pop-up menu. The **Run Options** window is displayed.
2. Select the check box to restart the job from the appropriate checkpoint. Note that only the last successful checkpoint is saved when a job with multiple checkpoints is run. The saved-state information of the last successful checkpoint overwrites the information from earlier checkpoints. The check box in the sample job is named **Restart from checkpoint taken immediately before Extract**. This selection ensures that the job restarts with the second Extract transformation.
3. Click OK to restart the job. The following display shows the restarted job.

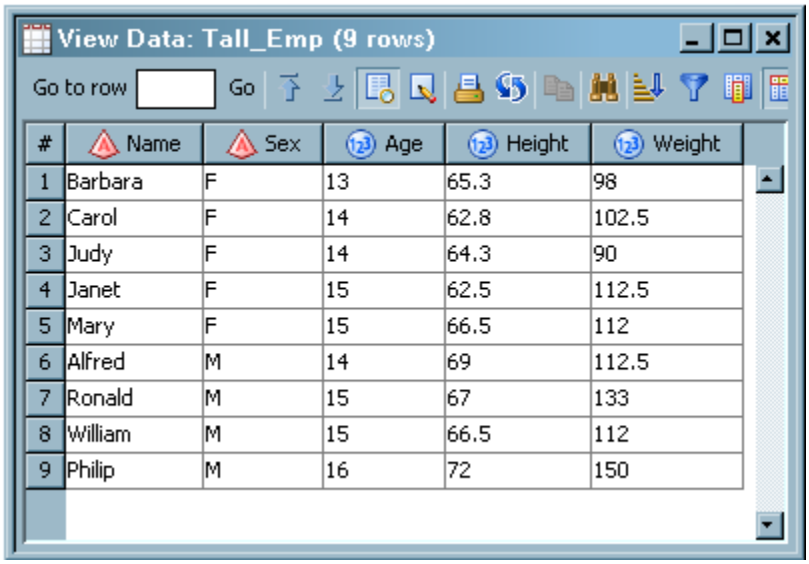
Display 8.3 *Sample Restarted Job*



Note that the steps for the Sort and Splitter transformations and the first Extract transformation are marked as **Skipped on last restart** on the **Status** tab. In addition, a note about the restart is added to the Details column for the step for the second Extract transformation.

4. Right-click the output table for the job. Then, click **Open** in the pop-up menu to access the View Data window for the table. The following display shows the output for the sample job.

Display 8.4 *Output for a Sample Job*



#	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

Note that only female employees with a height greater than 60 are included in the output. Thus, you can generate the desired output even when you restart a corrected job and skip some of its steps.

Chapter 9

Managing the Status of Jobs and Transformations

About Status Handling for Jobs and Transformations **209**

Default Conditions, Actions, and Conditional Action Sets **210**
Overview 210
Default Conditions 210
Default Actions 211
Conditional Action Sets 213

Prerequisites for Actions **215**

Perform Actions Based on the Status of a Job **216**
Problem 216
Solution 216
Tasks 216

Perform Actions Based on the Status of a Transformation **217**
Problem 217
Solution 217
Tasks 217

Macro Variables for Status Handling **219**
Overview 219
Example: Macro Variables for Status Handling in Generated Code 219
Macro Variables for Status Handling in User-Written Code 224

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**, and 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 9.1 *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.

Condition	Description
Table Does Not Exist	Table does not exist in physical storage.
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 9.2 *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 **Precode and Postcode** tab in a job or transformation, or a macro in a SAS Autocall library.	In the **Custom Code** field, enter a call to the user-defined code. One example is the following call to a macro in a SAS Autocall library: `%sendcustom;`

Action	Description	Input
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.
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 **Message** 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.

Action	Description	Input
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 **Lookups** tab of the Lookup transformation property window.	SAS expression.
Set Target Column Value to Missing	Sets the target column value to missing; accessible from the **Lookups** 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 9.3 *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

Conditional Action Sets	Description
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
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 9.4 *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 *SAS Intelligence Platform: Application Server Administration Guide*.
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	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 **Precode and Postcode** 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. 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 *SAS Intelligence Platform: Desktop Application Administration Guide*.
Any action that requires a libref	The libref must be assigned before the job or transformation executes. To assign a library within SAS Data Integration Studio, you can select the **Precode and Postcode** tab in the properties window for the job or transformation and then specify a SAS LIBNAME statement in the **Precode** area. 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 *SAS Intelligence Platform: Data Administration Guide*.

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 216
- "Run the Job and Verify the Status Handling Output" on page 216

Some actions require server setup, as described in "Prerequisites for Actions" on page 215.

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 217
- "Add a Return Code Check Transformation After the Transformation You Want to Monitor" on page 218

Then verify the job as described in "Run the Job and Verify the Status Handling Output" on page 219. Some actions require server setup, as described in "Prerequisites for Actions" on page 215.

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.

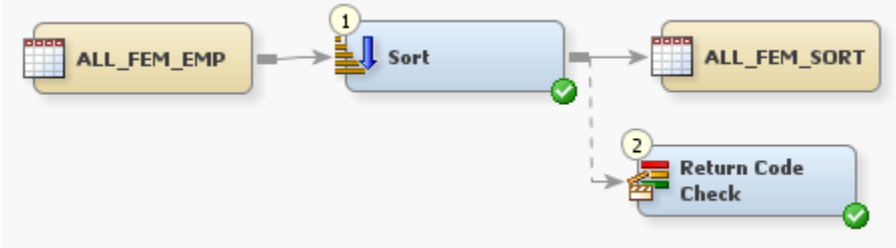
You are now ready to run the job and verify the status handling output.

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 9.1 *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 217. 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 219
- "Macro Variables for Status Handling in User-Written Code" on page 224

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 "aedecod"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 "bodysys"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 10

Deploying Jobs

About Deploying Jobs **226**

About Deploying Jobs for Scheduling **227**

Prerequisites for Deploying a Job for Scheduling **227**

Deploying Jobs for Scheduling **227**
Problem 227
Solution 227
Tasks 228

Using a Command Line to Deploy Jobs **229**
Problem 229
Solution 230
Tasks 230

Redeploying Jobs for Scheduling **233**
Problem 233
Solution 233
Tasks 233

Using Scheduling to Handle Complex Process Flows **235**
Problem 235
Solution 235
Tasks 235

Using Deploy for Scheduling to Execute Jobs on a Remote Host **236**
Problem 236
Solution 236
Tasks 236

About Deploying Jobs as Stored Processes **237**

Prerequisites for Deploying a Job as a Stored Process **237**
For Administrators 237
For Users 238

Deploying Jobs as Stored Processes **238**
Problem 238
Solution 238
Tasks 238

Redeploying Jobs to Stored Processes **241**
Problem 241
Solution 241
Tasks 241

Viewing or Updating Stored Process Metadata **242**
Problem 242
Solution 242
Tasks 242

About Deploying Jobs as Web Services **243**

Prerequisites for Web Service Jobs **243**
For Administrators 243
For Users 243

Requirements for Web Service Jobs **244**

Creating a Web Service Job **244**
Problem 244
Solution 245
Tasks 245

Deploying a Web Service Job as a Stored Process **249**
Problem 249
Solution 249
Tasks 249

Deploying a Stored Process as a Web Service **251**
Problem 251
Solution 251
Tasks 251

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 227.
- Deploy a job as a SAS stored process; see "About Deploying Jobs as Stored Processes" on page 237.
- 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 243.

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 235. 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 150.
- Execute a job on a remote host; see "Using Deploy for Scheduling to Execute Jobs on a Remote Host" on page 236. 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 227
- "Redeploying Jobs for Scheduling" on page 233
- "Using a Command Line to Deploy Jobs" on page 229
- "Using Scheduling to Handle Complex Process Flows" on page 235

See also "Prerequisites for Deploying a Job for Scheduling" on page 227.

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 228.
- Use other software to schedule the job for execution. For more information, see *Scheduling in SAS*. For information about scheduling prerequisites, see "Prerequisites for Deploying a Job for Scheduling" on page 227.

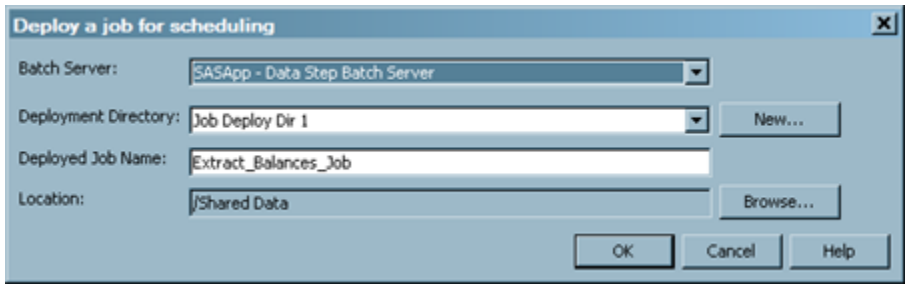
Tasks

Deploy a Job for Scheduling

Perform the following steps to deploy a job for scheduling:

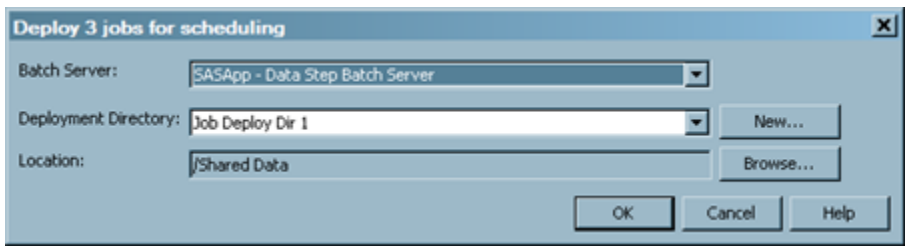
1. Right-click the job that you want to deploy. Then, select **Scheduling** ⇨ **Deploy** in the pop-up menu to access the Deploy for a job for scheduling window. The following display shows the window if you select only one job for deployment.

   Display 10.1 *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 10.2 *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.

   Note: If you want to run multiple deployed jobs on multiple Data Step Batch Servers, you need to create a separate deployment directory for each Data Step Batch Server. If you run multiple deployed jobs that are defined for different Data Step Batch Servers in a single deployment directory, all of the jobs are run on the Data Step Batch Server that is defined for the first job that is run. This process even occurs when the **Preserve deployed value** option in the properties window for the deployed job is enabled for all of the jobs in the directory.

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.

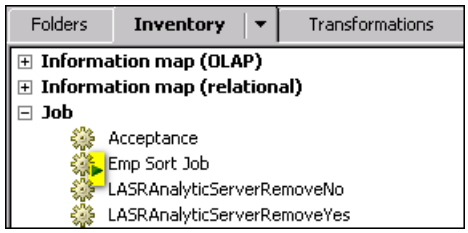
4. Check the **Deployment Directory** field to ensure that the deployed job is stored in the appropriate directory. If the wrong directory is displayed, select another director from the drop-down list, or click **New** to create a new directory if you have permission to create directories on the server.

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**.

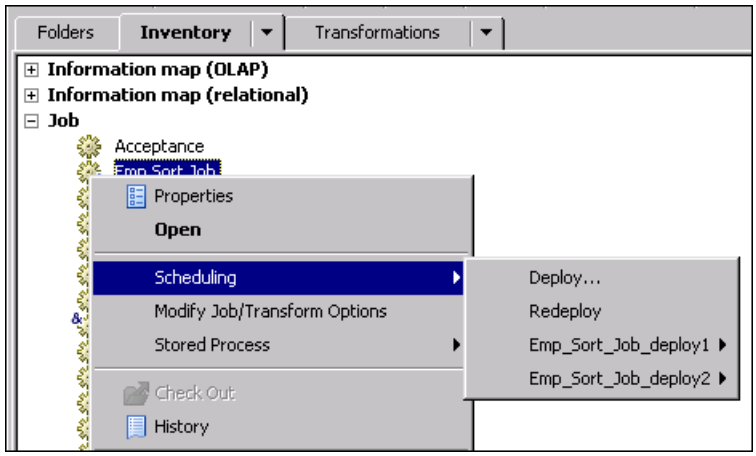
Code is generated for the selected job or jobs and is saved to the directory that is specified in the **Deployment Directory** field. Metadata about the deployed jobs is saved to the current SAS Metadata Server. A status window is displayed and indicates whether the deployment was successful. In the Inventory tree, metadata for the deployed job is added to the **Deployed job** folder. Also, a blue triangle overlay is added to the icon for the job in the **Job** folder. In the next display, the icon for a job named **Emp Sort Job** has the blue triangle overlay.



This job is now available for scheduling.

A Job Can Be Deployed to Multiple Locations

A single job can be deployed to multiple locations. Each deployed instance has its own name. For example, the following display shows that a job named **Emp Sort Job** has two deployed instances: **Emp_Sort_Job_deploy1** and **Emp_Sort_Job_deploy2**.



Using a Command Line to Deploy Jobs

Problem

You want to deploy jobs in batch mode.

Solution

You can use the command-line deployment tool. This tool enables you run an Ant script to deploy many jobs at once through a simple command-line interface. You can create a manifest file containing one or more paths and pass that file into the deployment script along with the appropriate parameters. Use other software to schedule the job for execution. For more information, see *Scheduling in SAS*.

Perform the following tasks:

- "Review the Prerequisites" on page 230
- "Prepare for Windows Deployment" on page 232
- "Prepare for Other Operating System Deployments" on page 232
- "Prepare the Manifest" on page 232
- "Run the Script" on page 233

Note: The batch deployment feature does not work when the host name contains a hyphen (-) character.

Tasks

Review the Prerequisites

In order to use the command-line tool to deploy jobs, you must meet the prerequisites described in "Prerequisites for Deploying a Job for Scheduling" on page 227. You must also gather server addresses, passwords, and other information that you will need.

For example, you should review the installation properties listed in the following table

Table 10.1 *Installation Properties*

Property	Value
-InstallRoot	C:\Program Files\SASHome
-PlatformVJR	${InstallRoot}\SASVersionedJarRepository\eclipse
-WorkingDirectory	${PlatformVJR}\plugins \sas.etl.migration.batch_904000.1.0.20121105190000_ v940>
-PickList	${InstallRoot}\SASDataIntegrationStudio\4.7\plugins \sas.etl.migration.batch\picklist
-SASLauncher	${PlatformVJR}\plugins\sas.launcher.jar
-DISLocation	${InstallRoot}\SASDataIntegrationStudio\4.7

Specify the following properties for the SAS Metadata Server and the SAS Application Server that will be used to deploy the job in the deploySASJobs.xml file:

Table 10.2 *Server-Specific Properties*

Property	Description
The properties that follow are specified for the SAS Metadata Server that will be used to deploy the job:	
-metaserver	IP address of the SAS Metadata Server.
-metaserverid	Metadata ID for the SAS Application Server that will be used to deploy the job. Typically, this will be the metadata ID for SASApp.
-metaport	Port for the specified server.
-metarepository	Metadata ID for the SAS Metadata Server repository.
-metauser	A registered user of the SAS Metadata Server.
-metapass	Password for the specified user.
The properties that follow are specified for the SAS Application Server that will be used to deploy the job:	
-servermachine	IP address of the SAS Workspace Server that will deploy the job.
-serverport	Port for the specified server.
-username	A registered user of the SAS Workspace Server. This user must be an external user, such as a real person. It cannot be an internal user such as the SAS Trusted User.
-password |	Password for the specified user.
-batchserver	Name of the batch server to use for deployment. The default value is the SAS DATA Step Batch Server component of the SAS Application Server.
-sourcedir	Location of the SAS source code.
-manifest	Location of the manifest file.
-deploymentdir	Fully qualified path to the job deployment directory that is specified in the Ant script. This directory also must be specified in the metadata for the SAS Application Server that is associated with the SAS Workspace Server. To view this directory, you can use the Schedule Manager module in SAS Management Console. Alternatively, an administrator can supply you with the value.

Prepare for Windows Deployment

Perform the following steps to prepare the script:

1. Locate the deploySASJobs.xml Ant script that is deployed by default in the **SASHome/SASDataIntegrationStudio/<version>/batchdeploy** directory. Consider creating a copy of the file to configure so that the original retains the default settings.
2. Open your copy of deploySASJobs.xml.
3. Enter appropriate values for the following arguments. See "Review the Prerequisites" on page 230 for information about appropriate values.
4. Save the values and close the deploySASJobs.xml script.

Prepare for Other Operating System Deployments

If you want to use the command-line deployment in an operating system other than Microsoft Windows, verify that a folder called **Security** is installed in the installation folder for SAS Management Console. SAS Management Console must be installed on the computer where you will be running command-line deployment.

If you do not see a **Security** folder, then copy the **<*PlatformVJR*>** ⇨ **plugins** ⇨ **sas.etl.migration.batch <*version*>** folder from the computer where SAS Data Integration Studio is installed. Put these folders in the installation folder for SAS Management Console. After you do this, update the paths to these locations inside the deploySASJobs.xml file. See "Review the Prerequisites" on page 230 for information about appropriate values for other operating system deployments.

Note: On z/OS, running the command-line deployment tool might cause a Java Virtual Machine (JVM) initialization issue or JVM out of memory issue. If you see these Java error messages, comment out the <jvmarg value="-Xmx768m"/> line in the deploySASJobs.xml file.

Prepare the Manifest

Once you have prepared the Ant script, you must create the manifest file that lists the jobs that you want to deploy. The manifest file should be formatted as a standard text file, but name and extension do not matter. Each line of the manifest file is deployed individually. A line can specify a directory to deploy, a specific job to deploy, or a regular expression to deploy.

When you specify a directory, all of the jobs within that directory are deployed. When you specify a specific job, only the matching job is deployed (if it exists). When you specify a regular expression, any job in the given path that matches the regular expression passed in by the user is deployed. You can also deploy jobs recursively by adding the argument *-r* to the end of the line. When recursion is enabled, all subdirectories of the given directory are deployed in the same manner as the given directory.

Perform the following steps to prepare the manifest:

1. Create a new text file to contain the manifest and save in the location specified in the deployment script file. Note that you can also modify one of the sample manifest files provided in the **SASHome** ⇨ **SASDataIntegrationStudio** ⇨ **<version>** ⇨ **batchdeploy** directory.
2. Enter the information about the jobs that you want to deploy. For example, samplemanifest2 contains the following information:

```
/SampleFolder/SampleJob -r
```

```
/SampleFolder/^[A-Za-z]+$ -r
```

3. Save the manifest file.

Note: A line in the manifest should not end with a forward slash (/) unless the slash is followed by a valid expression. If the line ends with a forward slash and no expression, the job will run, return a 0 return code, but nothing will be deployed.

Run the Script

You must install Ant, which you can obtain from **`http://ant.apache.org/`**, to run a deployment script.

Perform the following steps to run the deployment script from a command line:

1. Open a command line.
2. Run the script in Ant, by specifying the script file as follows:

```
ant –f deploySASJobs.xml
```

.

Note: If you do not want to store your credentials in an openly available script file, you can also pass the user name and password through the command line. Use the following syntax:

```
ant -f deploySASJobs.xml -Dusername=mysuername -Dpassword=mypassword
```

(where each argument is preceded by a **`-D`**, the name of the parameter matches the name being set inside the ANT script, and key and value pairs are separated by the equal sign).

Redeploying Jobs for Scheduling

Problem

After a job was deployed for scheduling, either the job or the computing environment changed. For example, additional transformations might have been added to the process flow for the job. Alternatively, the job might have been exported to another environment where the servers and libraries are different.

Solution

Use a redeployment option in SAS Data Integration Studio to regenerate the code for one or more deployed jobs and save the new code to a job deployment directory. The redeployed jobs can then be rescheduled.

Tasks

Redeploy All Deployed Jobs in the Current Repository

Perform the following steps to redeploy all deployed jobs in the current repository:

1. Select **Tools** ⇨ **Redeploy for Scheduling** in the menu bar. Any jobs that have been deployed are found.

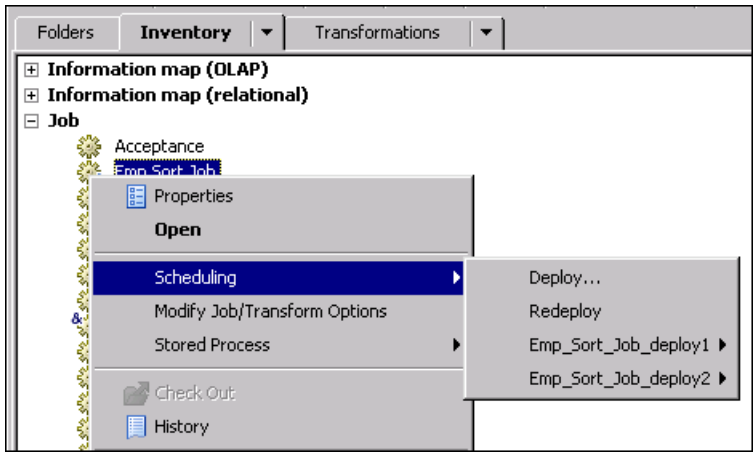
2. Click **Yes** to continue the redeployment process. The Redeployed scheduled jobs window is displayed. Verify that the appropriate options have been set, and click **OK** to redeploy the jobs. Code is generated for all deployed jobs and saved to the job deployment directory for the SAS Application 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.

Redeploy All Deployed Instances of a Selected Job

As described in "A Job Can Be Deployed to Multiple Locations" on page 229, a single job can be deployed to multiple locations. Each deployed instance has its own name. Perform the following steps to redeploy all instances of a selected job. If the job has one deployed instance, that one instance will be redeployed.

1. Expand the **Job** folder in the Inventory tree. The icons for deployed jobs have a blue triangle overlay.
2. Right-click a deployed job. The pop-up menu lists the name of each deployed instance. For example, the following display shows that a job named **Emp Sort Job** has two deployed instances: **Emp_Sort_Job_deploy1** and **Emp_Sort_Job_deploy2**.



3. Select **Redeploy** to redeploy all deployed instances of the selected job.

Redeploy One Instance of a Selected Job

As described in the previous topic, a single job can be deployed to multiple locations. Each deployed instance has its own name. You can redeploy one deployed instance of a selected job from either the **Job** folder or the **Deployed Job** folder.

Perform the following steps in the **Job** folder of the Inventory tree:

1. Expand the **Job** folder. The icons for deployed jobs have a blue triangle overlay.
2. Right-click a deployed job. The pop-up menu lists the name of each deployed instance. For example, a job named **Emp Sort Job** could have two deployed instances: **Emp_Sort_Job_deploy1** and **Emp_Sort_Job_deploy2**.
3. Select the deployed instance that you want to redeploy, such as **Emp_Sort_Job_deploy1**.
4. Select **Redeploy**.

Alternatively, you can perform the following steps in the **Deployed Job** folder of the Inventory tree.

1. Expand the **Deployed Job** folder.

2. Right-click a deployed instance that you want to redeploy, such as **Emp_Sort_Job_deploy1**.
3. Select **Redeploy**.

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 235 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 150. For information about scheduling prerequisites, see "Prerequisites for Deploying a Job for Scheduling" on page 227.

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 them 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 238
- "Redeploying Jobs to Stored Processes" on page 241
- "Viewing or Updating Stored Process Metadata" on page 242

See also "Prerequisites for Deploying a Job as a Stored Process" on page 237. For information about creating stored processes that are not based on deployed jobs, see "Working with Stored Processes" on page 43.

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. You can deploy an existing job as a version 1.0 or version 2.0 stored process. For more information about the differences between the versions, see "Working with Stored Processes" on page 43.

Note that when you deploy a job as a stored process, the generated code for the stored process always begins with these lines:

```
*ProcessBody;
%stpbegin;
```

If you want to specify code that should come before these two lines when a job is deployed as a stored process, then set the **Stored process pre-process code** option for the job. To access this option, display the properties window for the job and select **Options** ⇨ **General** section. Specify the desired code in the **Stored process pre-process code** option.

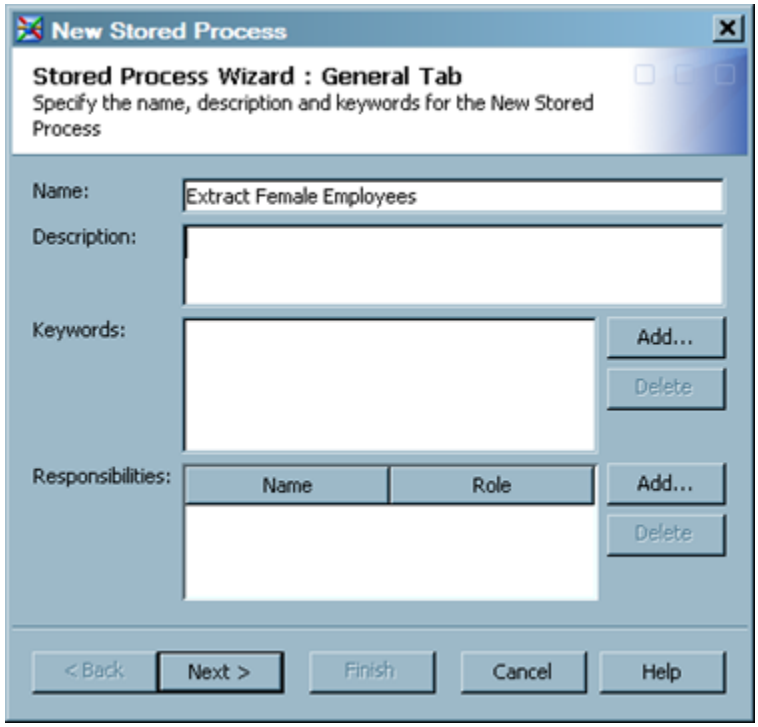
Tasks

Deploy a Job as a Version 1.0 Stored Process

You might want to deploy a job as a version 1.0 stored process in order to run it on an older server (a server with a version prior to SAS 9.3). Perform the following steps:

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 9.2** from the pop-up menu. The first window of the Stored Process wizard is displayed.

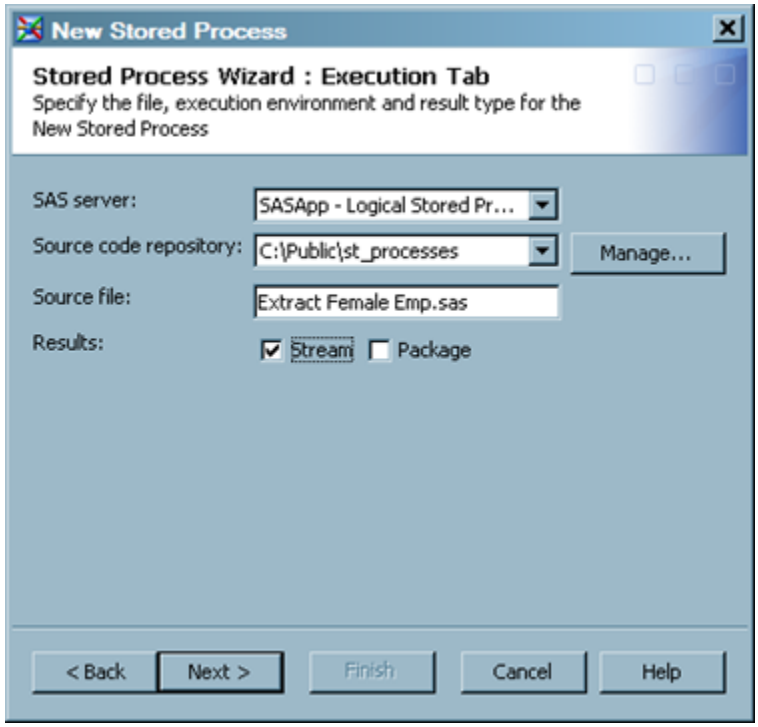
Display 10.3 *General Tab*



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 10.4 *Execution Tab*



Click **Next** to access the **Parameters** tab, where you can specify any parameters that you need for the stored process.

4. Click **Next** to access the **Data** tab, 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.

Deploy a Job as a Version 2.0 Stored Process

You might want to deploy a job as a version 2.0 stored process in order to run it on a SAS 9.3 or later server. Perform the following steps:

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 9.3** from the pop-up menu. The **New 9.3** selection is appropriate for any version 2.0 stored process, whether it will run on a SAS 9.3 server or a later server. The first window of the Stored Process wizard is displayed.
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. For more information about the additional servers available for version 2 stored processes, see "Working with Stored Processes" on page 43.

4. Click **Next** to access the **Parameters** tab, where you can specify any parameters that you need for the stored process.
5. Click **Next** to access the Data screen, where you can specify any data sources and targets that are used by the stored process. For information about data sources and targets, click **Help** in the Modify Data Source and Modify Data Target windows. To access these windows, select a source or target and click **Edit**.
6. 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 241.

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 242.

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 244.
- Deploy the job as a stored process. See "Deploying Jobs as Stored Processes" on page 238.
- Deploy the stored process for execution by a Web service client. See "Deploying a Stored Process as a Web Service" on page 251.

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 243 and the specific requirements that are described in "Requirements for Web Service Jobs" on page 244.

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 237 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 10.5 *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 244. One way to meet these requirements is to create a job with a process flow similar to the flow in the following display.

Display 10.6 *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 245
- "Create XML Libraries for the Inputs and Outputs" on page 246
- "Register the XML Inputs and Outputs" on page 247
- "Create a Generated Transformation That Produces the Desired Output" on page 247
- "Create the Job" on page 248

It is assumed that the general prerequisites have been met, as described in "Prerequisites for Web Service Jobs" on page 243.

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 244, 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 Farenheit </CalculatedTemperature>
   </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 81.
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 277.

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.
7. 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:

1. 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.
2. Enter a name for the job. The sample job is named **Convert Temp Job**. Click **OK**. An empty job opens in the Job Editor.
3. Drag your custom transformation from a tree view into the job.
4. 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.
5. 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 244.
6. 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.

7. 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.
8. 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.

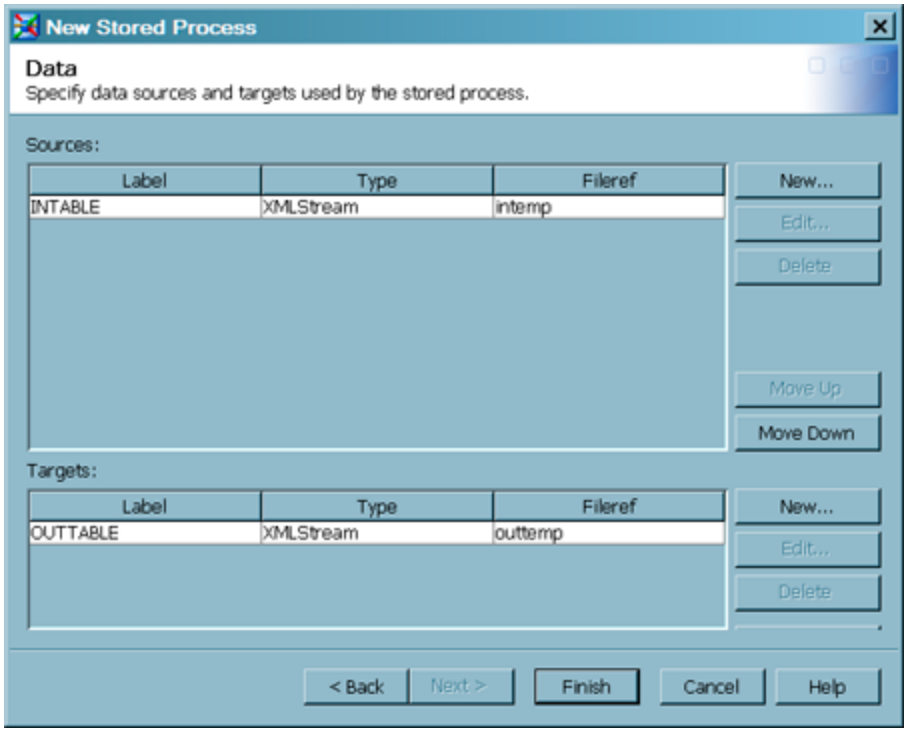
   Source code repository specifies the path where the SAS server saves the source code for the stored process. For the sample job, this path is c:\public\st_processes.

   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 10.7 *Data Page of the New Stored Process Wizard*



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

6. 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 249.

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 11

Working with Versions

About Versions **253**

Prerequisites for Version Control **254**
Overview 254
Usage Notes 254

Example Setup for an Apache Subversion (SVN) Server **255**
Overview 255
Remove the CVS Plug-in 255
Install and Configure the Apache Subversion (SVN) Server 255
Specify the SVN Server on the SVN Plug-in Tab 256

Creating a Version **257**
Problem 257
Solution 257
Tasks 257

Reviewing and Managing Versions **258**
Problem 258
Solution 258
Tasks 258

Comparing Versions **260**
Problem 260
Solution 260
Tasks 260

About Versions

Version control enables you track changes that are happening over time to SAS Data Integration Studio objects. Versioning works by moving content such as jobs and other objects into a file and archiving that file in a versioning system. SAS Data Integration Studio creates the file as a SAS Package and writes it into the source management system. To bring content back into the repository, SAS Data Integration Studio retrieves the content stored in the source management system and places it back into the SAS metadata repository. In this way, you can create different versions of content and restore previous versions of content when needed.

Objects can be versioned independently or with other objects to make up a package of related content. This ability enables you to archive sets of objects that are logically related, such as all of the content in a project. You can also choose to generate source code for a job and store it along with the job as text content. This function makes it easy

to see the source code associated with a specific version of a job. You can view archived results of any object to see when it was last versioned. This function lets you identify previous version of objects that you might want to restore and maintain a history about changes.

After you have created versions of a selected object, you can access the versioned objects in the Archived SAS Packages window. The window displays a list of all the versions of all the archived objects so that you can access and maintain the versions. You can select an object and view the differences between versions of the selected object or between an archived version and the current version of that object.

Prerequisites for Version Control

Overview

The following prerequisites must be met before you can use the version control features in SAS Data Integration Studio:

- A third-party version control server must be installed in a location that is accessible to SAS Data Integration Studio. The following servers are supported by default: Apache Subversion (SVN) server 1.6.x, and Concurrent Versions System server CVS 1.11.x, CVSNT2.0.x, CVSNT2.5.x, and CVSNT2.8.x. For an example of how a third-party version control server can be installed and configured, see "Example Setup for an Apache Subversion (SVN) Server" on page 255.
- By default, the global Options window in SAS Data Integration Studio includes a **CVS Plug-in** tab and an **SVN Plug-in** tab. Information about your version control server must be specified in the appropriate plug-in tab. For an example of this task, see "Specify the SVN Server on the SVN Plug-in Tab" on page 256.
- If you have installed a CVS server, specify information about that server on the **CVS Plug-in** tab.
- If you have installed an SVN server, specify information about that server on the **SVN Plug-in** tab. You must also remove the CVS plug-in from the installation path. SAS Data Integration Studio uses the first plug-in that it finds in the installation path. It will always find the CVS plug-in first unless it is removed from the installation path. The general method for removing plug-ins is described in "Remove the CVS Plug-in" on page 255.

  The SVN plug-in works only with command-line SVN clients such as the 32–bit version of Subversion for Windows. The plug-in does not work with a graphic user interface-only SVN client such as rapidSVN. It does work with packages such as VisualSVN Server that include a command-line client. For more information, consult the user documentation provided with the client that you select. If the SVN command-line client is not available in your existing SVN package, you should consider obtaining a command-line client from **`http://subversion.apache.org/packages.html`**. This command-line client enables you to use SAS Data Integration Studio with the SVN server.

Usage Notes

SAS Data Integration Studio can use only one version control server at a time. It uses only the first server plug-in that it finds in the installation path.

Version control cannot be used on the contents of a project repository in SAS Data Integration Studio.

If you want to use a version control server other than a CVS server or an SVN server, you must create a custom plug-in for that server. There is a documented application programming interface (API) to integrate other versioning systems with SAS Data Integration Studio. See this API documentation at **http://support.sas.com/rnd/gendoc/versioncontrol/43/en/**. Keep in mind that SAS Data Integration Studio uses only the first plug-in that it finds in the installation path. You must either name your custom plug-in a name that can be found before all other plug-ins, or you must remove other plug-ins from the installation path.

Example Setup for an Apache Subversion (SVN) Server

Overview

This topic describes one way to install and set up an Apache Subversion (SVN) server. You can use any Subversion server or server package that meets the criteria that are described in "Prerequisites for Version Control" on page 254.

Remove the CVS Plug-in

By default, the global Options window in SAS Data Integration Studio enables you to specify either a CVS server or an SVN server for version control. However, if you prefer to use an SVN server, you must remove the JAR files for the CVS plug-in from the installation path. SAS Data Integration Studio uses only the first plug-in that it finds in the installation path. It always finds the CVS plug-in first unless it is removed from the installation path.

Perform the following steps:

1. Navigate to the default path for SAS Data Integration Studio plug-ins. Here is an example path: **C:\Program Files\SASHome\SASVersionedJarRepository\eclipse\plugins**.
2. Find all folders with this root filename: **sas.dbuilder.versioncontrol.cvsplugin.***
3. Move these folders out of the plug-ins folder.

The next time you display SAS Data Integration Studio, the **CVS Plug-in** tab does not appear in the global Options window.

Install and Configure the Apache Subversion (SVN) Server

Perform the following steps:

1. Download the Apache Subversion server or server package from the appropriate website.
2. Install and configure the SVN server according to the instructions provided with the software.

3. Create an SVN server repository according to the instructions provided with the software.
4. Create an SVN server user and password according to the instructions provided with the software.
5. Perform any other configuration that is required for the SVN server, such as specifying an SSL certificate.

Specify the SVN Server on the SVN Plug-in Tab

Perform these steps to specify information about an SVN server and client on the **SVN Plug-in** tab:

1. Run SAS Data Integration Studio.
2. Select **Tools** ⇨ **Options** ⇨ **SVN Plug-in** tab.
3. Specify the appropriate values on the **SVN Plug-in** tab. The next table provides some example values.

Field	Example Value	Description
Program Path	`C:\Program Files (x86)\VisualSVN Server\bin\svn.exe`	Path to the client for the version control server.
Local Root Path	`C:\Users\<user_name>\AppData\Local\Temp\2\`	Path to a folder where temporary data can be stored.
Server	*myVCServer.com*	Version control server.
Repository	`/svn/svn_repos`	Path to the repository for the version control server.
User Name	*svn_user1*	Version control server user.
Password	******	Password for the version control server user.
Port	*443*	Connection port for the version control server. See the server documentation for more information about ports.
Type	*HTTPS*	Connection type for the version control server. See the server documentation for more information about connection types.

4. Click **Test Connection** to verify the values on this tab. If the test connection is successful, you are ready to use the version control features in SAS Data Integration Studio.

5. Click **OK** to save the values on this tab.

Creating a Version

Problem

You want to create a version of a selected object in SAS Data Integration Studio. You can use the version to track changes to the object.

Solution

You can archive the object as a SAS package. For example, you can use archiving to create a version of a job.

Tasks

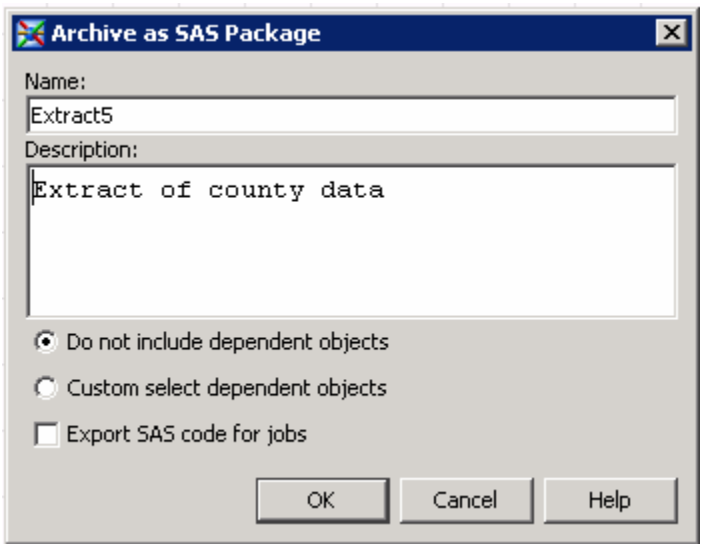
Archive an Object as a SAS Package

Perform the following steps to archive an object:

1. Right-click the object or objects that you need to archive. For example, you can archive a job that extracts data from a source table, such as Extract county data. Then, you can track how the data changes by comparing the versions that you create over time.
2. Click **Archive as SAS Package** in the pop-up menu.
3. Enter an appropriate name and description for the object in the Archive as SAS Package window. The version control system uses this name to archive the object and increment the version numbers. If you change the name of the archive, you will start a new series of version numbers.

   The window is shown in the following display:

Display 11.1 *Archive as SAS Package Window*



Note that dependent objects are not included in this archive, and SAS code is not exported for the job.

By default, the export process does not include objects that are depended on by the objects that you are exporting. If you select **Custom select dependent objects**, the export wizard launches. The wizard enables you to select objects with more precision.

Selecting **Export SAS code for jobs** creates a note object for each job being archived. This action sets the text of that note to the generated SAS code for the given job. These note objects are then archived along with the jobs.

4. Click **OK** to process the archive. You can review the log from the export wizard when the processing is completed. You should check the log to ensure that the archive is submitted to the version control system.

   Note: The archive contains the latest saved version of your object. Be sure to save your changes before you create the archive.

Reviewing and Managing Versions

Problem

You want to review and manage the versions that you have created of a SAS Data Integration Studio object.

Solution

You review a list of all of the SAS packages archived on your source control server.

You can also perform the following tasks:

- "Locate an Archived Version" on page 258
- "Edit an Archived Version" on page 259
- "Re-Archive an Archived Version" on page 259
- "Manage Archived Versions" on page 260

Tasks

Locate an Archived Version

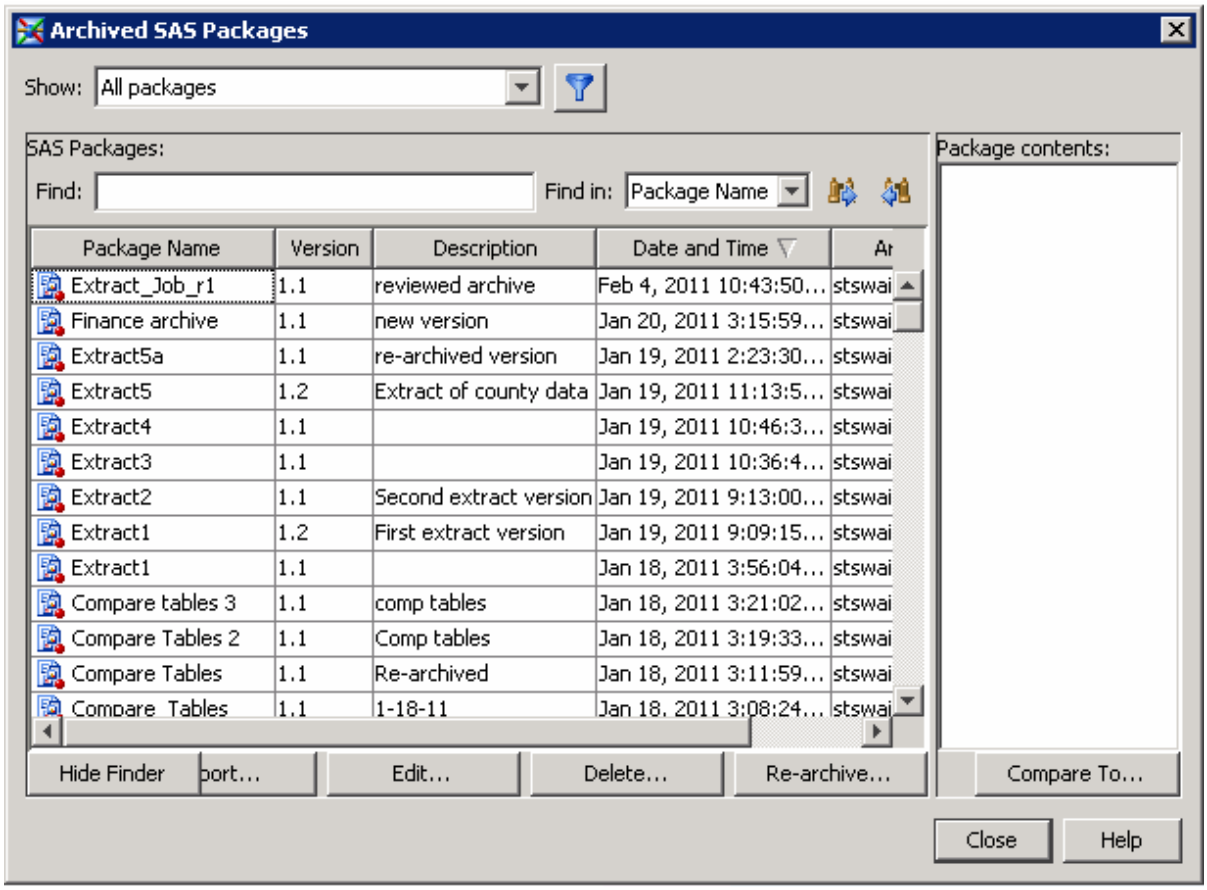
Use the Archived SAS Packages window to locate archived objects such as the Extract county data job.

Perform the following steps:

1. Right-click an archived object and click **Archived SAS Packages** in the pop-up window.
2. Click the drop-down menu in the **Show** field and select **All packages**. Note that you can click **Filter** to filter the list by package name, description, or archivist. You can also click **Show Finder** to access fields that you can use to search the list.

The Archived SAS Packages window is shown in the following display:

Display 11.2 *Archived SAS Packages Window*



Edit an Archived Version

Use the Edit SAS Package window to change the name and description of an archived object.

Perform the following steps:

1. Select an archived object in the window. For example, you can select the **Extract Job** object.
2. Click **Edit**.
3. Enter a name and description in the Edit SAS Package window. You can rename the version *Extract Job_r1* and enter a description of *reviewed archive*.
4. Click **OK** to save the edited version and add it to the list in the Archived SAS Packages window.

Re-Archive an Archived Version

Use the Archive as SAS Package window to re-archive a selected archived object. The re-archive process searches through the selected archive and tries to find the current version of all of those items. Then it packages all of the items that it finds and exports them.

This function is useful when you have a set of objects that change internally but have dependent objects that do not change. You can then easily create a new archived version with the same contents without having to search for the dependent objects.

Perform the following steps:

1. Select an archived object. For example, you can select the **Extract5** object.
2. Click **Re-archive**.
3. Enter a name and description in the Archive as SAS Package window. You can name the re-archived version *Extract5a* and enter a description of *re-archived version*.
4. Click **OK** to save the re-archived version and add it to the list in the Archived SAS Packages window. The re-archived object in this example has a 1.1 version number. The version number is incremented from the current number assigned by the version control system.

Manage Archived Versions

You can also perform the following management functions on the archived versions listed in the Archived SAS Packages window:

- Import: Opens the Import SAS Package Wizard.
- Delete: Deletes a selected archive. Note that deleting a specific version of an object does not delete the corresponding archived package for that object in the version control system.
- Compare To: Enables you to compare a selected object to another archive or object. For more information, see "Comparing Versions" on page 260.

Comparing Versions

Problem

You want to compare a selected object to another archive or object.

Solution

You can use the compare to function in the Archived SAS Packages window.

Tasks

Use the compare function to compare a selected object to another archive or object.

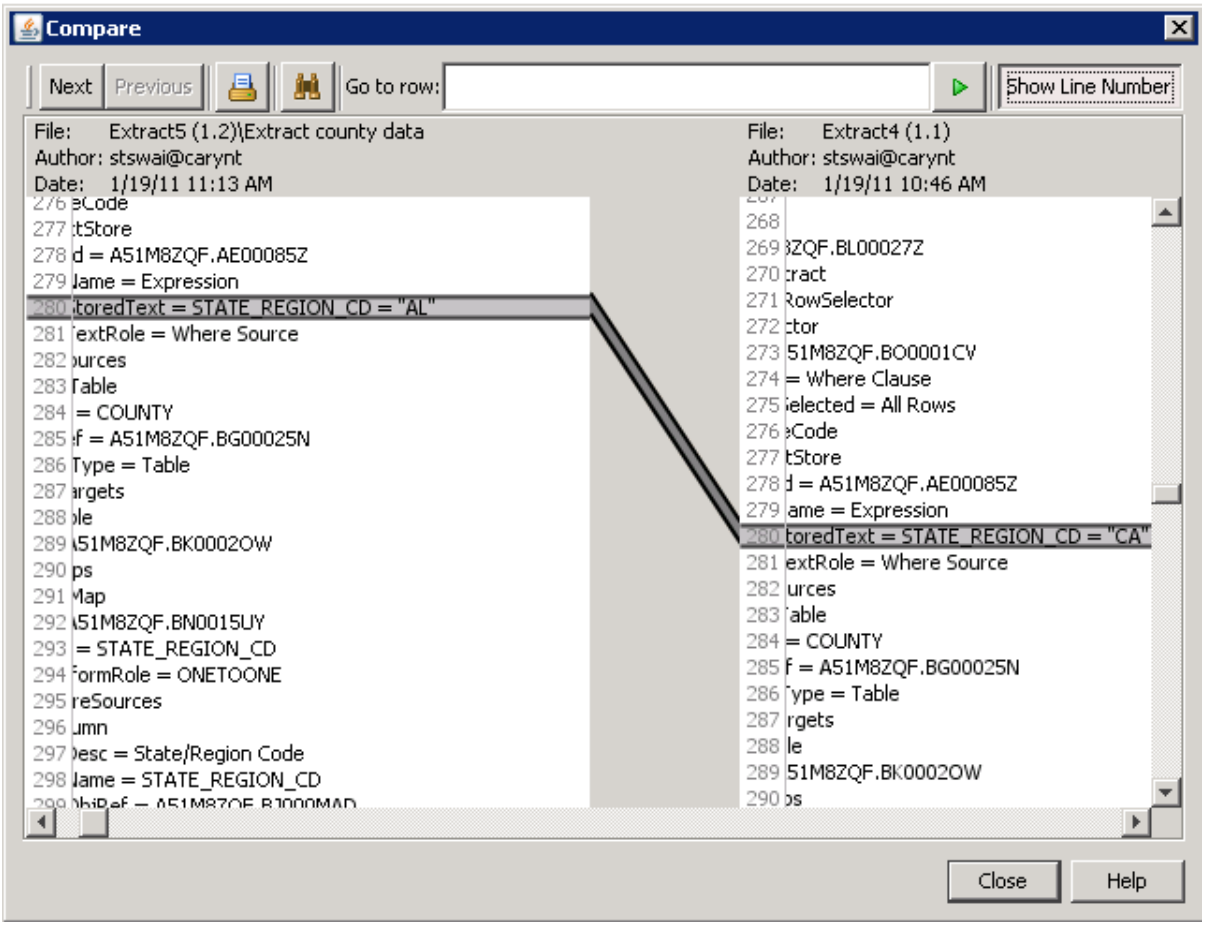
Perform the following steps:

1. Right-click the first archive that you want to compare. For example, you can select the **Extract5** archive.
2. Navigate through the folder hierarchy in the **Package Contents** pane until you see the root object for the archive that you just selected. Then, select the object (the Extract county data job in this case).
3. Click the **Compare To** button, which enables you compare two objects that share the same metadata ID.
4. Add the path to the second object in the comparison to the Compare To window. For example, you can enter the **Extract4** archive in the **Other archive** field. (The **Browse** button enables you to select from a list of all of the archives associated with the Extract county data job).

5. Click **OK** to create and review the comparison.

   The comparison is shown in the following display:

Display 11.3 *Compare Window*



   Note that the name, author, and date of the archives are listed above the comparison. You can also see that the differences between the selected archives are clearly highlighted.

6. Click **Close** to return to the Archived SAS Packages window.

Chapter 12

Working with Generated Code

About Code Generated for Jobs **263**
Overview 263
LIBNAME Statements 264
SYSLAST Macro Statements 264
Remote Connection Statements 265
Macro Variables 265
User Credentials in Generated Code 266

Displaying the Code Generated for a Job **267**
Problem 267
Solution 267
Tasks 267

Displaying the Code Generated for a Transformation **268**
Problem 268
Solution 268
Tasks 268

Specifying Options for Jobs **268**
Problem 268
Solution 268
Tasks 268

Specifying Options for a Transformation **269**
Problem 269
Solution 269
Tasks 269

Modifying Configuration Files or SAS Start Commands for Application Servers **269**

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 268 and "Specifying Options for a Transformation" on page 269.

LIBNAME Statements

When SAS Data Integration Studio generates code for a job, a library is considered local or remote in relation to the SAS Application Server that executes the job. If the library is stored on one of the machines that is specified in the metadata for the SAS Application Server that executes the job, it is local. Otherwise, it is remote.

SAS Data Integration Studio generates the appropriate LIBNAME statements for local and remote libraries.

Here is the syntax that is generated for a local library:

```
libname libref <enginer> <"lib-specification"> <connectionOptions>
<libraryOptions>
<schema=databaseSchema>
<user=userID>
<password=password>;
```

Here is the syntax that is generated for a remote library:

```
options
comamid=connection_type;
%let remote_session_id=host_name <host_port>;
signon
remote_session_id<user=userID>
<password=password>;
rsubmit remote_session_id;
   libname <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.

See also "User Credentials in Generated Code " on page 266.

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 pass-through statements.
_INPUT_engine	Specifies the library engine. This macro variable can be used for explicit pass-through 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
_INPUT_options	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.
_INPUT_alter	Specifies an alter or password option text so that the table can be deleted or altered. This macro variable is a subset of the _options string.
_INPUT_path	Specifies the location of table on metadata server.
_INPUT_type	Specifies a macro given by the prompting framework. This macro variable should always be 1 for usage with SAS Data Integration Studio.
jobID	Specifies the unique metadata ID code that is given to the job when the job is first created.
JOB_RC	Specifies a status handling macro variable that is set and reset (as the job runs) to be the maximum return code value (&trans_rc) of the completed transformations.
_OUTPUT_count	Specifies the count of output tables.
SYSLAST	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.
trans_rc	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. The _INPUT and _OUTPUT variables 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 it is the default if a name was not specified for the macro.

Users can add references to any of these in user-written code. See "About User-Written Code" on page 271. 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

By default, SAS Data Integration Studio looks up user credentials rather than explicitly including them in the code that it generates when it accesses tables in a library. This behavior can be changed by selecting **Tools** ⇨ **Options** from the main menu, clicking the **General** tab, and then selecting or deselecting the **Use runtime lookup for credentials for statements requiring credentials** check box. When this option is

selected, the code generated by SAS Data Integration Studio does not contain user names and passwords for SAS Connect sign-on statements. Instead, the user name and password are looked up at run time, using the Authentication Domain that has been specified for the user is his or her metadata identify on the SAS Metadata Server.

If this option is not selected, 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 pass-through, 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 435.

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 13

Working with User-Written Code

About User-Written Code **271**

Adding User-Written Code to the Precode and Postcode Tab **272**
Problem 272
Solution 272
Tasks 272

Adding a User Written Code Transformation to a Job **274**
Problem 274
Solution 274
Tasks 274

Creating and Using a Generated Transformation **277**
Problem 277
Solution 277
Tasks 278

Updating a Generated Transformation **284**
Problem 284
Solution 284
Tasks 284

Editing the Generated Code for a Job or Transformation **286**
Problem 286
Solution 286
Tasks 286

Replacing the Generated Code for a Job or Transformation **287**
Problem 287
Solution 287
Tasks 287

Converting a SAS Code File to a Job **288**
Problem 288
Solution 288
Tasks 289

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
- convert a SAS program and import into SAS Data Integration Studio as a job

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.

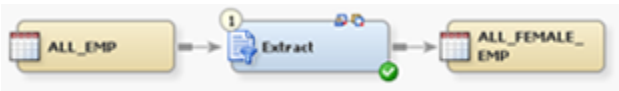
Tasks

Add the User-Written Code to the Precode or Postcode Field

Perform the following steps to insert the user-written code:

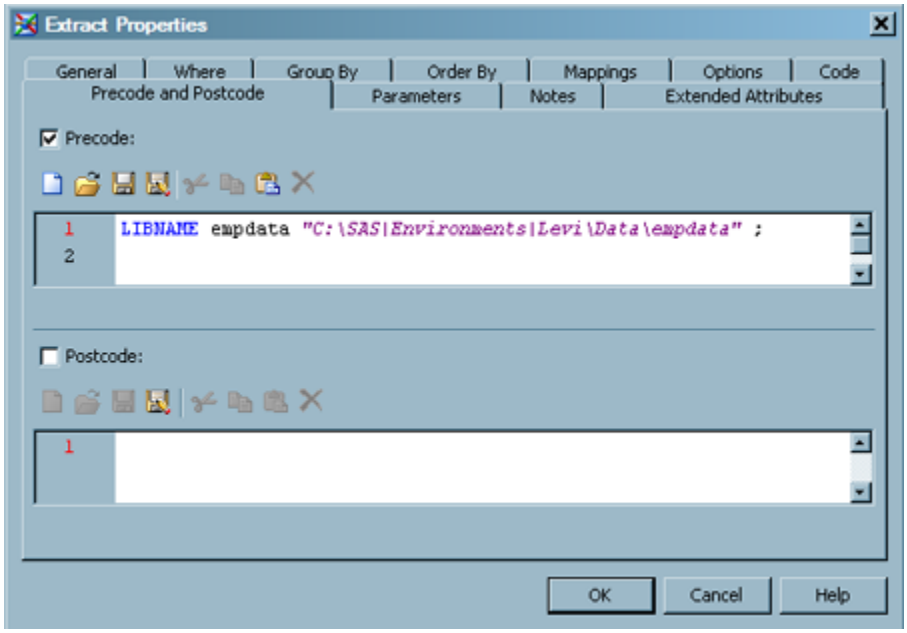
1. Create a job, or open an existing job. The sample job, which is named Extract Job, is shown in the following display.

   Display 13.1 *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 13.2 *Sample User-Written Precode*

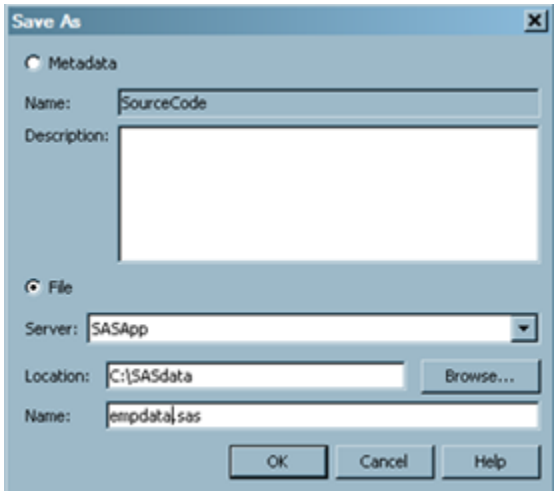


Save the User-Written Code to a File

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 13.3 *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 13.4 *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 274
- “Add User-Written Code to the User Written Code Transformation and Map Columns” on page 275
- “Run the Job” on page 276
- “View the Output” on page 276

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.
4. 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 150.

The flow for the sample job is shown in the following display.

Display 13.5 *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:

1. Write SAS code and test it to ensure that it produces the required output. The following code was written for the sample job:

```
data
&_OUTPUT;
  set &SYSLAST;
  length sex $1;
  if gender = "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.

2. 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**.
3. 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.
4. Place the cursor in an editable section of the **Code** tab.

5. Enter the SAS code.
6. 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

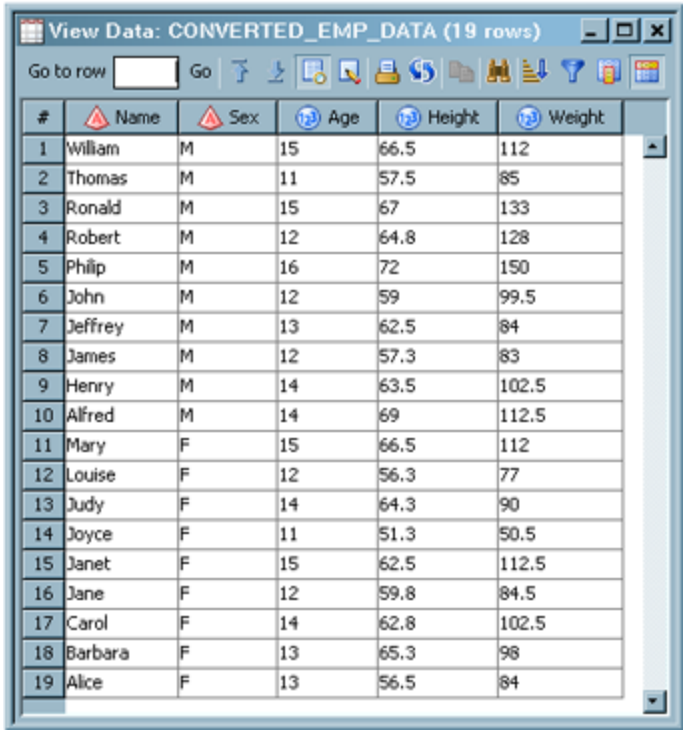
Perform the following steps to submit and run the job:

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 13.6 *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 278
- “Use a Generated Transformation in a Job” on page 281

Tasks

Create a Generated Transformation

Perform the following steps to create a generated transformation:

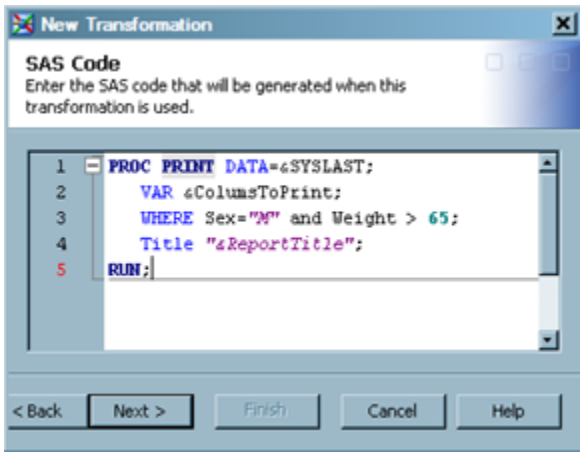
1. Right-click the destination folder for the generated transformation.

   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.

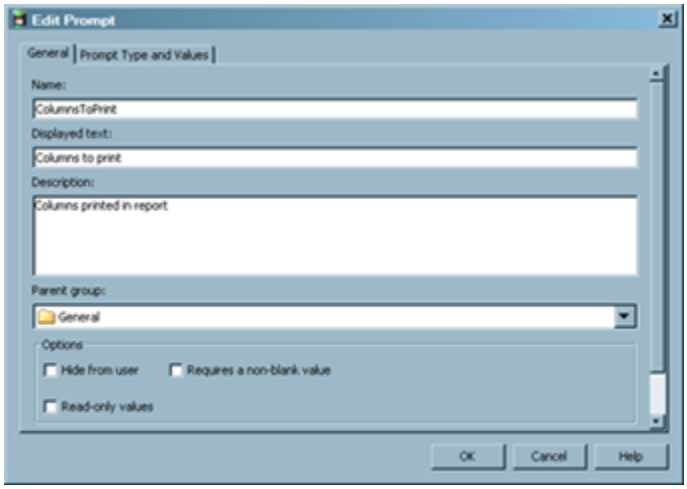
   Display 13.7 *Sample Transformation Code Page*



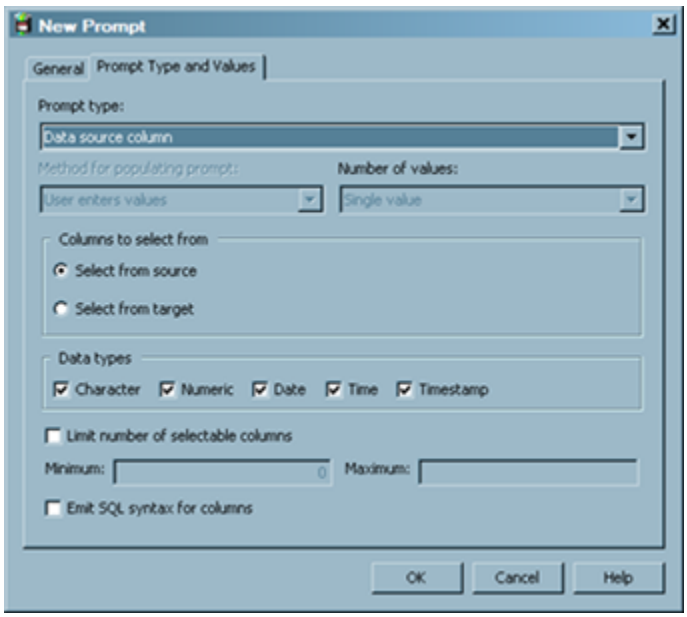
   A number of macro variables appear in this sample code. One of these macro variables, &SYSLAST, is normally available and refers to the last data set created. The transformation also includes other macro variables, such as &ColumnsToPrint and &ReportTitle. The type of each such variable is defined in 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 13.8 *General Prompt Tab for the Columns to Print Option*

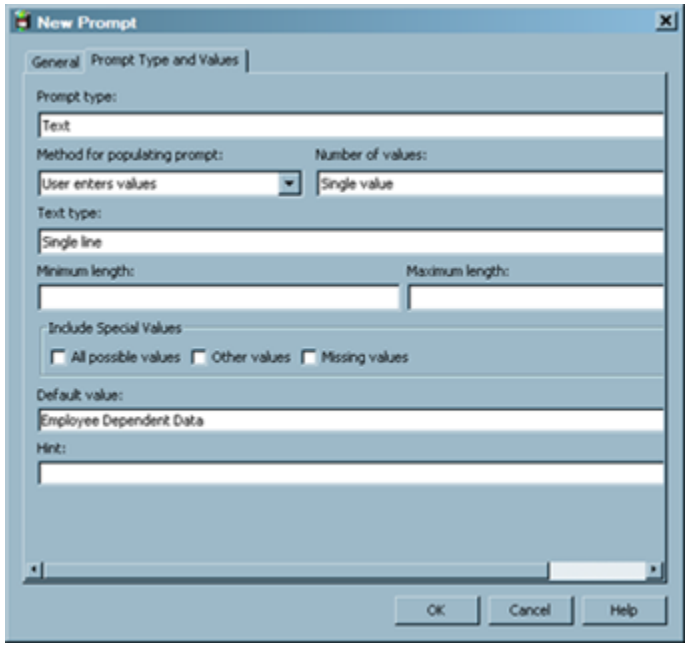


Display 13.9 *Prompt Type and Values Tab for the Columns to Print Option*



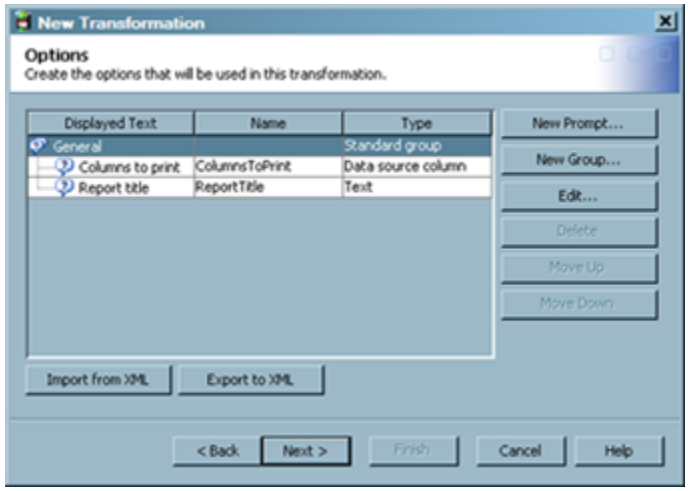
Each prompt window contains a **General** tab where you can enter general information about the option. Each prompt window also contains a **Prompt Type and Values** tab where you can select settings that are appropriate for each prompt type. For example, the second macro variable for the sample transformation, ReportType, requires an option that uses the text prompt type, as shown in the following display.

Display 13.10 *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.

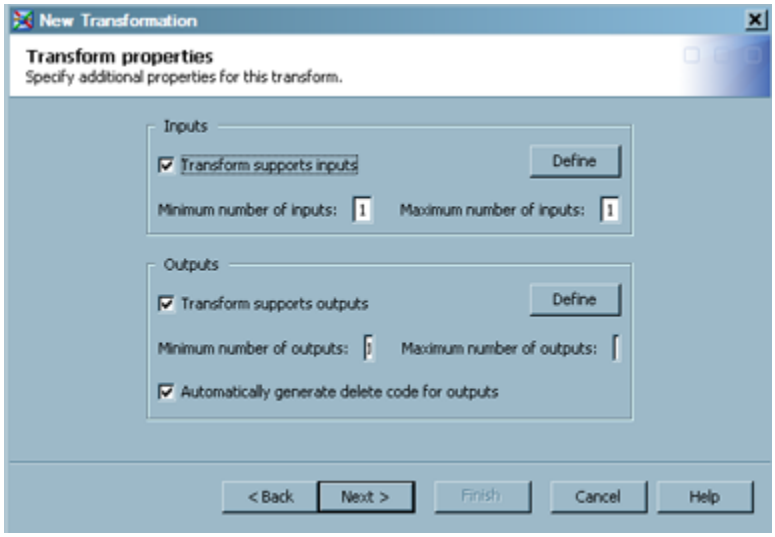
Display 13.11 *Completed Options Page*



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 13.12 *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.

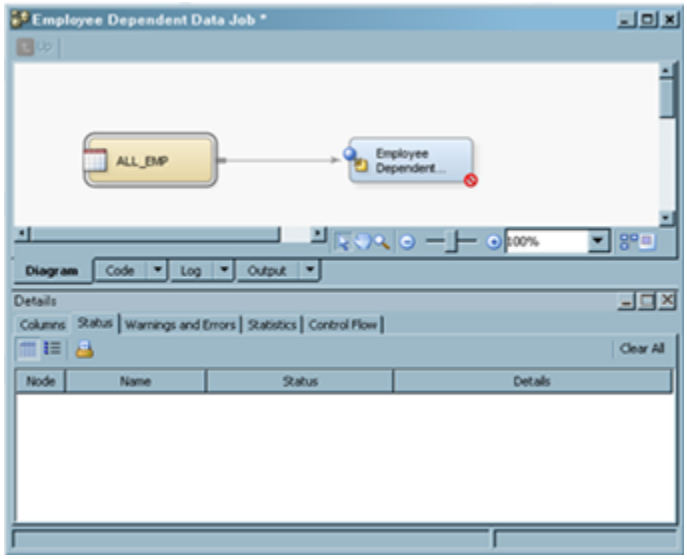
6. Click **Next** to access the Finish page. Verify that the metadata is correct, and then click **Finish**. Your transformation is created and saved.
7. Verify that the generated transformation is available in the destination folder.

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.

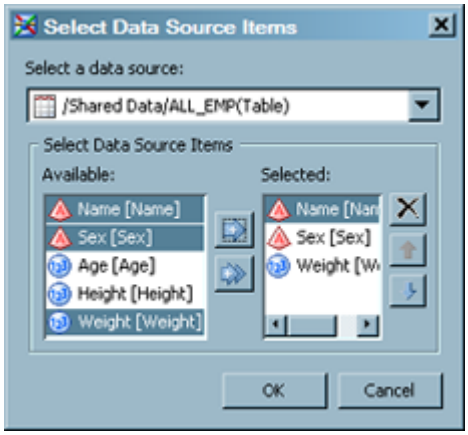
Display 13.13 *Generated Transformation in a Sample Job*



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.

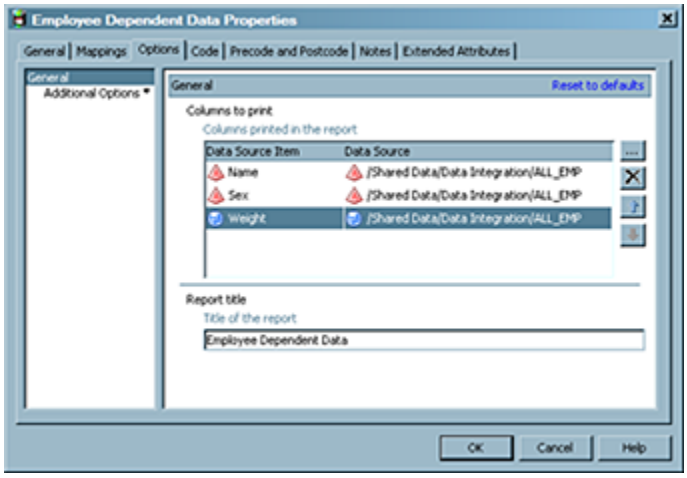
6. Open the **Options** tab in the properties window for the generated transformation. Enter appropriate values for each of the options that are created for the transformation. Then, set the properties for the first option in the transformation. The following display shows the Select Data Source Items window, which is used to select the columns that are printed in the report.

Display 13.14 *Sample Select Data Source Items Window*



The following display shows the completed **Options** tab.

Display 13.15 *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 13.16 *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
```

Updating a Generated Transformation

Problem

You want to update a generated transformation.

Solution

Each generated transformation has a unique ID. 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.

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 evaluate whether you want to make the change.

Perform the following tasks:

- "Identify a Generated Transformation" on page 284
- "Analyze the Impact of Generated Transformations" on page 284
- "Update Generated Transformations" on page 285

Tasks

Identify a Generated Transformation

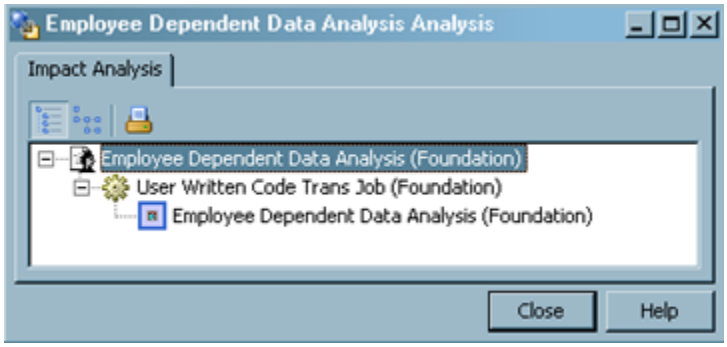
All transformations in the Transformation tree that have this icon () are generated transformations.

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 is displayed:

Display 13.17 *Impact Analysis on a Sample Generated Transformation*



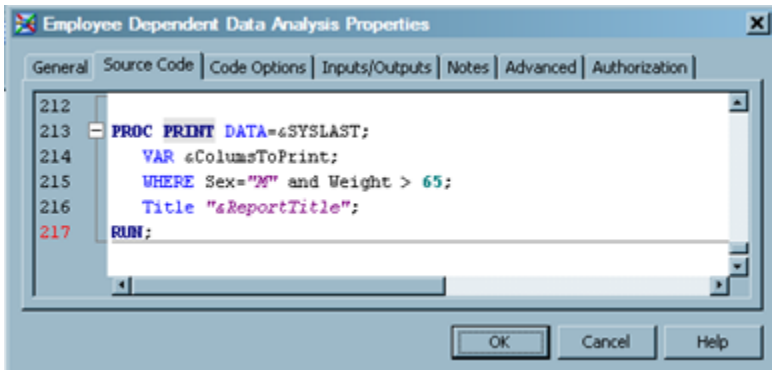
The selected generated transformation is named Employee Dependent Data. The Impact Analysis window shows that the selected transformation is used in a job. You can right-click the objects in the Report view to access their properties windows and view the jobs that contain them. For a data-flow view of 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.

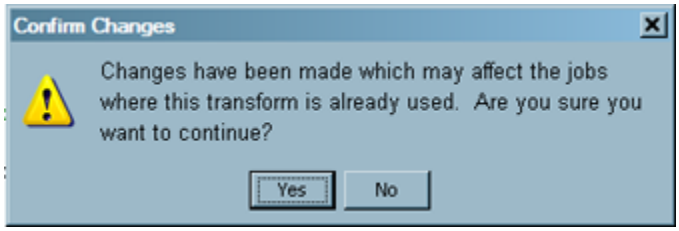
1. Access the properties window of the transformation that you want to update by double-clicking the transformation's name in the Transformations tree.
2. Click on a tab that you want to update.
3. Make any needed changes to the source code. Click **OK** to save these changes to the SAS code. The following display depicts an update to the source code of a sample transformation.

Display 13.18 *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 13.19 *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.

   Note: You can modify existing generated input and output macros. These macros use the following naming convention: *_input_xxxx* and *_output_xxxx*. The *xxxx* suffix is a descriptive keyword for the value of the macro.
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.

Converting a SAS Code File to a Job

Problem

You want to convert a SAS program file to a SAS Data Integration Studio job.

Solution

You can use the Import SAS Code wizard in SAS Data Integration Studio to convert a SAS program file and import it into SAS Data Integration Studio. The sources, targets, and procedures in the program file are rendered as metadata objects in a job.

The Import SAS Code wizard enables you to analyze your code and to automatically create SAS Data Integration Studio jobs. Behind the scenes, it calls the SCA (SAS Code Analyzer) procedure to analyze your SAS program. The SAS Code Analyzer captures information about input, output, and the use of macro symbols from a SAS job while it is running. The output generated is a file with your SAS program and any additional comments.

Note: The Import SAS Code wizard cannot parse all possible LIBNAME options for DBMS engines. If you import SAS code that includes LIBNAME options for DBMS engines, verify that the imported LIBNAME statement is correct, and that you can access the appropriate library. If some LIBNAME options are missing, configure them manually.

Two additional options are available as check boxes. You can select the **Expand macros** check box. This option creates a node for each step inside of your macros and provides additional detail about your job and how it works, including performance information about slow running steps, which steps use more memory or I/O, and CPU performance. You can also select the **Register work tables as physical tables** check box. This option registers all work tables as physical tables in a WORK library so that your imported SAS code uses temporary tables that are both the source and target of a step. You can also analyze your job to determine the type and number of steps in your job. This information is provided in a report that you can review prior to importing the job.

Perform the following tasks:

- "Review the SAS Program File" on page 289
- "Import the SAS Program File" on page 289

- "Open and Run the Job" on page 290
- "Review the Output" on page 291

Tasks

Review the SAS Program File

Review the SAS program file that you want to import, such as the following sample file:

```
libname ditest 'c:\\DISdata';

data temp.burgers;
 input where $ 1-18 food $ 19-34 calories fat $ sodium $ id $;
     cards;
 Burger King       cheeseburger    380  19g 780mg 1
 Hardees           cheeseburger    390  20g 990mg 10
 Jack In The Box   cheeseburger    320  15g 670mg 0
 McDonalds         cheeseburger    320  14g 750mg 35
 Wendys            cheeseburger    320  13g 770mg 20
 ;
run;

data temp.lesscalories;
   set temp.burgers;
   where calories < 390;
run;
```

Note: You can use comment tags to embed comments into the converted job, as follows:

- **ALTERNATE_NODE_NAME**: the node name
- **ALTERNATE_NODE_DESCRIPTION**: the node description
- **COMMENT**: tags that are grouped together into a private note attached to the node

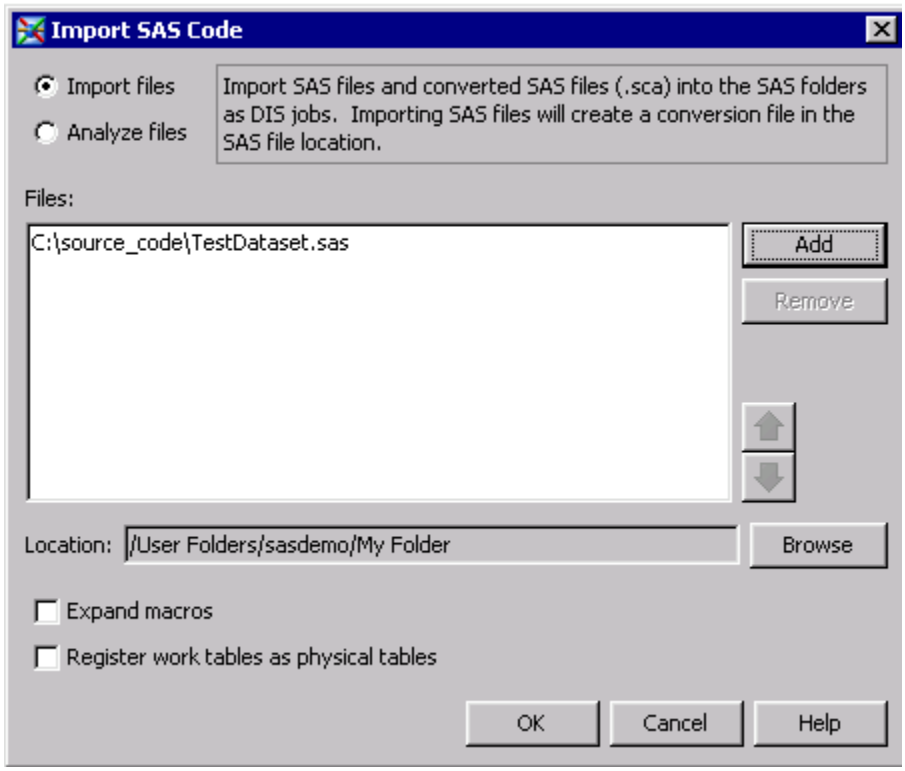
These tags should be placed after the code block for which they are intended.

Import the SAS Program File

Perform the following steps to import the program file:

1. Right-click the destination folder in SAS Data Integration Studio for the imported program file. Then, click **Import SAS Code** in the pop-up menu to access the Import SAS Code window. The following display shows the window for a sample program file.

Display 13.20 *Import SAS Code Window*



2. Click **Add** and select the SAS program file that contains the code that you need.
3. Click **OK** to run the wizard.

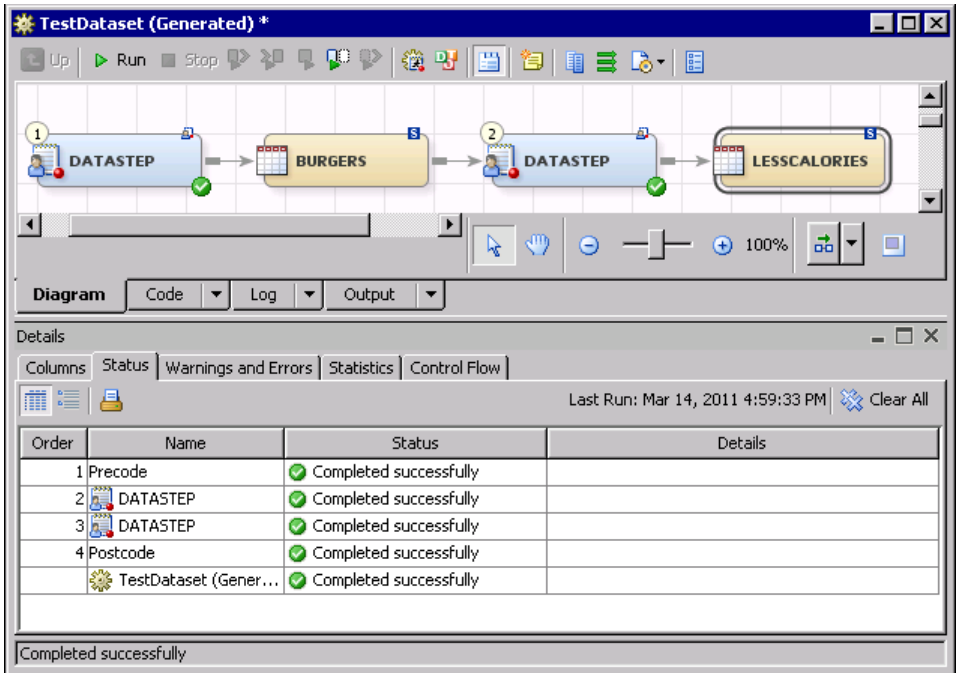
   Note: You can view the log file for the run. This log file is created whenever any action is taken. The log file will have a name that equals the program name.log. Therefore, the log in this example is named TestDataset.log.

Open and Run the Job

Perform the following steps to open and run the job:

1. Open the job that you imported and converted. The job will have the same name as the program file, with (Generated) appended. For example, the SAS program TestDataset.sas becomes the job that is identified as TestDataset (Generated).
2. Run the job. The following display shows a successfully completed sample job.

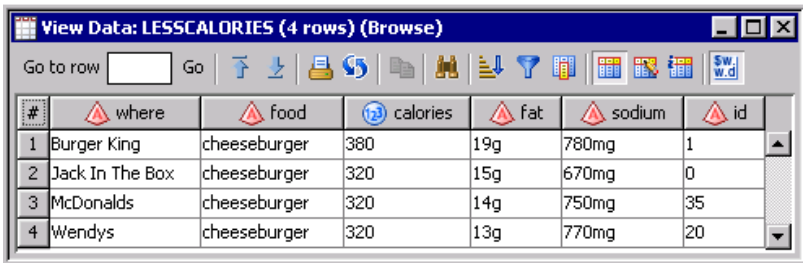
Display 13.21 *Sample Imported Job*



Review the Output

If the job completes without error, right-click the target table and click **Open**. The View Data window appears, as shown in the following example.

Display 13.22 *Sample Target Table Output*



#	where	food	calories	fat	sodium	id
1	Burger King	cheeseburger	380	19g	780mg	1
2	Jack In The Box	cheeseburger	320	15g	670mg	0
3	McDonalds	cheeseburger	320	14g	750mg	35
4	Wendys	cheeseburger	320	13g	770mg	20

Chapter 14

Optimizing Process Flows

About Process Flow Optimization **293**

Managing Process Data **294**
Problem 294
Solution 294
Tasks 294

Managing Columns **297**
Problem 297
Solution 297
Tasks 298

Streamlining Process Flow Components **299**
Problem 299
Solution 299
Tasks 299

Using Simple Debugging Techniques **301**
Problem 301
Solution 301
Tasks 301

Using SAS Logs **304**
Problem 304
Solution 304
Tasks 304

Reviewing Temporary Output Tables **306**
Problem 306
Solution 306
Tasks 307

Additional Performance Optimization Information **308**

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 294

- "Managing Columns" on page 297
- "Streamlining Process Flow Components" on page 299

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 301
- "Using SAS Logs" on page 304
- "Reviewing Temporary Output Tables" on page 306

See Also

"Additional Performance Optimization Information" on page 308

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 294
- "Delete Intermediate Files" on page 295
- "Cleanse and Validate Data" on page 297
- "Minimize Remote Data Access" on page 297

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 306. 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("sqlobs"), best.);
        do n = 1 to num_members;
            this_member = scan(work_members, n, ",");
            call symput("member"||trim(left(put(n,best.))),trim(this_member));
        end;
        call symput("num_members", trim(left(put(num_members,best.))));
    run;
    %if &num_members gt 0 %then %do;
        proc datasets library = work nolist;
            %do n=1 %to &num_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 268.

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 277.

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 298
- "Avoid Adding Unneeded Columns" on page 298
- "Aggregate Columns for Efficiency" on page 299
- "Match the Size of Column Variables to Data Length" on page 299

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 do not 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 185. For information about column propagation, see "Managing the Scope of Column Changes in Jobs" on page 189.

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 299
- "Use Transformations for Star Schemas and Lookups" on page 300
- "Use Surrogate Keys" on page 300

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, and 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 302.

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 522.

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 522.

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 301
- "Verify Output From a Transformation" on page 302
- "Limit Input to a Transformation" on page 302
- "Add Debugging Code to a Process Flow" on page 302
- "Set SAS Invocation Options on Jobs" on page 303
- "Set and Check Status Codes" on page 303

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 170 and "Diagnosing and Correcting an Unsuccessful Job" on page 175.

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 111.

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 268.

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. Note that these options do not work when the query generates a view.

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 14.1 *Methods for Adding Custom Debugging Code*

Method	Documentation
Replace the generated code for a transformation with user-written code.	"Replacing the Generated Code for a Job or Transformation" on page 287
Add the User-Written Code transformation to the process flow.	"Adding a User Written Code Transformation to a Job" on page 274
Add a generated transformation to the process flow.	"Creating and Using a Generated Transformation" on page 277
Add a return code to the process flow.	"Set and Check Status Codes" on page 303

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 269.

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 268.

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 269.

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 217.

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 301.

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 304
- "Capture Additional SAS Options in the SAS Log" on page 305
- "View or Hide SAS Logs" on page 305
- "Redirect Large SAS Logs to a File" on page 306

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 `Scalability and Performance 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 **`Scalability and Performance 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. If 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 303.

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 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 268. 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).

Note: You can also redirect temporary output tables to an alternative location. For details, see "Redirecting Temporary Output Tables" on page 196.

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 307
- "View Temporary Output Tables" on page 307
- "Redirect Temporary Output Tables" on page 307
- "Add the List Data Transformation to a Process Flow" on page 308
- "Add a User-Written Code Transformation to the Process Flow " on page 308

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 303.

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\SASHome\SASFoundation\
[release_number]\sas.exe"
-work "C:\Users\sasapb\My Documents\My SAS Files\
[release_number]\My SAS WorkFolder"
-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 111.

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 only print 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:\Users\ramich\My Documents\My SAS Files\[release_number]";
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 274.

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 479. For specific information about the

performance of the Table Loader transformation, see "Selecting a Load Technique in the Table Loader" on page 429 and "Removing Non-Essential Indexes and Constraints during a Load" on page 432.

You can also access a library of SAS Technical Papers that cover a variety of performance-related topics. You can find these papers at **`SAS Technical Papers`**.

Chapter 15

Working with Impact Analysis and Data Lineage

Impact Analysis and Data Lineage **311**
Overview 311
Using Impact Analysis in SAS Data Integration Studio 312
Using SAS Lineage 313

Performing an Impact Analysis **313**
Problem 313
Solution 313
Tasks 313

Performing Impact Analysis on a Generated Transformation **316**
Problem 316
Solution 317
Tasks 317

Performing Reverse Impact Analysis **317**
Problem 317
Solution 318
Tasks 318

Using SAS Lineage **318**
Overview 318
Prerequisites 318
Use the Analyze in Web Viewer Option for Tables and External Files 319

Impact Analysis and Data Lineage

Overview

Impact analysis identifies the potential consequences of a change, such as adding a new column to a table that is used in a job. SAS Data Integration Studio enables you to perform impact analysis on columns, tables, external files, information maps, reports, stored processes, SAS Enterprise Guide projects and associated objects, and the levels and measures in OLAP cubes. You can also generate impact analyses for generated transformations.

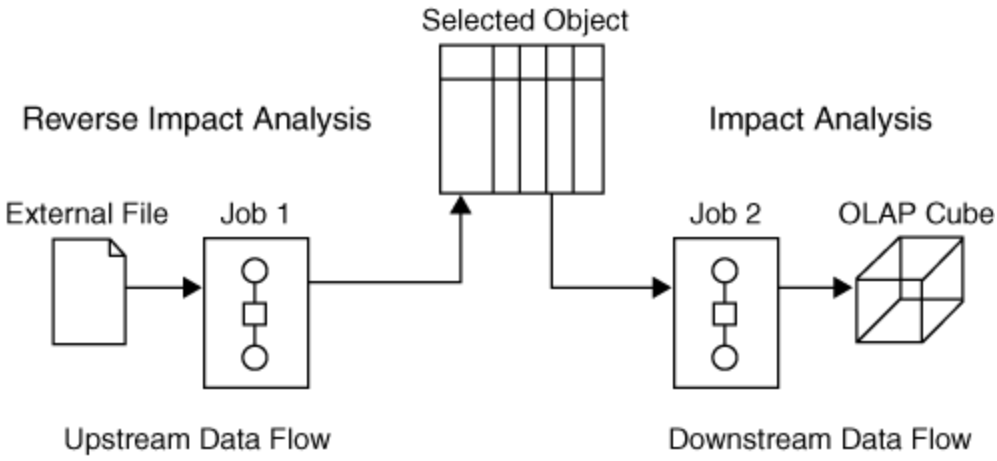
Data lineage analysis identifies where data originates and how it is used. You can use a web client, SAS Lineage, to display lineage information for tables and external files that are used in SAS Data Integration Studio jobs.

Using Impact Analysis in SAS Data Integration Studio

In SAS Data Integration Studio, the impact analysis features identify the tables, columns, jobs, and transformations that are affected by a change to a selected table or column. The reverse impact analysis features identify 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 15.1 *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. Analysis extends into cubes. You can generate impact and reverse impact analyses for most types of data objects, including columns, tables, external files, information maps, reports, stored processes, 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 316.

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*.

Using SAS Lineage

If the SAS Lineage web client is available on your network, you can use it to display lineage for tables and external files that are used in SAS Data Integration Studio.For information about this web client, see "Use the Analyze in Web Viewer Option for Tables and External Files" on page 319.

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.

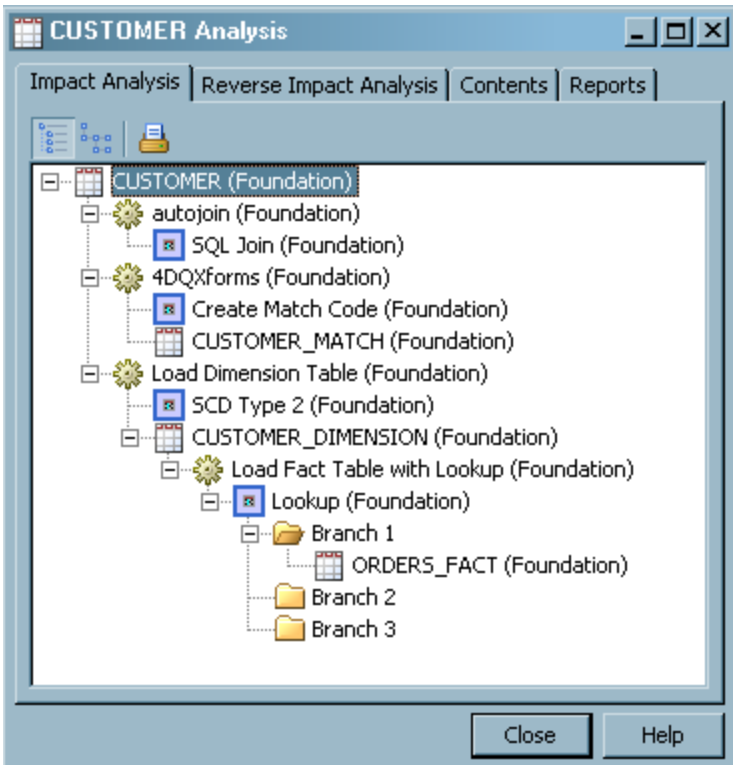
Tasks

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.

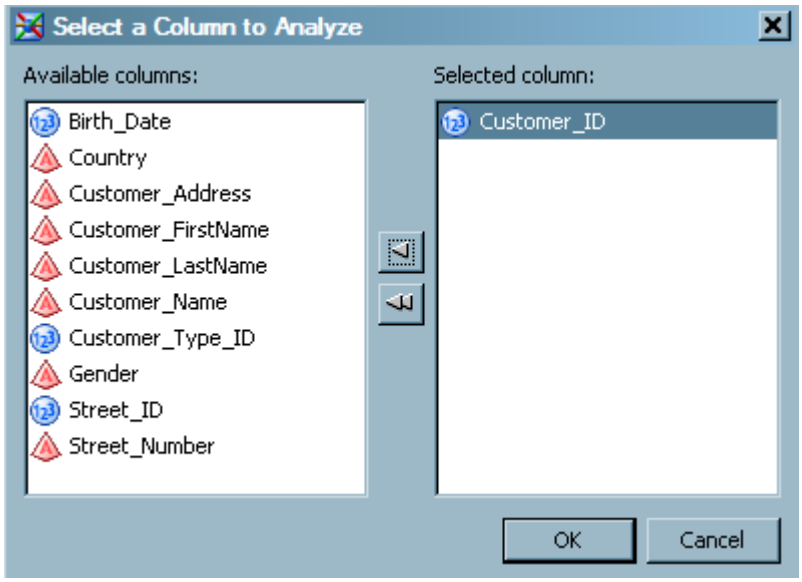
Display 15.1 *Impact Analysis Tab*



Perform the following steps to trace the impact of the metadata for a table column:

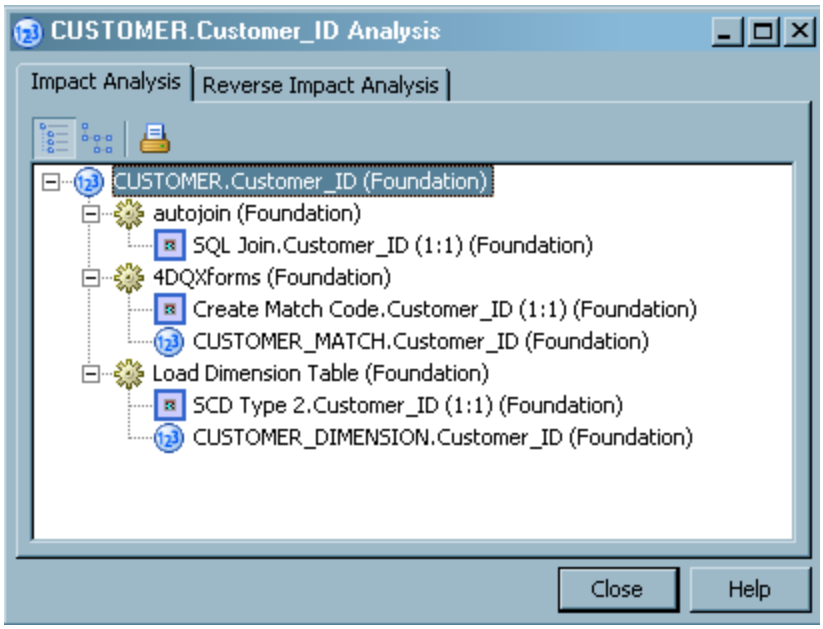
1. In a tree view or in the context of a process flow, right-click on the metadata object for the table that contains the column to be analyzed. Select **Analyze**.
2. In the Analyze window, right-click on the metadata object for the table, and then select **Analyze Columns**.
3. Select the column that you want from the **Available columns** pane. Use the arrow key to move it to the **Selected column** pane.

Display 15.2 *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 15.3 *Analysis Results*

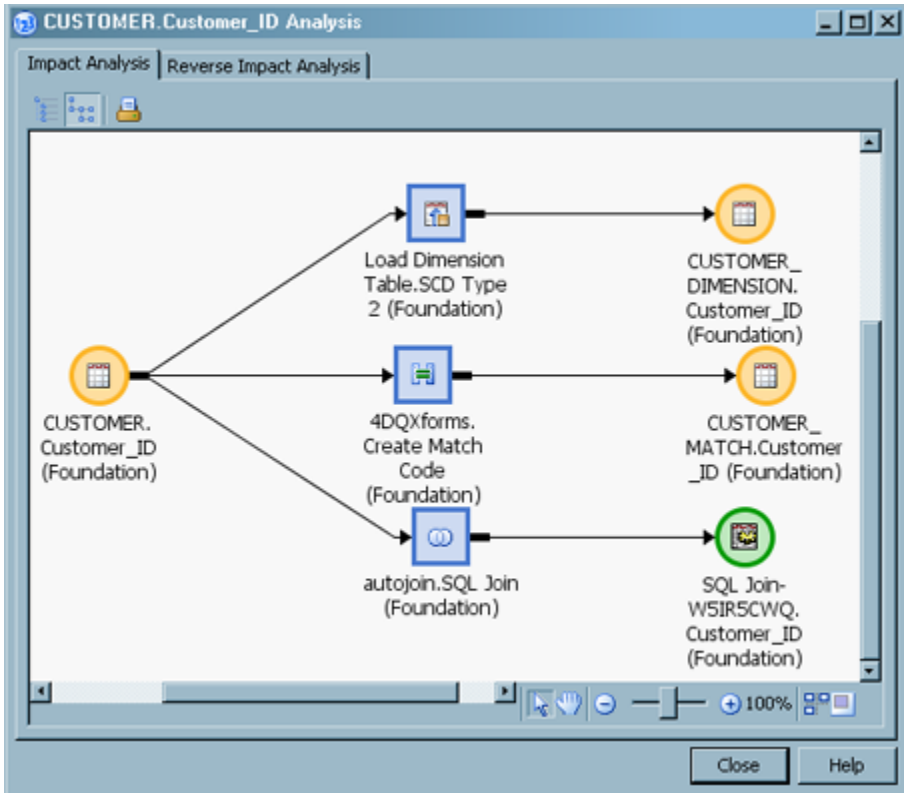


The **Tree View** window uses a hierarchical list to illustrate the impact of the selected object (Customer_ID column) on later objects in a process flow. In the previous display, the tab contains three jobs. In this example, the third job contains the following objects:

- **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 15.4 *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 277.

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

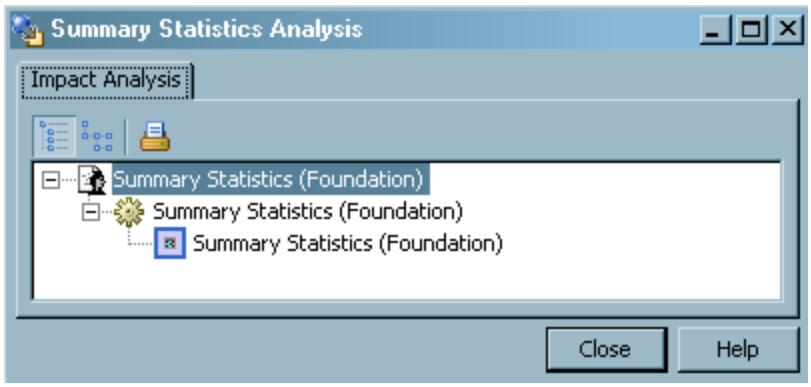
Perform Impact Analysis on a Generated Transformation

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, and then select **Actions** from the menu bar, and then select **Analyze**. The following display shows the tree view of the analysis.

Display 15.5 *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 313.

Using SAS Lineage

Overview

If the SAS Lineage web client is available on your network, you can use it to display lineage for tables and external files that are used in SAS Data Integration Studio. For example, you could use SAS Lineage to display lineage for a table or external file that is used in a SAS Data Integration Studio job. You could also right-click a table or external file in SAS Data Integration Studio and select the **Analyze in Web Viewer** option. This option enables you to log on to SAS Lineage and view lineage information for that table or file. For more information about SAS Lineage, see **`http://support.sas.com/documentation/onlinedoc/dmlin`**.

Prerequisites

You must meet the following prerequisites in order to use the **Analyze in Web Viewer** option.

- The URL for SAS Lineage must be specified in the **Impact Analysis web viewer URL** field on the **Tools** ⇨ **Options** ⇨ **General tab** in SAS Data Integration Studio. The URL must be in the following format: **`http://server-name:port-number/SASLineage`**. The port-number value can be omitted only if the mid-tier server is configured with the default port of 80. This URL supports the **Analyze in Web Viewer** option for tables and external files.
- Both SAS Data Integration Studio and SAS Lineage must be connected to the same SAS Metadata Server.
- You must have a login ID and password for a SAS Lineage user as defined in the SAS Metadata Server.

- The computer on which you are running SAS Data Integration Studio must have Adobe Flash Player version 11.1.0 or later.

Note: Microsoft Internet Explorer 9 cannot be used to access SAS Lineage.

Use the Analyze in Web Viewer Option for Tables and External Files

Right-click a table or an external file in the Folders tree or the Inventory tree, and then select **Analyze in Web Viewer**. A web browser opens. You are prompted to log on to SAS Lineage. Provide the login ID and password for a SAS Lineage user. You can then view lineage information for the selected table or file.

Chapter 16

Working with Reports

About Metadata Reports **322**

Opening the Reports Window **322**
Problem 322
Solution 322
Tasks 322

Selecting the Reports Perspective **323**
Problem 323
Solution 323
Tasks 324

Customizing the Tables Report **324**
Problem 324
Solution 324
Tasks 325

Customizing the Job Documentation Report **325**
Problem 325
Solution 325
Tasks 326

Running and Saving a Report **326**
Problem 326
Solution 326
Tasks 327

Saving a Report As a Document Object **328**
Problem 328
Solution 328
Tasks 328

Viewing a Report **329**
Opening a Report 329
Contents of a Tables Report 329
Contents of a Job Report 330
Contents of Your Own Report 331

Creating Your Own Report **331**
Problem 331
Solution 331
Tasks 331

About Metadata 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 331.

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 328.

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.

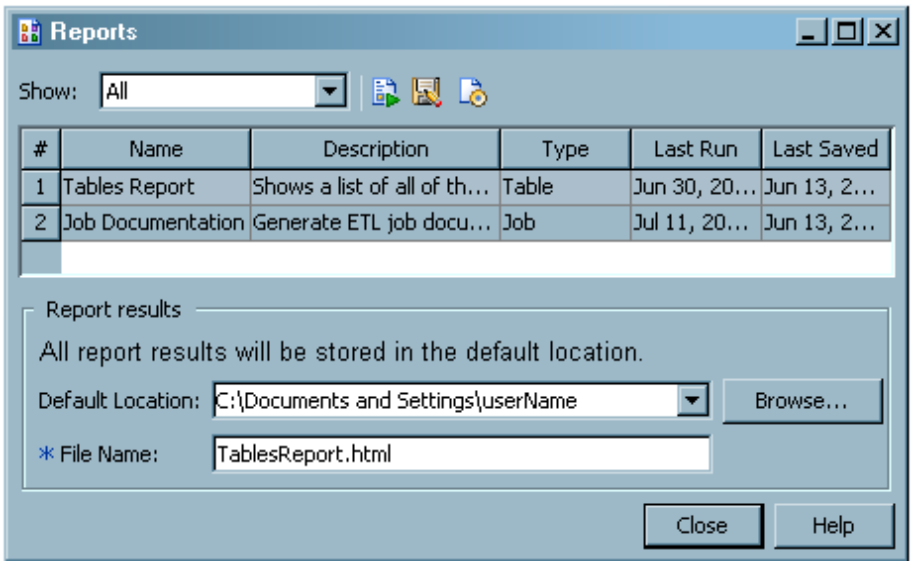
Tasks

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 16.1 *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 16.1 *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.

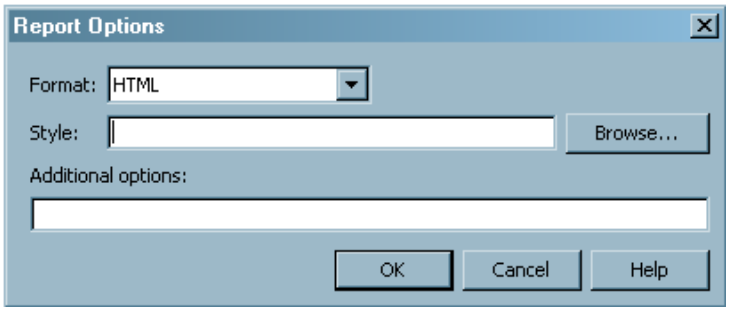
Tasks

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 16.2 *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.

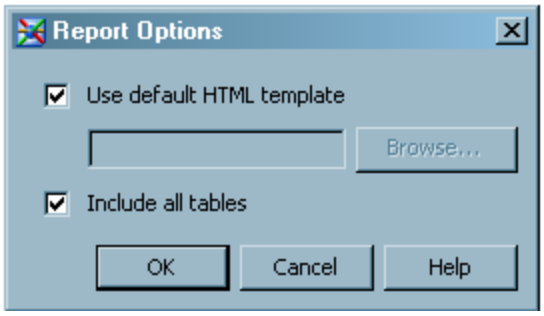
Tasks

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 16.3 *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.

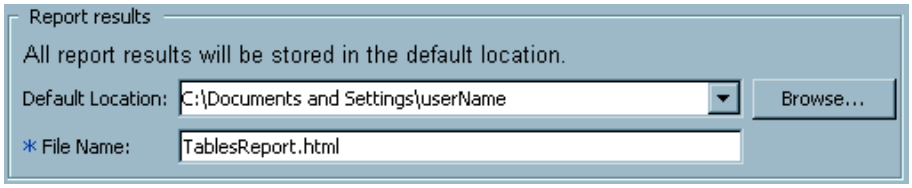
Tasks

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.

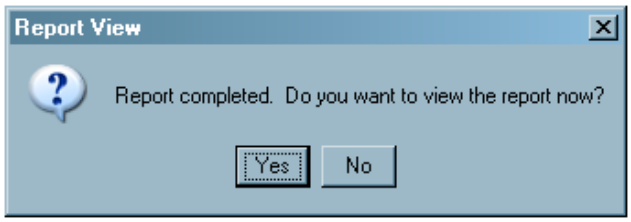
Display 16.4 *Report Results Pane*



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 16.5 *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 329.

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.

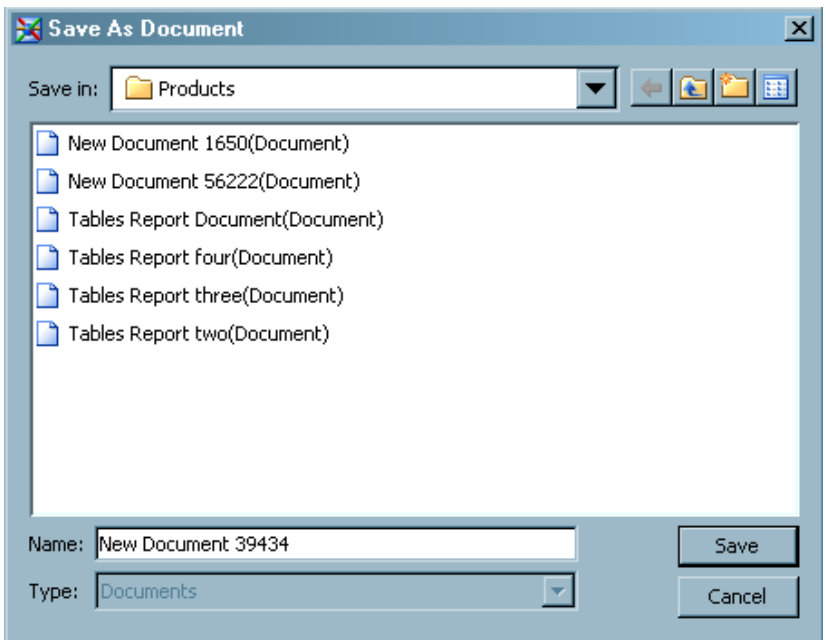
Tasks

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 16.6 *Save As Document Dialog Box*



4. 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 16.7 *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:

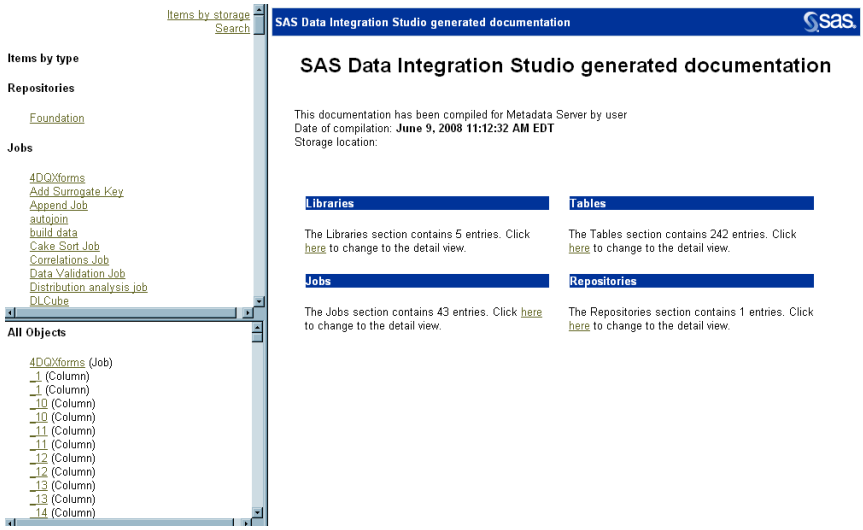
- an observation number for each table
- the name of a table
- a description of the table
- the date that the table was created
- the date that the table was last modified
- the owner of the table

- 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 corner of the job report. The third window is an Objects window, and it is located in the lower left corner of the job report.

Display 16.8 *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 331.

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 763. For explanations of the methods in the report plug-in interface, see "Reporting Interface Methods" on page 769.

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\SASHome\SASDataIntegrationStudio\<version>\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 that 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\SASHome\SASDataIntegrationStudio\<version>\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.

Chapter 17

Working with Data Management Platform

Integration with DataFlux Data Management Platform **334**
Overview 334
Transformations in the Data Quality Folder 335

General Prerequisites for Data Quality Transformations **336**
DataFlux Software 336
Global Options on the Data Quality Tab 336

Prerequisites for Running a DataFlux Job or Profile in a SAS Data Integration Studio Job **338**
Overview 338
Verify How Users Are Authenticated on the DataFlux Data Management Server 338
Deploy the DataFlux Job, Service, or Profile to a DataFlux Data Management Server 339
Grant Privileges on the DataFlux Data Management Server 339
Next Tasks 341

Analyzing the Quality of Data Sources **341**

Standardizing Values with a Standardization Scheme **341**
Problem 341
Solution 342
Tasks 342

Standardizing Values with a Definition **346**
Problem 346
Solution 346

Using Match Codes to Improve Record Matching **347**
Problem 347
Solution 347
Tasks 347
Usage Notes 351

Using a DataFlux Data Service in a SAS Data Integration Studio Job **351**
Problem 351
Solution 352
Tasks 352

Using a DataFlux Job or Profile in a SAS Data Integration Studio Job **355**
Problem 355
Solution 355
Tasks 356

Integration with DataFlux Data Management Platform

Overview

SAS enterprise software offerings such as SAS Data Management and SAS Data Integration Server include SAS Data Integration Studio, SAS Data Quality Server, and the DataFlux Data Management Platform. The SAS Data Quality Server consists of a Quality Knowledge Base (QKB) and SAS language elements. The DataFlux Data Management Platform provides a single environment for managing data quality, data discovery, and master data management (MDM).

Many of the features in SAS Data Quality Server and the DataFlux Data Management Platform can be used in SAS Data Integration Studio jobs. For example, you can use DataFlux standardization schemes and definitions in SAS Data Integration Studio jobs. You can also execute DataFlux jobs, profiles, and services from SAS Data Integration Studio.

If your site has licensed the appropriate DataFlux software, you can take advantage of the following components:

DataFlux Web Studio
: a web-based application with separately licensed modules that enable you to perform data management tasks.

DataFlux Data Management Studio
: a desktop client that combines data quality and data discovery features. You can use this client to create jobs, profiles, standardization schemes, and other resources that can be included in SAS Data Integration Studio jobs.

DataFlux Data Management Server
: provides a scalable server environment for large DataFlux Data Management Studio jobs. Jobs can be uploaded from DataFlux Data Management Studio to a DataFlux Data Management Server, where the jobs are executed. SAS Data Integration Studio can execute DataFlux jobs on this server.

data job
: a DataFlux job that specifies a set of data cleansing and enrichment operations that flow from source to target.

data service
: a data job that has been configured as a real-time service and deployed to a DataFlux Data Management Server.

process job
: a DataFlux job that combines data processing with conditional processing. The process flow in the job supports logical decisions, looping, events, and other features that are not available in a data job flow.

profile
: a job that executes one or more data profiling operations and displays a report based on the result of these operations. Data profiling encompasses discovery and audit activities that help you assess the composition, organization, and quality of databases.

DataFlux Quality Knowledge Base (QKB)
: a collection of files and reference sources that allow Blue Fusion and consequently all DataFlux software to do parsing, standardization, analysis, matching, and other processes. A QKB includes locales, standardization schemes, and other resources.

locale
: a collection of data types and definitions that are pertinent to a particular language or language convention. A locale for **English – UK**, for example, has an address parse definition different than an **English – US** parse definition. The address format is significantly different even though the language is similar.

standardization scheme
: a file that contains pairs of data values and standardized values. Schemes are used to standardize columns by providing a set of acceptable values.

standardization definition
: a set of logic used to standardize an element within a string. For example, a definition could be used to expand all instances of "Univ." to "University" without having to specify every literal instance such as "Univ. Arizona" and "Oxford Unv." in a scheme.

Transformations in the Data Quality Folder

The **Transformations** tree in SAS Data Integration Studio includes a **Data Quality** folder. This folder includes the following transformations. In general, you could use Apply Lookup Standardization, Create Match Code, and Standardize with Definition for data cleansing operations. You could use DataFlux Batch Job and DataFlux Data Service to perform tasks that are a specialty of DataFlux software, such as profiling, monitoring, or address verification.

Apply Lookup Standardization
: enables you to select and apply DataFlux schemes that standardize the format, casing, and spelling of character columns in a source table.

Create Match Code
: enables you to analyze source data and generate match codes based on common information shared by clusters of records. Comparing match codes instead of actual data enables you to identify records that are in fact the same entity, despite minor variations in the data.

DataFlux Batch Job
: enables you to select and execute a DataFlux job that is stored on a DataFlux Data Management Server. You can execute DataFlux Data Management Studio data jobs, process jobs, and profiles. You can also execute Architect jobs that were created with DataFlux® dfPower® Studio.

DataFlux Data Service
: enables you to select and execute a data job that has been configured as a real-time service and deployed to a DataFlux Data Management Server.

Standardize with Definition
: enables you to select and apply DataFlux standardization definitions to elements within a text string. For example, you might want to change all instances of "Mister" to "Mr." but only when "Mister" is used as a salutation. Requires SAS Data Quality Server.

If you export and import SAS Data Management Studio jobs that contain DataFlux Batch Job transformations or DataFlux Data Service transformations, then there are

some special considerations. For more information, see "Preparing to Import or Export SAS Package Metadata" on page 61.

General Prerequisites for Data Quality Transformations

DataFlux Software

Transformations in the **Data Quality** folder require either SAS Data Quality Server or DataFlux Data Management Platform. These products are included with SAS enterprise software offerings such as SAS Data Management and SAS Data Integration Server. For more information about configuring DataFlux software for use with SAS Data Integration Studio, see the SAS Data Integration Studio chapter of the *SAS Intelligence Platform: Desktop Administration Guide*.

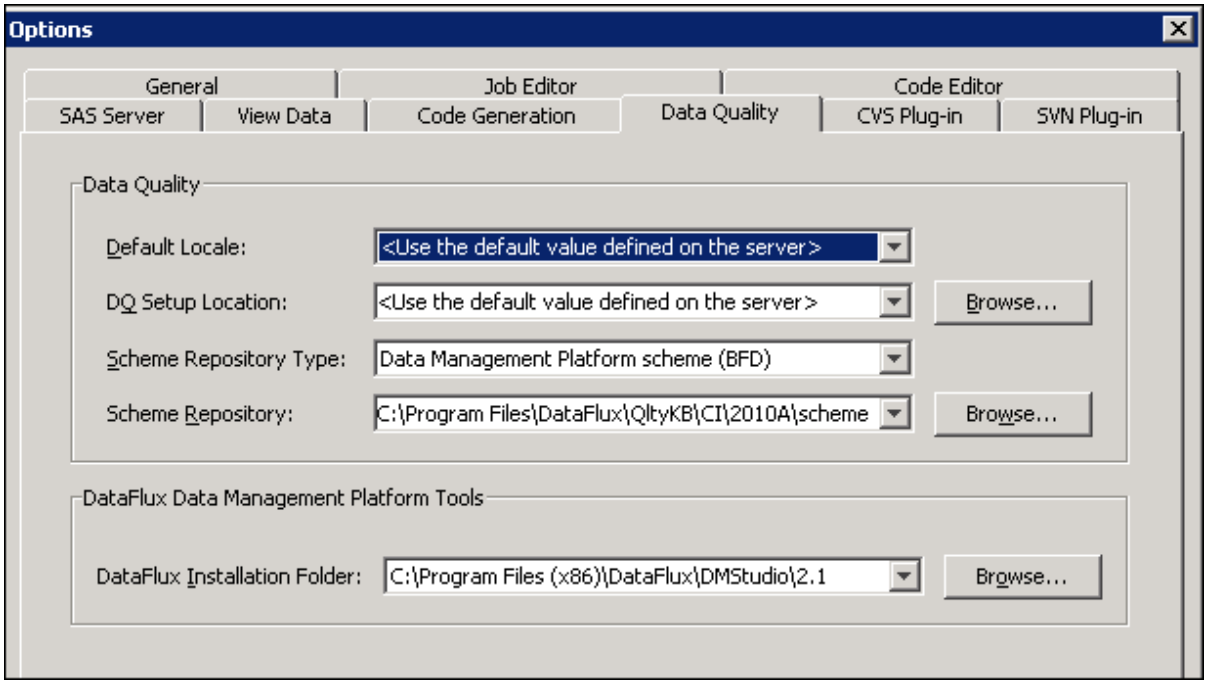
Review the DataFlux components that are described in "Overview" on page 334. Identify the components that you want to use in SAS Data Integration Studio, and then configure or create these components. For example, if you want to use a DataFlux standardization scheme in a SAS Data Integration Studio job, you must create the scheme in DataFlux software. For more information, see the DataFlux documentation such as the *DataFlux Data Management Studio User's Guide*.

Note: With the exception of the DataFlux Batch Job transformation, which can be used to execute DataFlux dfPower Studio Architect jobs that do not contain macros, the current version of SAS Data Integration Studio works only with the DataFlux Data Management Platform. Other DataFlux dfPower Studio objects must be migrated to the DataFlux Data Management Platform. For more information, see the *DataFlux Migration Guide*.

Global Options on the Data Quality Tab

After the DataFlux resources have been configured or created, you can specify some global data quality options in SAS Data Integration Studio. Select**Tools** ⇨ **Options** to display the Options window, and then click the **Data Quality** tab. The next figure shows some typical values in this tab.

Display 17.1 *Data Quality Tab*



Paths specified in the **Data Quality** group box are relative to the current SAS Application Server. The group box contains the following items:

Default Locale
specifies the locale that is referenced by SAS data quality jobs when a different locale is not specified in those jobs. The default value is **Use the value defined on the server**. The default uses the value of the SAS system option DQLOCALE, which is set on the SAS Application Server that executes SAS data quality jobs.

In a standard deployment, the SAS Application Server is not configured to use any specific locale. There are three main ways to set the locale. You can configure the DQLOCALE option on the SAS Application Server that executes SAS data quality jobs. You can select a locale in the **Default Locale** field above. Also, you can select a locale for an individual data quality transformation in a SAS Data Integration Studio job.

DQ Setup Location
specifies the location of a DataFlux Quality Knowledge Base (QKB). In a standard deployment, the SAS Application Server is configured to use the sample QKB that is provided by SAS Data Quality Server. The sample QKB is typically located at the following path: *C:\Program Files\SASHome\SASFoundation\[release_number]\dquality\sasmisc\QltyKB\sample*

There are two main ways to set the QKB. You can configure the DQSETUPLOC option on the SAS Application Server that executes SAS data quality jobs. You can also select a QKB in the **DQ Setup Location** field above.

If you change the global DQ Setup Location, you have the option to apply the new location to data quality transformations in existing jobs. To apply the global DQ Setup Location to a transformation, click the **Reset DQ Setup Location** button in the appropriate tab, such as the **Standardization** tab for the Apply Lookup Standardization transformation. The following data quality transformations support this option: Apply Lookup Standardization transformations, Standardize with Definition transformations, and Create Match Codes transformations.

Scheme Repository Type
specifies that the scheme data sets in the specified scheme repository are stored in SAS format (option value **NOBFD**) or in DataFlux format (option value **BFD**, the default). The Apply Lookup Standardization transformation uses schemes to standardize data.

Scheme Repository
specifies the location of the scheme data sets that are used by the Apply Lookup Standardization transformation. To display scheme filenames in the transformation, specify:

QKB-root/scheme

To display scheme descriptions in the transformation, specify:

QKB-root

QKB-root is the directory that was specified when the Quality Knowledge Base was installed. *QKB-root* contains approximately nine subdirectories, with names such as regex, locale, and scheme.

Paths that are specified in the **DataFlux Data Management Platform Tools** group box are relative to the SAS Data Integration Studio application. This group box contains the following item:

DataFlux Installation Folder
specifies the folder where DataFlux Data Management Studio is installed. Under the 64-bit version of Windows, the default path is **C:\Program Files (x86)\DataFlux\DMStudio*instance_name***. Use the keyboard, drop-down list, or the **Browse** button to specify a different installation folder.

If you specify the path to DataFlux Data Management Studio and click **OK** to save your changes, the next time you start SAS Data Integration Studio, you can run DataFlux Data Management Studio by selecting **Tools** ⇨ **DataFlux Data Management Platform Tools** ⇨ **Data Management Studio**.

Prerequisites for Running a DataFlux Job or Profile in a SAS Data Integration Studio Job

Overview

These additional prerequisites apply if you want to incorporate a DataFlux data job, process job, data service, or profile into the flow for a SAS Data Integration Studio job. The job, service, or profile must be deployed to a DataFlux Data Management Server. It is assumed that this server is secured with a SAS Metadata Server, as described in the next section.

Verify How Users Are Authenticated on the DataFlux Data Management Server

SAS Data Integration Studio can access DataFlux jobs, services, or profiles if they have been deployed to a DataFlux Data Management Server. In production environments, this server is usually secured. It can be secured with either a DataFlux Authentication Server or a SAS Metadata Server. SAS offerings that include SAS Data Integration Studio 4.8 and later typically use the SAS Metadata Server to authenticate users on a DataFlux Data

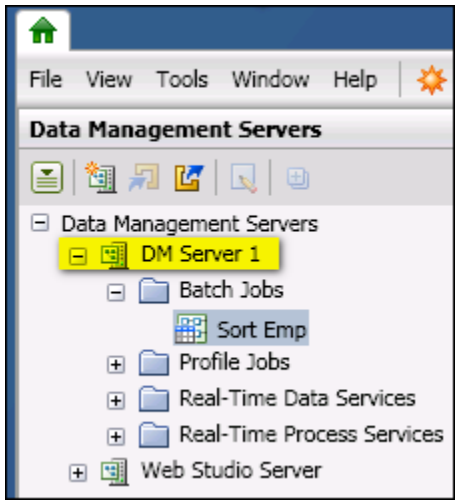
Management Server. The administrator who maintains your data management environment should know which authentication method is being used to secure the server where the DataFlux jobs, services, or profiles have been deployed.

If authentication is handled by a SAS Metadata Server, then follow the steps in the next section. If authentication is handled by a DataFlux Authentication Server, then see "Prerequisites for Running a Job When a DataFlux Server Is Used for Authentication" on page 664.

Deploy the DataFlux Job, Service, or Profile to a DataFlux Data Management Server

A DataFlux Data Management Studio user deploys jobs, services, or profiles to a DataFlux Data Management Server. He or she should ensure that the objects can be executed on the server and that they deliver the expected results. The next display shows a job, Sort Emp, that has been deployed to a server called DM Server 1.

Display 17.2 *Data Job Deployed to a DataFlux Data Management Server*



For information about deploying jobs, services, and profiles, see the chapters for data jobs, process jobs, and profiles in the *DataFlux Data Management Studio User's Guide*.

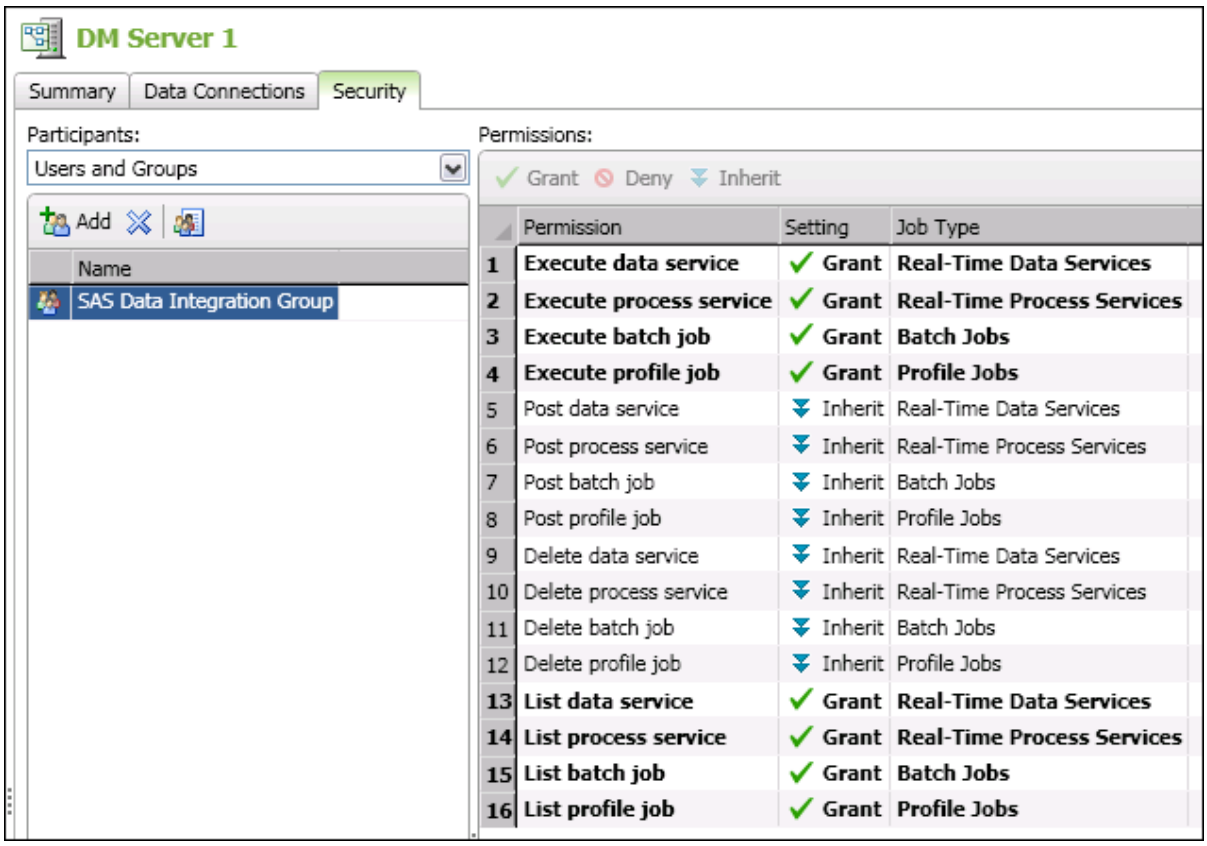
Grant Privileges on the DataFlux Data Management Server

This task is performed on the Data Management Servers riser in DataFlux Data Management Studio. It is performed by the administrator for the DataFlux Data Management Server where jobs, services, or profiles have been deployed. This administrator grants users or groups the general permission to list and execute deployed objects on the data management server. He or she also grants the appropriate user or group access to these specific objects.

Note: Both the List permission and the Execute permission must be granted to the SAS Data Integration Studio users or groups who execute jobs, services, or profiles on a DataFlux Data Management Server.

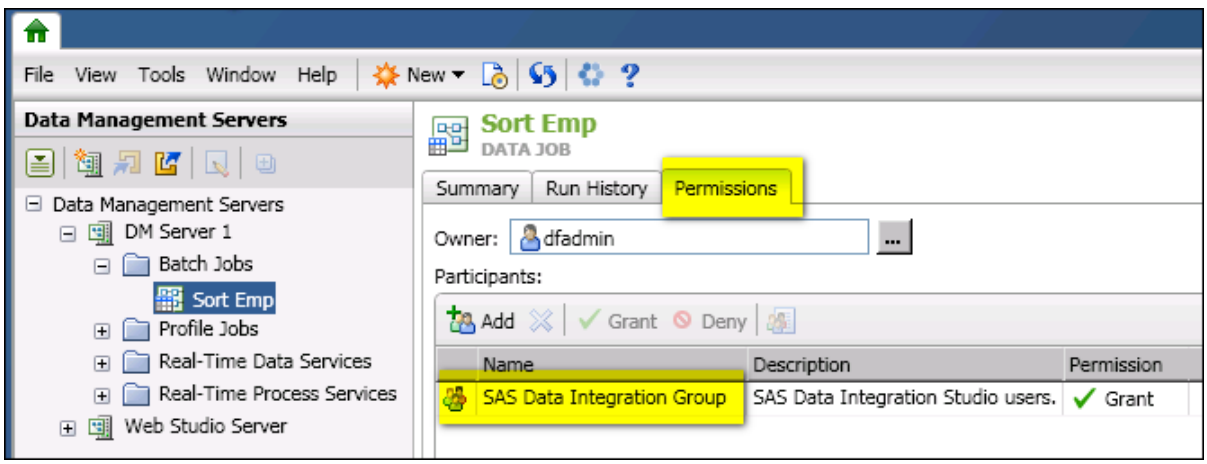
For example, you can grant List and Execute permissions to the SAS Data Integration Group, as shown in the next display.

Display 17.3 *Granting List and Execute Permissions for Jobs, Services, and Profiles*



Next, identify the individual jobs, services, or profiles on the server that SAS Data Integration Studio users should be able to execute. Grant the appropriate user or group access to these specific objects. For example, you can grant permissions so that the SAS Data Integration Group can access Sort Emp, as shown in the next display.

Display 17.4 *Granting Permission to Access Individual Jobs, Services, and Profiles*



For more information about these tasks, see the "Security Administration" chapter in the *DataFlux Data Management Server Administrator's Guide*.

Next Tasks

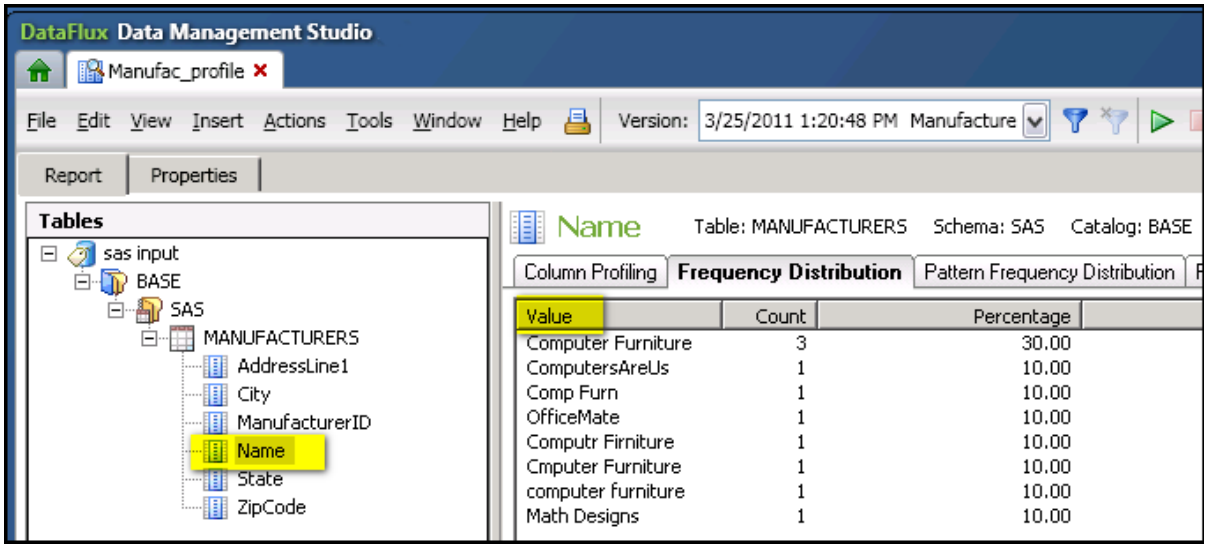
After you have met the prerequisites above, you can do the following tasks:

- "Using a DataFlux Job or Profile in a SAS Data Integration Studio Job" on page 355
- "Using a DataFlux Data Service in a SAS Data Integration Studio Job" on page 351

Analyzing the Quality of Data Sources

You can use DataFlux Data Management Studio to analyze the quality of the data sources that are used in SAS Data Integration Studio jobs. For example, in DataFlux Data Management Studio, you could create a profile that analyzes the data in a table called **MANUFACTURERS**. The profile could reveal problems with the data, such as a column that contains misspellings of company names. The profile in the next figure shows a number of misspellings for the **Computer Furniture** company.

Display 17.5 *Profile Shows Data Errors in the Name Column of the MANUFACTURERS Table*



You can use the results of data quality analysis to create SAS Data Integration Studio jobs that will correct problems with the data. For more information about the data quality features in DataFlux Data Management Studio, see the *DataFlux Data Management Studio User's Guide*.

Standardizing Values with a Standardization Scheme

Problem

You want to standardize the values in one or more character columns in a source table.

Solution

Get detailed information about the incorrect values. Use that information to create a standardization scheme that maps incorrect values to the correct values. Use the scheme in a SAS Data Integration Studio job to standardize the data in the problematic columns. Perform the following tasks:

- "Identify Incorrect Values" on page 342
- "Create a Standardization Scheme" on page 342
- "Verify Prerequisites" on page 343
- "Create and Populate the Job" on page 343
- "Configure the Apply Lookup Standardization Transformation" on page 344
- "Run the Job and View the Output" on page 345

Tasks

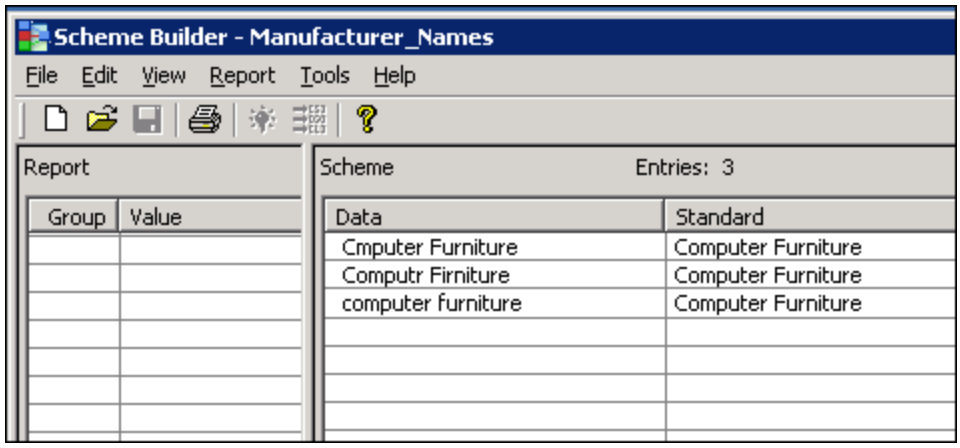
Identify Incorrect Values

You can use DataFlux Data Management Studio to get detailed information about problems with source data. For example, you could identify all of the incorrect spellings of a company name in a table column. Detailed information about incorrect values can help you create an effective standardization scheme. For more information, see "Analyzing the Quality of Data Sources" on page 341.

Create a Standardization Scheme

Use DataFlux Data Management Studio or the DQMATCH procedure in SAS Data Quality Server to create a standardization scheme that maps incorrect values to the correct ones. The next figure shows a scheme in DataFlux Data Management Studio that can be used to correct misspellings for the **Computer Furniture** company.

Display 17.6 *Standardization Scheme for a Company Name*



For more information about creating standardization schemes, see the scheme topics in the "Customize" chapter of the *DataFlux Data Management Studio User's Guide*. Alternatively, see the documentation for the DQMATCH procedure in the documentation for SAS Data Quality Server.

Verify Prerequisites

The Apply Lookup Standardization transformation that is used in this topic requires the "General Prerequisites for Data Quality Transformations". In SAS Data Integration Studio, verify that the appropriate **Scheme Repository Type** and **Scheme Repository** are selected, as described in "Global Options on the Data Quality Tab" on page 336. The scheme repository must contain the standardization schemes that you want to use in SAS Data Integration Studio.

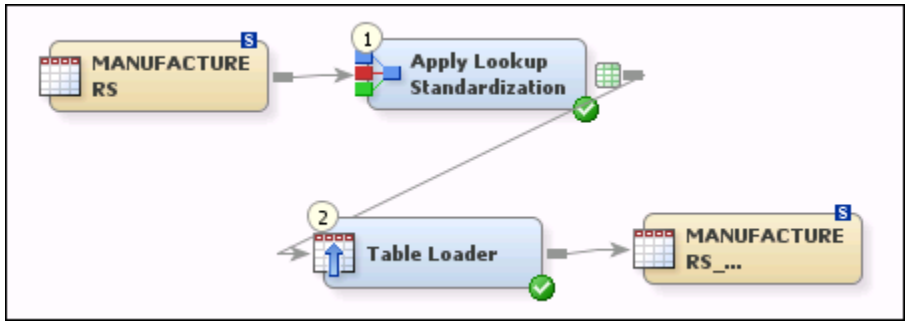
Note: On the **Data Quality** tab, if you change an existing value in the fields **Scheme Repository Type** or **Scheme Repository**, then you must replace any instances of the Apply Lookup Standardization transformation in any existing jobs that you intend to run using your current metadata profile. Replacement is required because scheme metadata is added to these jobs when they are run for the first time. To update a job to use a different scheme repository, add a new Apply Lookup Standardization transformation to the job, configure the new transformation, delete the old transformation, and move the new transformation into place.

Create and Populate the Job

The example job that is described in this section uses an Apply Lookup Standardization transformation. This transformation applies one or more standardization 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. The specific process of scheme application varies based on your input. However, in general, when you apply a scheme to a source column, each value in that column is compared to all data values in the scheme. If the source value matches a scheme data value, the associated standardization value in the scheme is written into the target as a replacement for the source value. If no match is found, the source value is written into the target without change.

The first task is to create a job flow that reads a table with nonstandard data (MANUFACTURERS), uses a standardization scheme to correct the data, and then writes the corrected output to a target table (MANUFACTURERS_STANDARDIZED). The flow would look similar to the following figure:

Display 17.7 *Example Job with an Apply Lookup Standardization Transformation*



Perform the following steps to create and populate the job.

1. Create an empty SAS Data Integration Studio job.
2. In the **Data** folder of the Transformations tree, drag the Apply Lookup Standardization transformation into the empty job in the **Diagram** tab.
3. Select and drag a source table from its folder and drop it before the Apply Lookup Standardization transformation. In this sample job, the name of the source is MANUFACTURERS. The source provides contact information for suppliers of computer equipment. In the MANUFACTURERS table, the **Name** column contains

inconsistent values for the supplier named **`Computer Furniture`**, as depicted in the following display:

Display 17.8 *Source Table Data with Errors in the Name Column*

View Data: MANUFACTURERS (10 rows)

Go to row Go

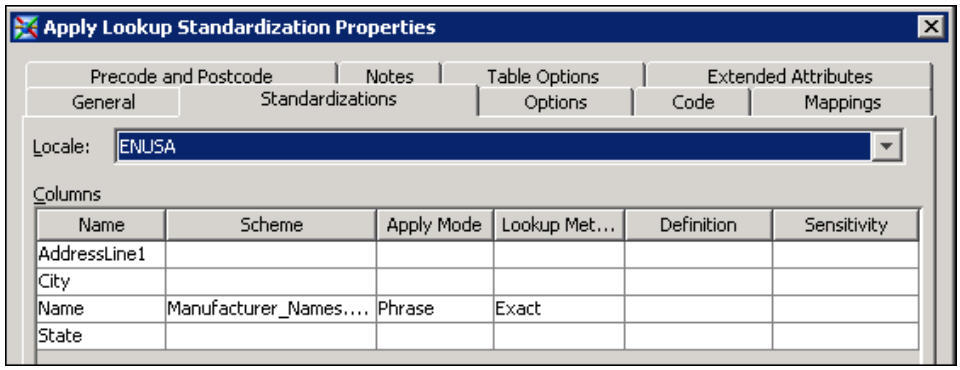
#	ManufacturerID	Name	AddressLine1	City	State	ZipCode
1	1001	Computer Furniture	404 Main St.	Atlanta	GA	30033
2	1002	ComputersAreUs	1116 Lafayette Dr.	Raleigh	NC	27615
3	1003	OfficeMate	105 Surrey Dr.	New York	NY	66809
4	1004	Math Designs	876 El Camino Dr.	Palo Alto	CA	54387
5	1001	Computer Furniture	404 Main St.	Atlanta	GA	30033
6	1001	Computer Furniture	404 Main St.	Atlanta	GA`	30033
7	1001	Cmputer Furniture	404 Main St.	Atlanta	GA	30033
8	1001	computer furniture	404 Main St.	Atlanta	GA	30033
9	1001	Computr Firniture	404 Main St.	Atlanta	GA	30033
10	1001	Comp Furn	404 Main St.	Atlanta	GA	30033

4. Drag the cursor from the source table to the input port of the Apply Lookup Standardization transformation. This action connects the source to the transformation.
5. In the **Access** folder of the Transformations tree, drag the Table Loader transformation into the empty job in the **Diagram** tab.
6. Drag the cursor from the output of the Apply Lookup Standardization to the input port of the Table Loader transformation. This action connects the two transformations. .
7. Drag the target table from its folder and drop it after the Table Loader transformation on the **Diagram** tab. In this sample job, the name of the target is MANUFACTURERS_STANDARDIZED. The target has the same columns as the source.
8. Drag the cursor from the output port of the Table Loader transformation to the target table. This action connects the transformation to the target. The job flow should now look similar to Display 17.7 on page 343.

Configure the Apply Lookup Standardization Transformation

The goal for this task is to associate the standardization scheme to the column or columns in the source that contain inconsistent values. This is done by selecting options on the **Standardizations** tab in the Apply Lookup transformation. An example set of options is shown in the next figure.

Display 17.9 *Options Selected on the Standardizations Tab*



Perform the following steps to configure the Apply Lookup Standardization transformation:

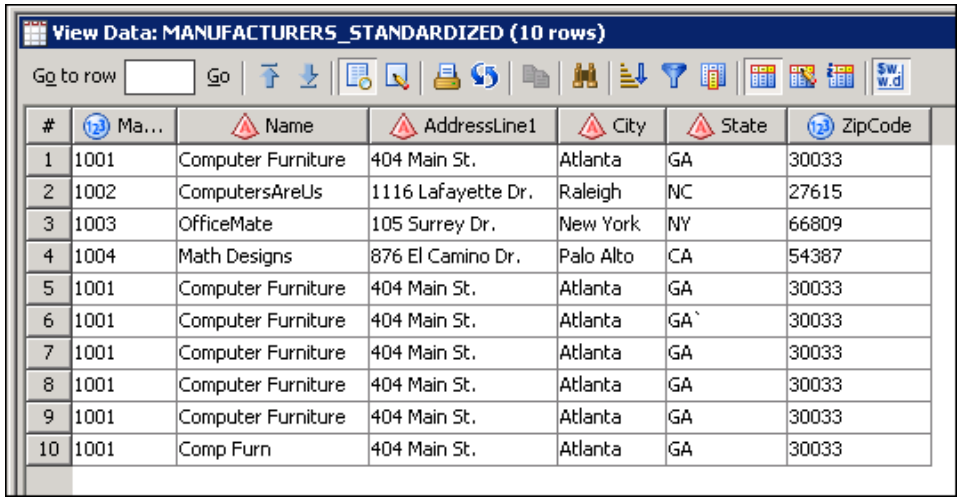
1. Open the properties window of the Apply Lookup Standardization transformation and display the **Standardizations** tab.
2. Right-click the down arrow in the **Locale** field to display the available locales. Select the locale that represents the national language and region that best represents your data. In the sample job, you could select **ENUSA** (English language, as implemented in the United States of America).
3. Specify the schemes to be applied to specified columns. In the sample job, right-click in the table cell of the **Name** row and the **Scheme** column. This action displays a list of available schemes in the scheme repository.
4. Select the scheme to be applied to the column. For the sample job, this is a scheme named **`Manufacturer_Names`**, which was created as described in "Create a Standardization Scheme" on page 342..
5. Click the **Apply Mode** column and select **Phrase**, which applies the standardizations to the entirety of each character string in the **Name** column.
6. The next step is to specify a value in the **Lookup Method** column. If you accept the default value of **`Exact`**, then only an exact match in your scheme will result in a corrected value being written to the target table. Alternatively, you could use match definitions as described in steps 7–9.
7. (Optional step) If appropriate match definitions are available in the selected locale, you could click the **Lookup Method** column and select **`Use Match Definition`**. Selecting **`Use Match Definition`** activates two related fields.
8. (Optional step associated with match codes) Click the **Definition** column to display a list of available match definitions. A match definition aims to help you decide whether two or more pieces of data might refer to the same real-life entity. To facilitate this, the definition generates a special string called a match code for each input. Any two inputs that generate the same match code are considered a match. Select a definition that is appropriate for the current column.
9. (Optional step associated with match codes) Use the **Sensitivity** column to control the precision of the match. A lower number is a less-exact match.
10. Click **OK** to save your input and close the properties window. The job is now ready to be run.

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.
2. After the completion of the job, right-click the target and select **Open** to view the standardized contents of the **Name** column. Note that one source value (**Comp Furn**) was not mapped in the standardization scheme that was created in "Create a Standardization Scheme" on page 342. All the other values were standardized. The following figure shows the target table data for the sample job.

Display 17.10 *Standardized Name Column in the Sample Target Table*



View Data: MANUFACTURERS_STANDARDIZED (10 rows)

Go to row Go

#	Ma...	Name	AddressLine1	City	State	ZipCode
1	1001	Computer Furniture	404 Main St.	Atlanta	GA	30033
2	1002	ComputersAreUs	1116 Lafayette Dr.	Raleigh	NC	27615
3	1003	OfficeMate	105 Surrey Dr.	New York	NY	66809
4	1004	Math Designs	876 El Camino Dr.	Palo Alto	CA	54387
5	1001	Computer Furniture	404 Main St.	Atlanta	GA	30033
6	1001	Computer Furniture	404 Main St.	Atlanta	GA`	30033
7	1001	Computer Furniture	404 Main St.	Atlanta	GA	30033
8	1001	Computer Furniture	404 Main St.	Atlanta	GA	30033
9	1001	Computer Furniture	404 Main St.	Atlanta	GA	30033
10	1001	Comp Furn	404 Main St.	Atlanta	GA	30033

Standardizing Values with a Definition

Problem

You want to standardize an element within a text string. For example, you might want to change all instances of "Court" to "Ct." but only when "Court" is used as a street suffix.

Solution

Get detailed information about the values that you want to change. Use that information to create a standardization definition that specifies the target element and maps old values to the new values. Use the definition in a SAS Data Integration Studio job to standardize the data in the appropriate columns.

In general, you would do the same tasks that are described in "Standardizing Values with a Standardization Scheme" on page 341. The main differences are as follows:

- Use DataFlux Data Management Studio to create a standardization definition that specifies the target element and maps old values to the new values. For more information about creating standardization definitions, see the standardization definition topics in the "Customize" chapter of the *DataFlux Data Management Studio User's Guide*. One way to find these topics is to display the help for DataFlux Data Management Studio. Click the **Search** tab in the left panel, then search for "standardization definition."
- Use SAS Data Integration Studio to create a job that includes a Standardize with Definition transformation. This transformation applies one or more standardization definitions to one or more columns in a source table.

Using Match Codes to Improve Record Matching

Problem

You want to use match codes to improve the quality of record-matching operations in jobs. Comparing match codes instead of actual data enables you to identify records that are in fact the same entity, despite minor variations in the data.

Solution

There are a number of ways to use match codes in SAS Data Integration Studio jobs. You can select **Use Match Definition** when this option is available for a transformation, as described in "Configure the Apply Lookup Standardization Transformation" on page 344. You can create a data service in DataFlux Data Management Studio that generates match codes and clustering information, and then call that service in a SAS Data Integration Studio job. For more information, see "Using a DataFlux Data Service in a SAS Data Integration Studio Job" on page 351.

You can also create a job in SAS Data Integration Studio that uses the Create Match Code transformation, as described in the "Tasks" section below. You would perform the following tasks:

- "Verify Prerequisites" on page 347
- "Create and Populate the Job" on page 347
- "Configure the Create Match Code Transformation" on page 349
- "Run the Job and View the Output" on page 350
- "Usage Notes" on page 351

Tasks

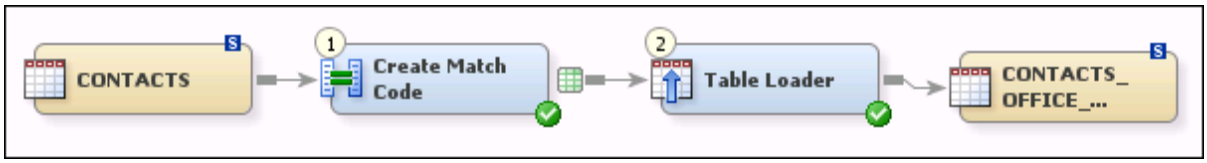
Verify Prerequisites

The Create Match Code transformation that is used in this topic requires SAS Data Quality Server. One or more locales must be available to SAS Data Integration Studio, as described in "Global Options on the Data Quality Tab" on page 336. Locales have a set of default match definitions that can be used to generate match codes. Assume that the sample job for this topic uses the standard match definitions for the **ENUSA** locale.

Create and Populate the Job

Match codes can be used to identify members of the same household in a set of demographic data. In order to do that, you could create a job flow that reads a table of demographic data (CONTACTS); generates match codes and cluster numbers for records that have the same last name and street address, and then writes the match codes and cluster numbers to a target table (CONTACTS_OFFICE_CLUSTER). The flow would look similar to the following figure.

Display 17.11 *Create Match Code Job Flow*



1. Create an empty SAS Data Integration Studio job.
2. From the **Data** folder in the Transformations tree, select and drag a **Create Match Code** 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 **Create Match Code** transformation on the **Diagram** tab.
4. Drag the cursor from the source table to the input port of the **Create Match Code** transformation. This action connects the transformation to the source. In this example, the source is a table of contact information called CONTACTS, which contains a large number of records. The data has not been standardized, so the spelling of names and addresses might differ while still referring to the same entities. The following display depicts the source data. When the job is run, rows 1004 and 1005 receive the same cluster number, as do rows 1007 and 1008, despite the fact that the data varies in the COMPANY and ADDRESS rows.

Display 17.12 *Source Data in the CONTACTS Table*

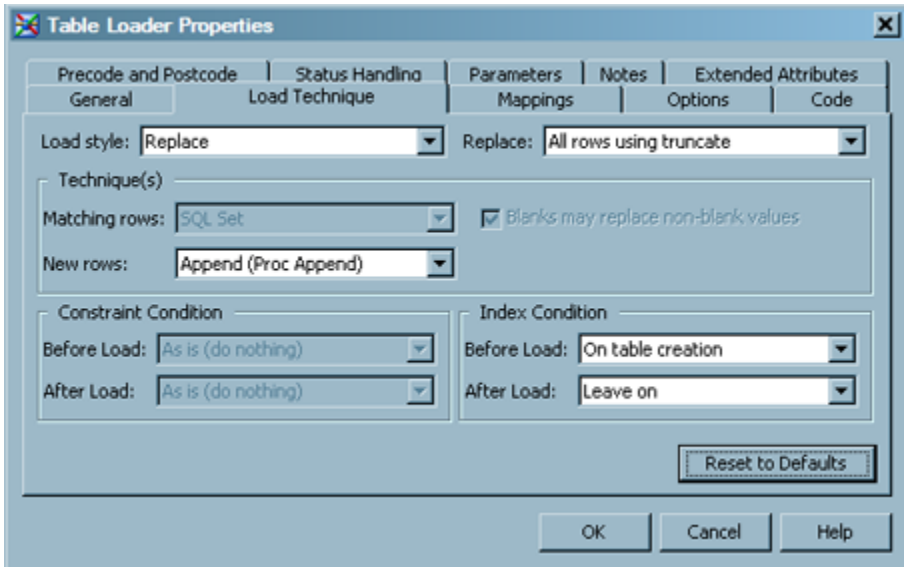
View Data: CONTACTS (3,276 rows)

Go to row Go

#	ID	COMPANY	CONTACT	ADDRESS	CITY	STATE	PHONE	DATE
1	1	First Merit Bank	James E. Brigg...	19 East Broad Street	Saint Lou...	Missouri	450-157-07...	14MAY19...
2	10	DataFlux Corporation	Bob Brauer	6512 Six Forks Road - 4...	Raleigh	North Carol...	323-198-32...	04MAY19...
3	100	KAISER HOSPITAL	LUTHER BAKE...	3560 E 116TH ST	Clevelan...	CA	651-245-79...	23MAR19...
4	1000	TransAmerica Life Insurance	Irene Greaves	555 W Fifth St	S Northfi...	OH	683-341-89...	19APR199...
5	1001	Transamerica Life Ins & Annuity	Rob Drain	7718 Elder Way	Clements...	OH	853-294-17...	22MAR19...
6	1002	Transamerica Occidental	Tonia Gerstne...	1018 N Hayworth Ave	Urb Cana...	OH	235-007-95...	08MAY19...
7	1003	Transamerica Financial Group	Joshua Hodge...	2134 Estado Cir	Old Missi...	PA	988-754-74...	08JAN199...
8	1004	Transamerica Financial Group	Nancy Weinsto...	2134 Estrado Cir	Old Missio...	PA	361-042-99...	21NOV19...
9	1005	Transamerica Financial Services	Mary Little	13845 N 56th Pl	Fort Wor...	OH	499-942-08...	11APR199...
10	1006	Transamerica Occidental Life	Kate Lindamoo...	33032 Lighthouse Ct	Charlesto...	OH	372-105-97...	18DEC199...
11	1007	Transamerica Financial Group	Justin Echavar...	2662 E 2nd St	Crawford...	OH	944-755-31...	18FEB199...
12	1008	Transamerica Financ Grp	Olivia Bach	2662 E 2nd St	Crawford...	OH	250-200-81...	26FEB199...
13	1009	Huntington Beach Union HS Dist	G. Weston	17272 Chapparal Ln	S. Creek	OH	779-369-75...	10MAY19...
14	101	KAISER HOSPITAL	KATHY JOHNS...	5798 GROVEWOOD DR	Mentor	CA	950-886-63...	01FEB199...
15	1010	Legislative Reference Bureau	V Eberhardt	3112 Quebrada Cir	Okauche...	OH	843-271-49...	14MAR19...
16	1011	Gallatin Medical Group Inc.	Jeremy Forero...	3405 W Danbury D230	Marion O...	OH	583-412-15...	29NOV19...

5. From the **Transformations** tab, under Access, drag a Table Loader transformation into the job and drop it after the Create Match Code transformation.
6. Select and drag from the transformation's temporary output table to the Table Loader transformation. This action connects the output of the transformation to the Table Loader. The Table Loader is used to ensure that the target is always completely overwritten each time the job is run. This default configuration for the Table Loader is depicted in the following display of the Table Loader's **Load Technique** tab.

Display 17.13 *Using the Table Loader to Overwrite the Target*



7. Select and drag the target table from its folder and drop it after the Table Loader transformation on the **Diagram** tab. In this example, the target is named CONTACTS_OFFICE_CLUSTER. The target contains the same columns as the source, plus a numeric column named CLUSTER and a character column named MATCH CODE (length 120).

8. Drag the cursor from the output port of the Table Loader transformation to the target table. This action connects the transformation to the target.

9. To propagate and map columns, right-click the Create Match Codes transformation and select **Propagate Columns ⇨ To Selected Transformation's Sources Sources ⇨ From Targets**. This action maps the source columns to the target and propagates the new columns in the target into the Create Match Codes transformation.

   The job flow should now look similar to Display 17.11 on page 348.

Configure the Create Match Code Transformation

Perform the following steps to configure the **Create Match Code** transformation:

1. In the Job Editor, double-click the **Create Match Code** transformation to display its properties window.

2. In the properties window, click the **Match Code** tab.

3. In the **Locale** field, select the locale that is most appropriate for your data. In this example, the locale is **ENUSA**.

4. In the **Cluster Column** field, select the new cluster column, which is named CLUSTER in this example.

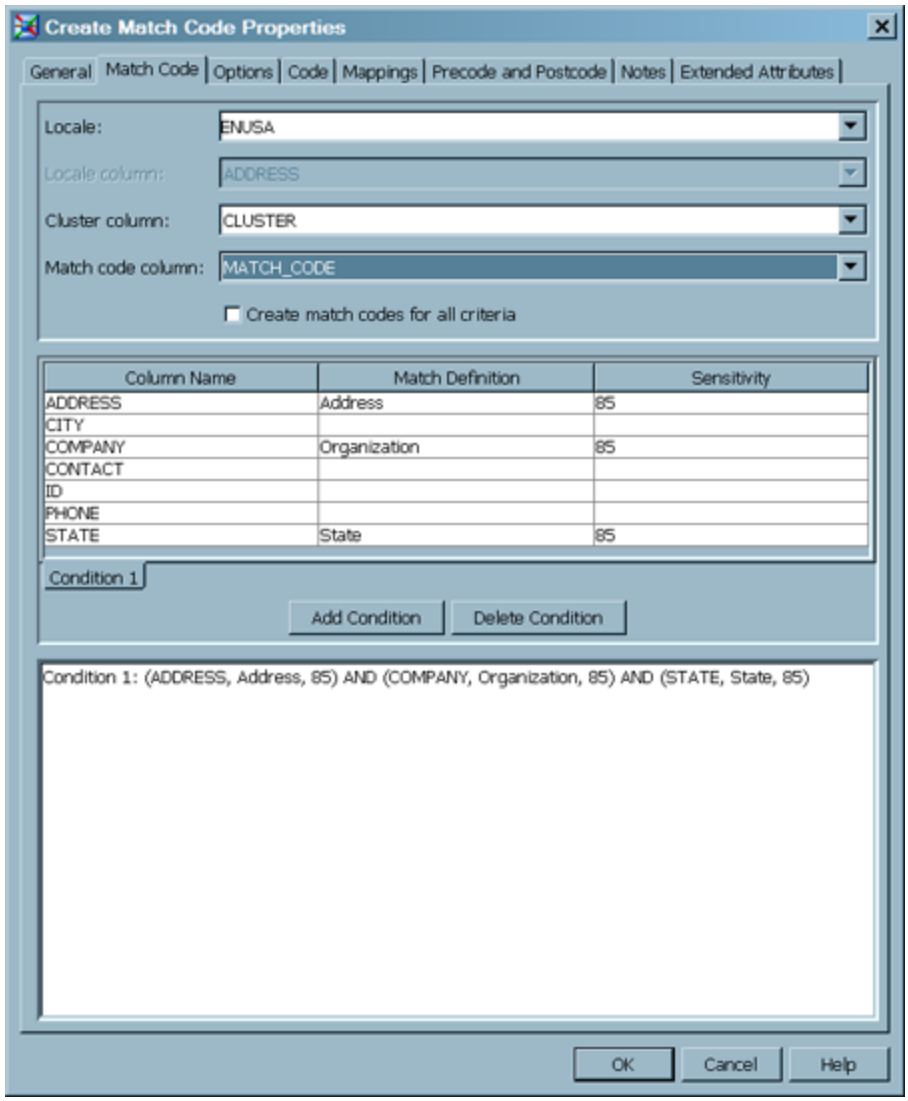
5. In the **Match code column** field, select the new match code column, which is MATCH_CODE in this example.

6. Set up one or more conditions that determine the assignment of cluster numbers. For this example, in the **Match Definition** column, for the **ADDRESS** column, pull down the list of available match definitions and select **Address**. In the **Sensitivity** column, leave the default value of **85** . A lower number is a less-exact match.

7. Repeat step 6 for the **COMPANY** column. Choose **Organization** as the match definition and leave the sensitivity value at **85**.

8. For the **STATE** column, choose the **State** match definition and leave the sensitivity setting of **85**. The following display shows the completed Match Code tab:

Display 17.14 *Fully Configured Match Code Tab*



9. Click **OK** to save your input and close the properties window. The job is now ready to run.

Run the Job and View the Output

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

1. Run the job.
2. If the job completes without error, go to the next step. If error messages appear, read and respond to the messages.
3. Right-click the target table and select **View Data**. The following display depicts the cluster and match code columns in the target.

Display 17.15 *Cluster Numbers and Match Codes in the Target Table*

View Data: CONTACTS_OFFICE_CLUSTER (3,276 rows)

Go to row Go

#	ID	COMPANY	CONTACT	CLUSTER	MATCH_CODE
3025	215	Allied Signal Inc.	Rich Temple	10	$$$$$$$$$$$$$$$!&W~4FPW$$$$$...
3026	219	Allied Signal Inc.	Todd Kotte	10	$$$$$$$$$$$$$$$!&W~4FPW$$$$$...
3027	483	Salt River Project	Susan Bradshaw	11	ZK00LYP$$$$$$$$!4W~YVYNYC$$$...
3028	486	Salt River Project	John Reiss	11	ZK00LYP$$$$$$$$!4W~YVYNYC$$$...
3029	490	San Diego County	Curt Delarosa	12	$$$$$$$5HZ$$$$$!4P8F3P~$$$$$$...
3030	488	San Diego County	Benjamin Mccorkle	12	$$$$$$$5HZ$$$$$!4P8F3P~$$$$$$...
3031	03	Jaxson Data Corporation	DONALD WILLIAMS	13	H56I2CL$$$$$$$$!CXP8~$$$$$$$$...
3032	04	The Jackson Data Corp.	DONALD F. WILLIAMS	13	H56I2CL$$$$$$$$!CXP8~$$$$$$$$...
3033	02	Jackson Data Co.	DON WILLIAMS	13	H56I2CL$$$$$$$$!CXP8~$$$$$$$$...
3034	06	Jackson Data Inc.	DONNY WILLIAMS	13	H56I2CL$$$$$$$$!CXP8~$$$$$$$$...
3035	07	The Jacksen Data Co...	DON WILIAMS	13	H56I2CL$$$$$$$$!CXP8~$$$$$$$$...
3036	05	Jackson Data	MR DON F WILLIAMS	13	H56I2CL$$$$$$$$!CXP8~$$$$$$$$...
3037	735	Northrop Corporation	Anthony Lutzker	14	D0Z$CGY$$$$$$$$!PY~2YN$$$$$$$...
3038	737	Northrop Corporation	W. Binns	14	D0Z$CGY$$$$$$$$!PY~2YN$$$$$$$...
3039	495	Glendale Advenist Me...	Elijah Mellor	15	ZK00LYP$$$$$$$$!FWP8W8VP4$$$...
3040	496	Glendale Advenist Me...	Debbie Cochrane	15	ZK00LYP$$$$$$$$!FWP8W8VP4$$$...
3041	977	Kings County	Bev Johanson	16	$$$$$$$$$$$$$$$!3PF3P~$$$$$$$...

Usage Notes

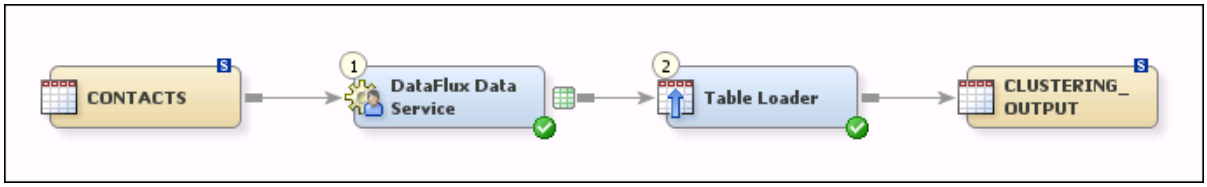
`ERROR: Failure in the clustering engine`. If you run a job that generates clustering information, and the job fails with this error in the log, try increasing the amount of memory that is allocated to the SAS Application Server that executes the job. To increase the memory allocation, set the option **–maxmemquery** to a higher value in the sasv9_usermods.cfg file. For example, you might set the option as follows:- **maxmemquery 600M**

Using a DataFlux Data Service in a SAS Data Integration Studio Job

Problem

You want to include a DataFlux Data Management Studio data service in the flow for a SAS Data Integration Studio job. For example, you could create a data service that generates match codes and clustering information. You could then call that service in the flow for a SAS Data Integration Studio job, as shown in the next figure.

Display 17.16 *SAS Data Integration Studio Job That Calls a Data Service*



For the purpose of illustration, the job shown above is similar in purpose to the sample job that is shown in Display 17.11 on page 348. However, you might want to use a DataFlux Data Service transformation to perform tasks that are a specialty of DataFlux software, such as profiling, monitoring, or address verification.

Solution

Create a data job in DataFlux Data Management Studio. Configure the job as a data service and deploy it to a DataFlux Data Management Server. Create a SAS Data Integration Studio job and add a DataFlux Data Service transformation to the flow. Configure this transformation so that it takes input from the SAS job, sends the input to the DataFlux data service, and then returns output from the service to the SAS job.

Perform the following tasks:

- "Verify Prerequisites" on page 352
- "Create a Data Service in DataFlux Data Management Studio" on page 352
- "Create and Populate a Job in SAS Data Integration Studio" on page 353
- "Run the Job and View the Output" on page 354

Tasks

Verify Prerequisites

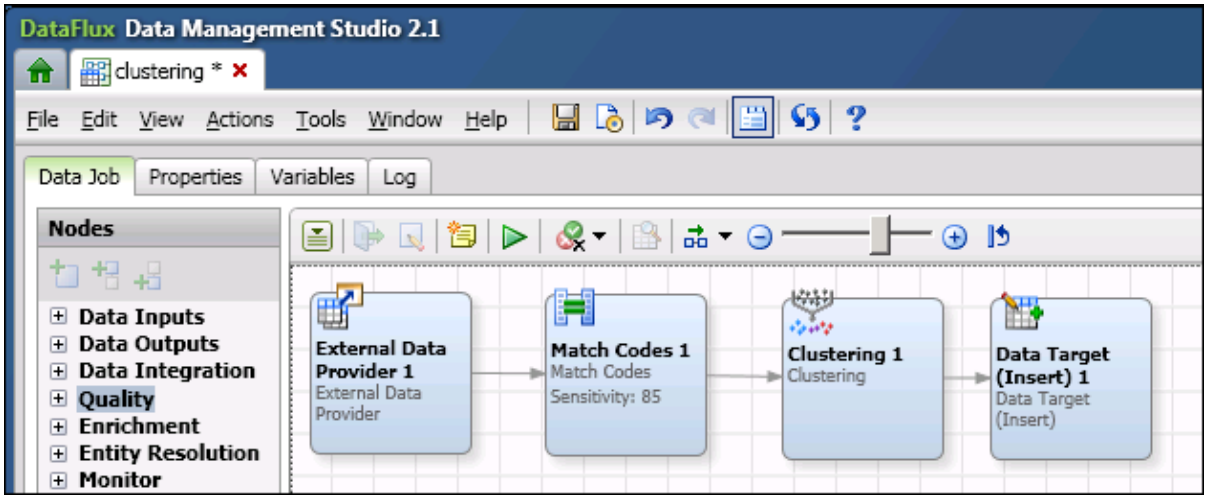
In addition to the "General Prerequisites for Data Quality Transformations", the "Prerequisites for Running a DataFlux Job or Profile in a SAS Data Integration Studio Job" on page 338 must be in place.

The current version of SAS Data Integration Studio can execute data services that were created with DataFlux Data Management Studio only. If you want to execute services that were created with DataFlux dfPower Studio, then the services must be migrated to the DataFlux Data Management Platform. For more information, see the *DataFlux Migration Guide*.

Create a Data Service in DataFlux Data Management Studio

A data service is a DataFlux Data Management Studio data job that has been configured as a real-time service and deployed to a DataFlux Data Management Server. For the current example, you would create a data service that generates match codes and cluster information. The flow for that job might look similar to the following figure.

Display 17.17 *Data Service in DataFlux Data Management Studio*

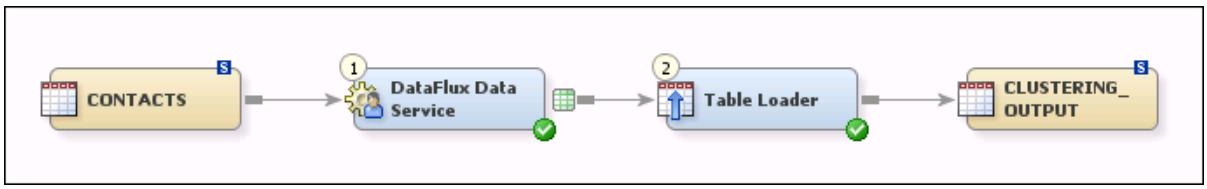


The job must be deployed to a DataFlux Data Management Server, so that it can be accessed from SAS Data Integration Studio. The first node in the flow (**External Data Provider**) takes input from the job in SAS Data Integration Studio, and the last node (**Data Target (Insert)**) return output to the job in SAS Data Integration Studio. For information about creating and deploying a data service in DataFlux Data Management Studio, see the topic "Deploying a Data Job as a Real-Time Service" in the Data Job chapter of the *DataFlux Data Management Studio User's Guide*.

Create and Populate a Job in SAS Data Integration Studio

For the current example, you would create a SAS Data Integration Studio job and add a DataFlux Data Service transformation to the flow, as shown in the next figure.

Display 17.18 *SAS Data Integration Studio Job That Calls a Data Service*

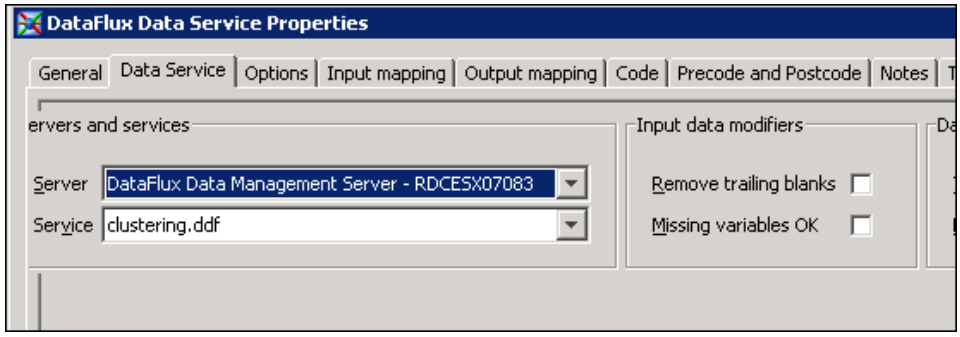


The sources and targets in the flow are added in the usual manner. The sources and targets shown above are similar to those in the sample job that is shown in Display 17.11 on page 348. In the current example, however, a data service is used instead of the Create Match Codes transformation.

Configure the DataFlux Data Service Transformation

Open the Properties window for the DataFlux Data Service transformation. On the **Data Service** tab, select the DataFlux Data Management Server and select the appropriate data service that was created in DataFlux Data Management Studio. The next figure shows the values for the sample job.

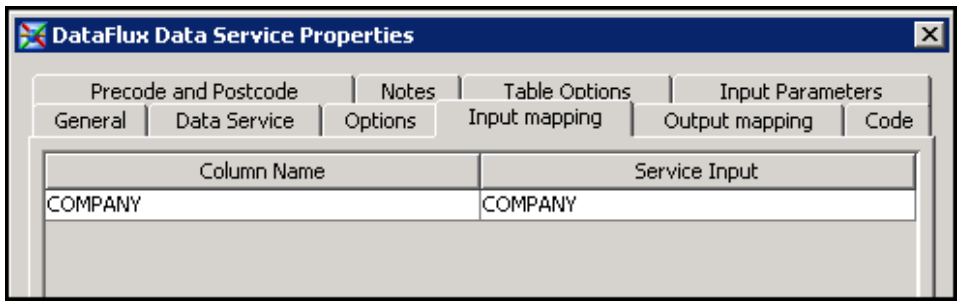
Display 17.19 *Data Service Tab*



In the previous figure, the **Server** field specifies the DataFlux Data Management Server where the data service was deployed. The **Service** field specifies the data service that you want to run in this step. The data service that you select here was created as described in "Configure the DataFlux Data Service Transformation" on page 353.

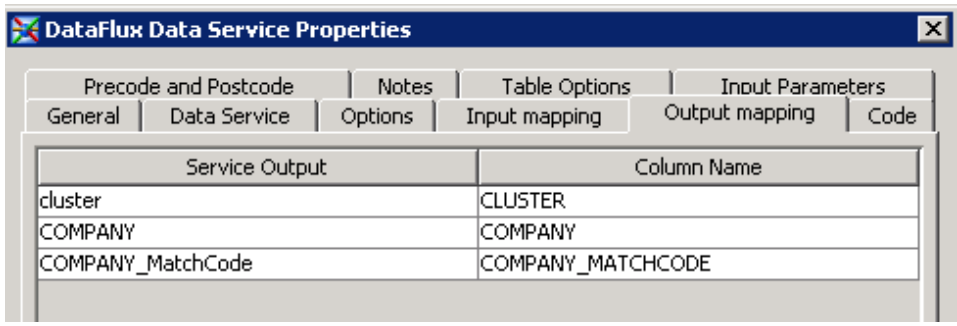
On the **Input Mapping** tab, map one or more input columns for the transformation to the corresponding inputs in the data service, as shown in the next figure.

Display 17.20 *Input Mapping Tab*



On the **Output Mapping** tab, map one or more output columns for the transformation to the corresponding outputs in the data service, as shown in the next figure.

Display 17.21 *Output Mapping Tab*



Click **OK** to save your input and close the Properties window. The job is now ready to run.

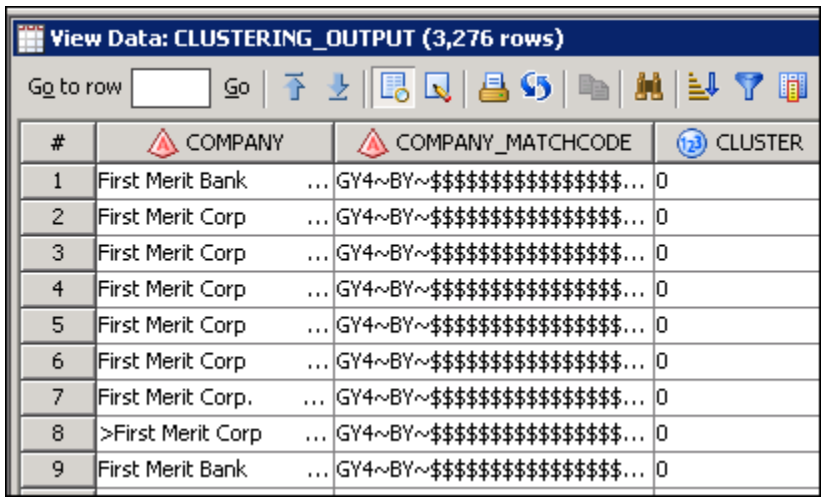
Run the Job and View the Output

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

1. Run the job.

2. If the job completes without error, go to the next step. If error messages appear, read and respond to the messages.
3. Right-click the target table and select **View Data**. The following display depicts the cluster and match code columns in the target.

Display 17.22 *Output from a DataFlux Data Service*



View Data: CLUSTERING_OUTPUT (3,276 rows)

Go to row Go

#	COMPANY	COMPANY_MATCHCODE	CLUSTER
1	First Merit Bank ...	GY4~BY~$$$$$$$$$$$$$$$...	0
2	First Merit Corp ...	GY4~BY~$$$$$$$$$$$$$$$...	0
3	First Merit Corp ...	GY4~BY~$$$$$$$$$$$$$$$...	0
4	First Merit Corp ...	GY4~BY~$$$$$$$$$$$$$$$...	0
5	First Merit Corp ...	GY4~BY~$$$$$$$$$$$$$$$...	0
6	First Merit Corp ...	GY4~BY~$$$$$$$$$$$$$$$...	0
7	First Merit Corp. ...	GY4~BY~$$$$$$$$$$$$$$$...	0
8	>First Merit Corp ...	GY4~BY~$$$$$$$$$$$$$$$...	0
9	First Merit Bank ...	GY4~BY~$$$$$$$$$$$$$$$...	0

Using a DataFlux Job or Profile in a SAS Data Integration Studio Job

Problem

You want to incorporate a DataFlux Data Management Studio data job, process job, or profile into the flow for a SAS Data Integration Studio job.

Solution

Create or identify a data job, process job, or profile in DataFlux Data Management Studio. Deploy the job or profile to a DataFlux Data Management Server. Create a SAS Data Integration Studio job and add a DataFlux Batch Job transformation to the job. Configure this transformation so that it specifies the DataFlux job or profile on the server. Execute the SAS Data Integration Studio job.

You will perform the following tasks:

- "Verify Prerequisites" on page 356
- "Create or Identify a DataFlux Job or Profile" on page 356
- "Create and Populate a Job in SAS Data Integration Studio" on page 356
- "Configure the DataFlux Batch Job Transformation" on page 357

Tasks

Verify Prerequisites

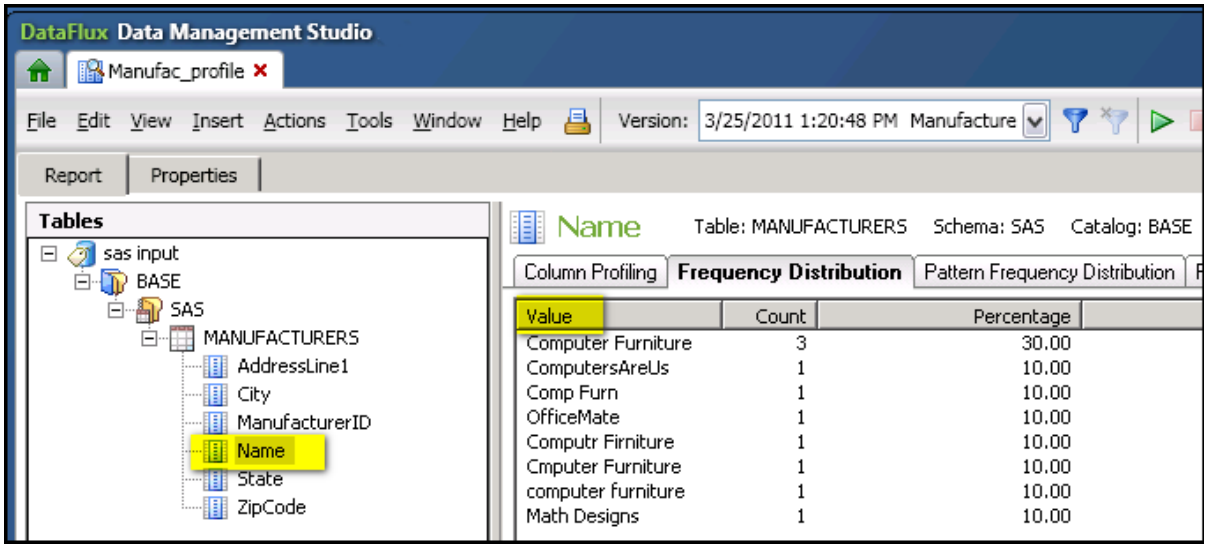
In addition to the "General Prerequisites for Data Quality Transformations", the "Prerequisites for Running a DataFlux Job or Profile in a SAS Data Integration Studio Job" on page 338 must be in place.

The current version of SAS Data Integration Studio can execute data jobs, process jobs, and profiles that were created with DataFlux Data Management Studio. You can also execute Architect jobs that were created with DataFlux dfPower Studio, if the Architect jobs do not contain macros. Architect jobs that contain macros must be migrated to DataFlux Data Management Studio. For more information, see the *DataFlux Migration Guide*.

Create or Identify a DataFlux Job or Profile

Create or identify a DataFlux job or profile. For example, you could choose the DataFlux Data Management Studio profile for the MANUFACTURERS table, as shown in the next figure.

Display 17.23 *Profile Shows Data Errors in the Name Column of the MANUFACTURERS Table*



The job must be deployed to a DataFlux Data Management Server, so that it can be accessed from SAS Data Integration Studio. For information about data jobs, process jobs, and profiles, see the appropriate chapters in the *DataFlux Data Management Studio User's Guide*.

Create and Populate a Job in SAS Data Integration Studio

Create a SAS Data Integration Studio job and add a DataFlux Batch job transformation to the job, as shown in the next figure.

Display 17.24 *Job with a DataFlux Batch Job Transformation*

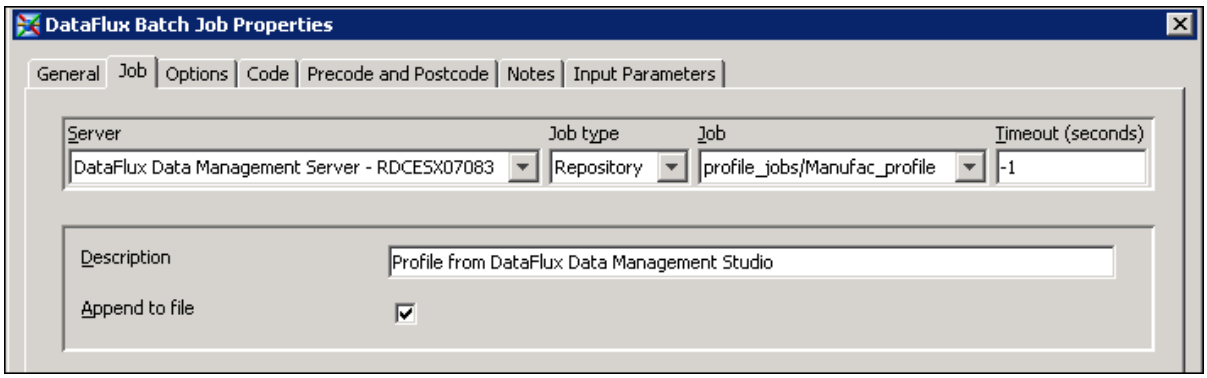


The DataFlux Batch Job transformation has no connection ports for data inputs or data outputs. It is just a reference to a DataFlux job.

Configure the DataFlux Batch Job Transformation

Open the Properties window for the DataFlux Batch Job transformation. On the **Job** tab, select the DataFlux Data Management Server and select the appropriate DataFlux job. The next figure shows the values for the sample job.

Display 17.25 *Job Tab for the DataFlux Batch Job Transformation*



In the previous figure, the **Server** field specifies the DataFlux Data Management Server where the job was deployed.

The **Job type** field specifies the type of DataFlux job that is available in the **Job** field. Select **Batch** for file-based jobs, such as data jobs and process jobs. Select **Repository** for repository-based jobs, such as profiles.

The **Job** field specifies the job to be executed.

Click **OK** to save your input and close the properties window. The job is now ready to run. When you run the job, the specified DataFlux job is executed. Depending on the nature of the job, the results might not be viewable in SAS Data Integration Studio.

Part 3

Working with Transformations

Chapter 18
Working with Analysis Transformations *361*

Chapter 19
Working with Loader Transformations *421*

Chapter 20
Working with SAS Sort Transformations *435*

Chapter 21
Working with SQL Join Transformations *441*

Chapter 22
Working with Other SQL Transformations *489*

Chapter 23
Working with Iterative Jobs and Parallel Processing *505*

Chapter 24
Working with Slowly Changing Dimensions *521*

Chapter 25
Working with Change Data Capture *557*

Chapter 26
Working with Message Queues *567*

Chapter 27
Working with SPD Server Cluster Tables *577*

Chapter 28
Working with Hadoop and SAS LASR Analytic Server *583*

Chapter 18

Working with Analysis Transformations

About Analysis Transformations **362**

Creating a Correlation Analysis **362**
Overview 362
Problem 363
Solution 363
Tasks 363

Creating a Distribution Analysis **370**
Overview 370
Problem 371
Solution 371
Tasks 372

Generating Forecasts **377**
Overview 377
Problem 378
Solution 378
Tasks 379

Frequency of Eye Color By Hair Color Crosstabulation **385**
Overview 385
Problem 386
Solution 386
Tasks 386

One-Way Frequency of Eye Color By Region **398**
Overview 398
Problem 398
Solution 399
Tasks 399

Creating Summary Statistics for a Table **407**
Overview 407
Problem 407
Solution 408
Tasks 408

Creating a Summary Tables Report from Table Data **413**
Overview 413
Problem 413
Solution 413
Tasks 414

About Analysis Transformations

The Analysis folder of the transformations tree contains seven transformations. You can use these transformations to add analytical functions to the process flows in your SAS Data Integration Studio jobs. The following analysis transformations are provided:

- "Creating a Correlation Analysis" on page 362
- "Creating a Distribution Analysis" on page 370
- "Generating Forecasts" on page 377
- "Frequency of Eye Color By Hair Color Crosstabulation" on page 385
- "One-Way Frequency of Eye Color By Region" on page 398
- "Creating Summary Statistics for a Table" on page 407
- "Creating a Summary Tables Report from Table Data" on page 413

Creating a Correlation Analysis

Overview

The Correlations transformation generates one of the following types of correlation statistics:

- Hoeffding
- Kendall
- Pearson
- Spearman

The Correlations transformation is based on the CORR procedure, which is documented in the *Base SAS Procedures Guide: Statistical Procedures*. The CORR procedure computes Pearson correlation coefficients, three nonparametric measures of association, and the probabilities associated with these statistics. The correlation statistics include the following:

- Pearson product-moment correlation
- Spearman rank-order correlation
- Kendall's tau-b coefficient
- Hoeffding's measure of dependence, D
- Pearson, Spearman, and Kendall partial correlation

Pearson product-moment correlation is a parametric measure of a linear relationship between two variables. For nonparametric measures of association, Spearman rank-order correlation uses the ranks of the data values and Kendall's tau-b uses the number of concordances and discordances in paired observations. Hoeffding's measure of dependence is another nonparametric measure of association that detects more general departures from independence. A partial correlation provides a measure of the correlation between two variables after controlling the effects of other variables.

You can specify which columns are correlated and which columns are analyzed. You can group rows in the output based on the values in specified grouping columns. Output appears in a target table or in the **Output** tab in the process designer. ODS output in the form of HTML, PDF, or RTF can also be sent to a folder on the SAS Application Server that executes the job or to any folder that is accessible to that SAS Application Server.

The target receives data only for the source columns that are involved in the correlation. The target requires two columns that the Correlations transformation populates: _TYPE_ specifies the type of the statistic and _NAME_ identifies the correlation column.

The Correlations transformation requires that grouping columns be sorted in ascending order in the source. If you specify grouping columns, you can sort those columns before the Correlations transformation by using a SAS Sort transformation.

Problem

You want to use the CORR procedure to generate a correlation analysis.

Solution

You can use the Correlations transformation in a job that generates a correlation analysis and creates an ODS document that contains its results. This transformation uses the CORR procedure to compute Pearson correlation coefficients, three nonparametric measures of association, and the probabilities associated with these statistics. For example, you can create a job similar to the sample job featured in this topic. Note that the output for this job is sent to a target table, the **Output** tab in the Job Editor window, and an ODS document that is configured in the job. This sample job generates a correlation analysis that is based on a table of botanical data. The sample job includes the following tasks:

- "Create and Populate the Job" on page 363
- "Configure Analytical Options" on page 364
- "Configure Reporting Options" on page 366
- "Run the Job and View the Output" on page 367

Tasks

Create and Populate the Job

Perform the following steps to create and populate the job:

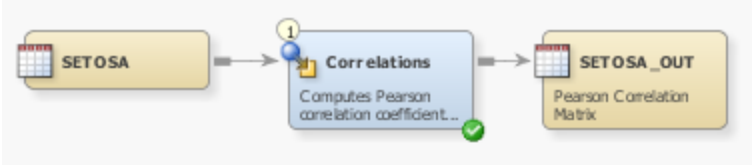
1. Create an empty SAS Data Integration Studio job.
2. Select and drag a Correlations transformation from the Analysis folder in the Transformations tree. Then, drop it in the empty job on the **Diagram** tab in the Job Editor window.
3. Select and drag the source table from the Inventory tree. Then, drop it before the Correlations transformation on the **Diagram** tab.
4. Drag the cursor from the source table to the input port of the Correlations transformation. This action connects the source to the transformation.
5. Right-click the Correlations transformation, and click **Add Output Port** from the **Ports** option in the drop-down menu. This step enables you to add an output port to the transformation.

Note: If you want multiple statistical output tables, you must first set the correct number of tables in the Output data window in the **Options** tab of the Properties window. Once you have set the number of tables in the Output data window, add the same number of output ports to the transformation.

6. Select and drag the source table from the Inventory tree. Then, drop it after the Correlations transformation on the **Diagram** tab.
7. Drag the cursor from the Correlations transformation output port to the target table. This action connects the target to the transformation.

The following display shows a sample process flow diagram for a job that contains the Correlations transformation:

Display 18.1 *Sample Process Flow*



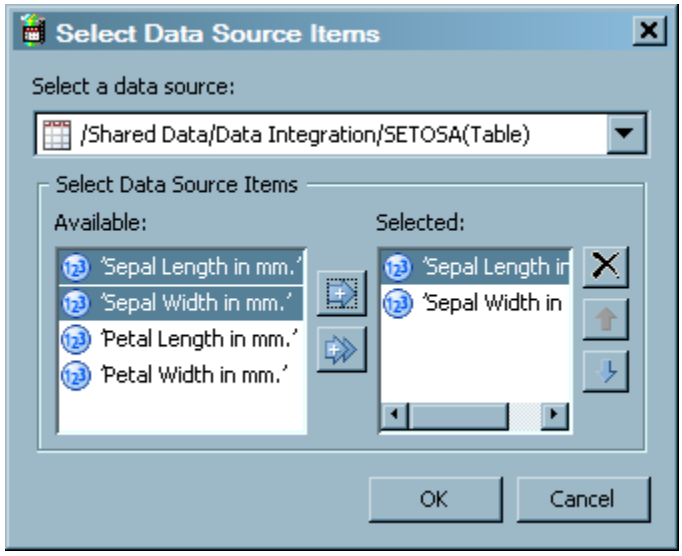
Note that the source table for the sample job is named SETOSA and that the target table is named SETOSA_OUT.

Configure Analytical Options

Use the **Options** tab in the properties window for the Correlations transformation to configure the output for your analysis. Note that the **Options** tab is divided into two parts, with a list of categories on the left-hand side and the options for the selected category on the right-hand side. Perform the following steps to set the options that you need for your job:

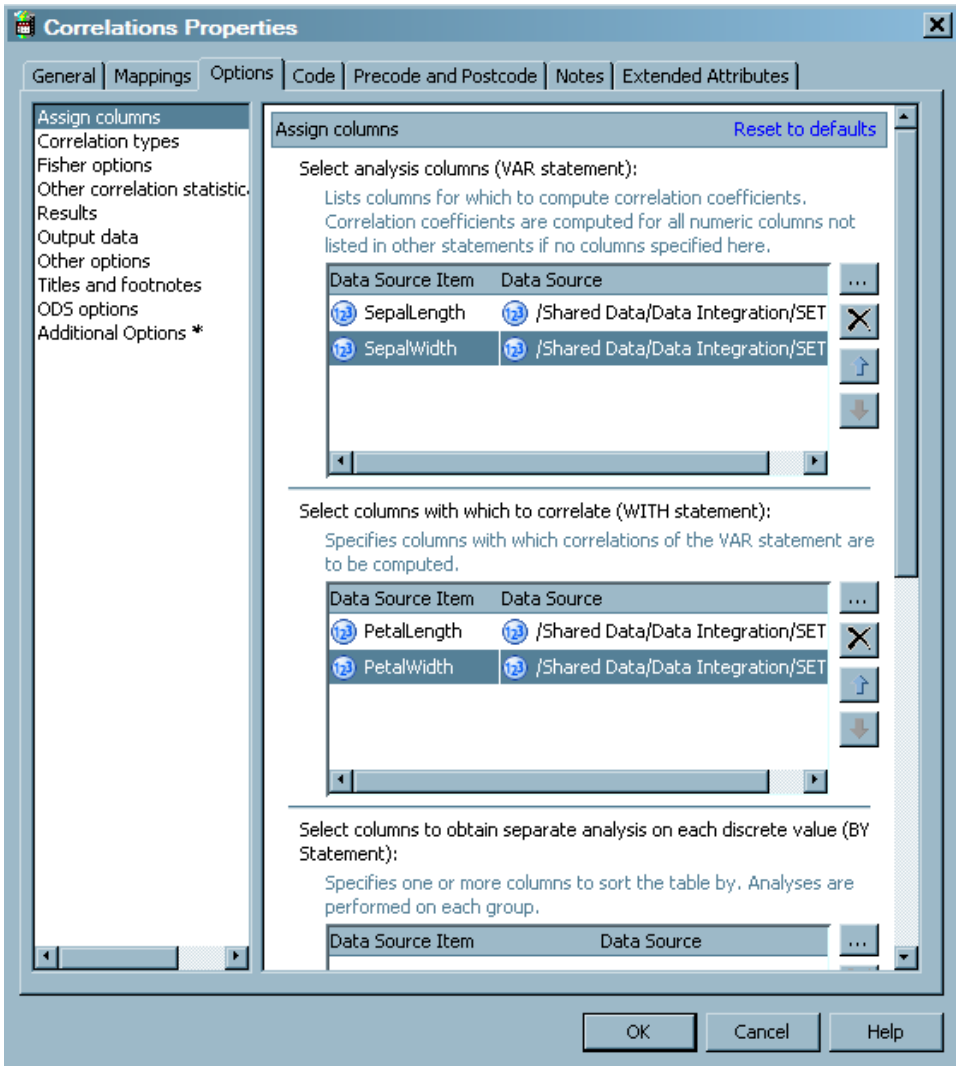
1. Open the properties window for the Correlations transformation in the **Diagram** tab in the Job Editor window. Then, click the **Options** tab.
2. Click **Assign columns** to access the Assign columns page. Use the column selection prompts to access the columns that you need for your job. For example, you can click ... for the **Select analysis columns (VAR statement)** to access the Select Data Source Items window, as shown in the following display:

Display 18.2 *Sample Select Data Source Items Window*



In the sample job, the VAR statement columns are SepalLength and SepalWidth. The column assignment options are shown in the following display:

Display 18.3 *Sample Options Properties*



3. Note that you must select the other columns that you need for your job, such as the PetalLength and PetalWidth columns in the WITH statement required for the sample job.

4. Set the remaining options for your analysis in the appropriate fields. The sample job keeps the default Pearson product-moment correlation type and adds the COV and SSCP options on the Correlation type page. These options are enabled when you select **Yes** in the drop-down menu for the field and disabled when you select **No**.

5. Set any necessary options on the remaining analytical options pages. For example, the **Update the metadata for the target tables** option on the Additional Options page is enabled and default options for the Fisher options, Other correlation statistical options, Output data, Results, and Other options pages are retained. A reporting option is also set on the Other correlation statistical options page.

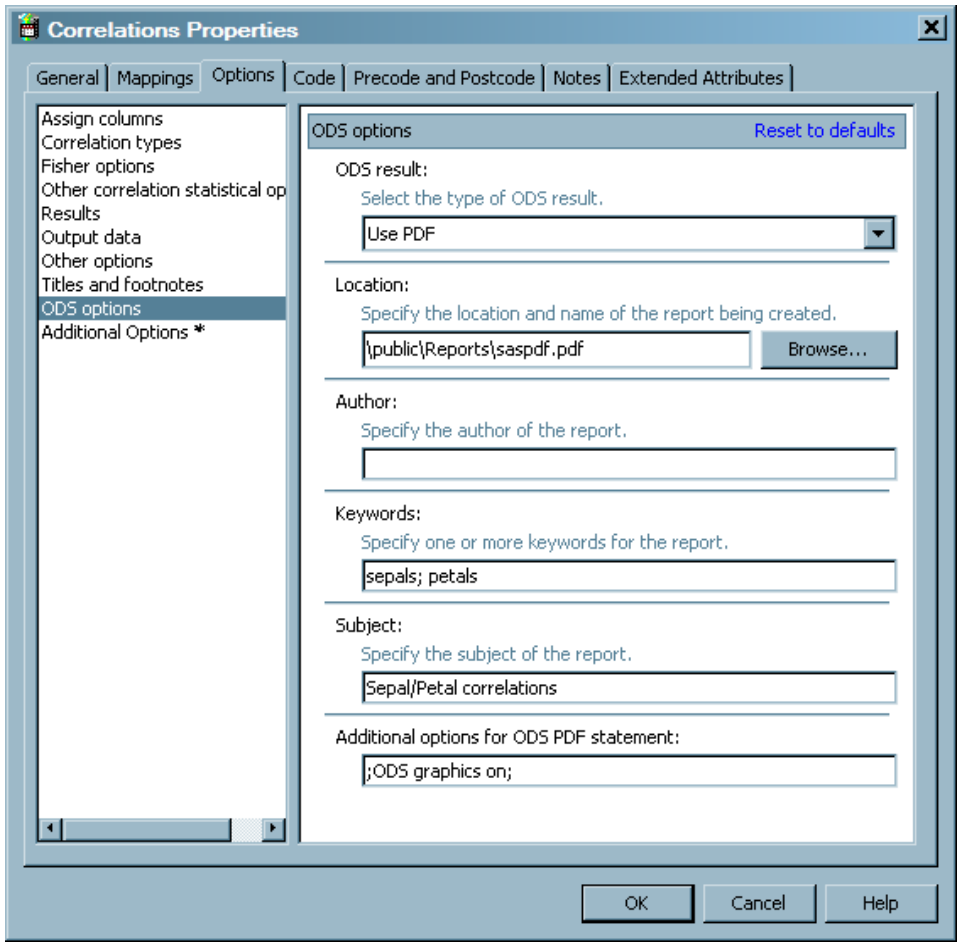
Configure Reporting Options

Use the remaining option pages to create and save a report that is based on the analysis conducted in the job. Perform the following steps to set the reporting options:

1. Click **Titles and footnotes** to access the Titles and footnotes page and enter up to three headings and two footnotes.

2. Click **ODS options** to access the ODS options page. You can choose between HTML, RTF, and PDF output and enter appropriate settings for each. The sample job uses PDF output. Therefore, a location, a set of keywords, the subject of the report, and code to enable ODS graphics are added to the fields that are displayed when **Use PDF** is selected in the **ODS Result** field. (The path specified in the **Location** field is relative to the SAS Application Server that executes the job.) These fields are shown in the following display:

Display 18.4 *Sample ODS Options*



Note: The plots for descriptive statistics option in the **Plots option (PLOTS)** field on the Other correlation statistical options page is also enabled. This step enables the inclusion of a scatter plot matrix in the PDF output.

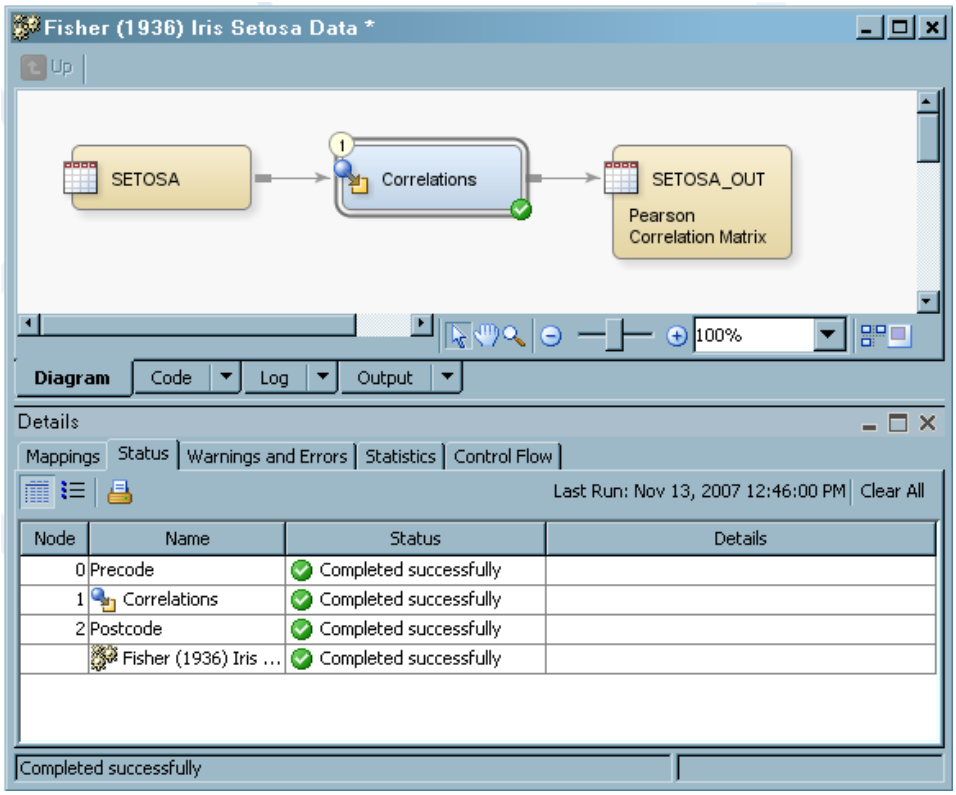
3. Click **OK** to save the settings for the **Options** tab.

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. The following display shows a successful run of a sample job:

Display 18.5 *Successfully Completed Sample Job*



2. If error messages are displayed on the **Status** tab, read and respond to the messages as needed.

3. To view the correlation analysis, click the **Output** tab in the Job Editor window. The following display shows the analysis for the sample job:

Display 18.6 *Sample Output in the Output Tab*

Sums of Squares and Crossproducts
SSCP / Row Var SS / Col Var SS

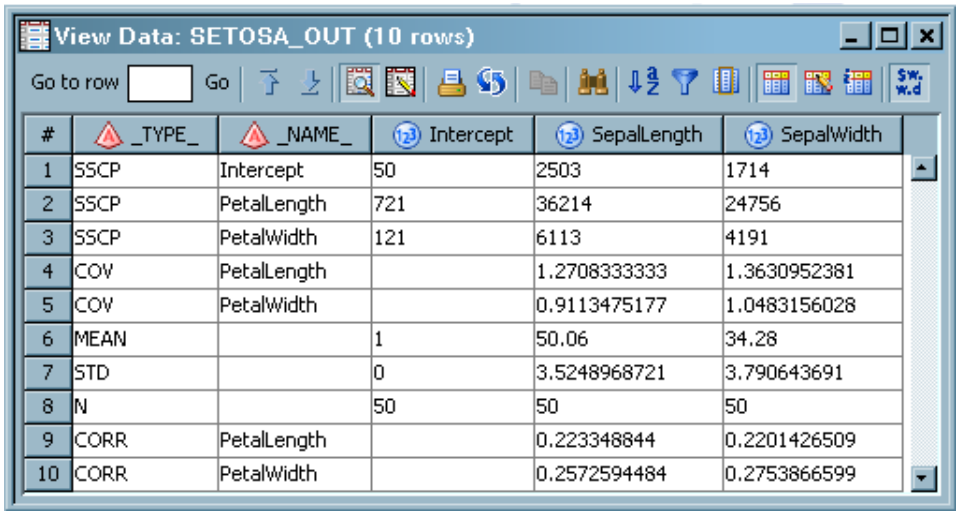
	SepalLength	SepalWidth
PetalLength	36214.00000	24756.00000
'Petal Length in mm.'	10735.00000	10735.00000
	123793.0000	58164.0000
PetalWidth	6113.00000	4191.00000
'Petal Width in mm.'	355.00000	355.00000
	121356.0000	56879.0000

Variances and Covariances
Covariance / Row Var Variance / Col Var Variance / DF

	SepalLength	SepalWidth
PetalLength	1.270833333	1.363095238
'Petal Length in mm.'	2.625000000	2.625000000
	12.33333333	14.60544218
	48	48
PetalWidth	0.911347518	1.048315603
'Petal Width in mm.'	1.063386525	1.063386525
	11.80141844	13.62721631
	47	47

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 18.7 *Sample Target Table Data*



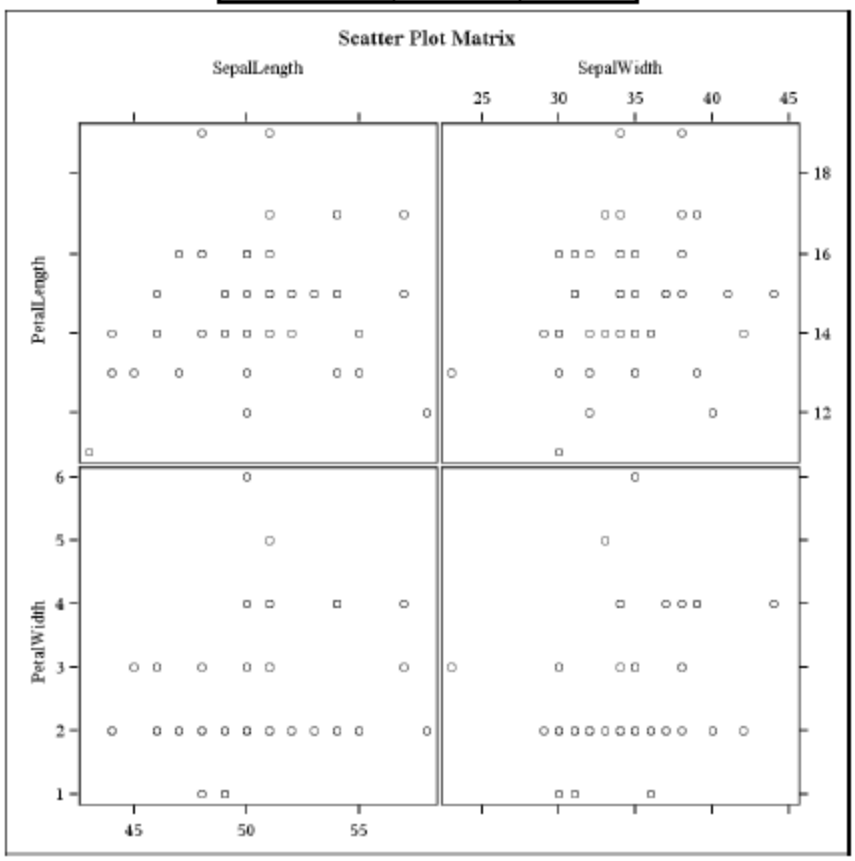
#	_TYPE_	_NAME_	Intercept	SepalLength	SepalWidth
1	SSCP	Intercept	50	2503	1714
2	SSCP	PetalLength	721	36214	24756
3	SSCP	PetalWidth	121	6113	4191
4	COV	PetalLength		1.2708333333	1.3630952381
5	COV	PetalWidth		0.9113475177	1.0483156028
6	MEAN		1	50.06	34.28
7	STD		0	3.5248968721	3.790643691
8	N		50	50	50
9	CORR	PetalLength		0.223348844	0.2201426509
10	CORR	PetalWidth		0.2572594484	0.2753866599

5. Open the PDF document that you created and saved earlier. The following display illustrates a sample report based on the correlations data:

Display 18.8 *Sample PDF Output*

The CORR Procedure

Pearson Correlation Coefficients Prob > |r| under H0: Rho=0 Number of Observations		
	SepalLength	SepalWidth
PetalLength 'Petal Length in mm.'	0.22335 0.1229 49	0.22014 0.1285 49
PetalWidth 'Petal Width in mm.'	0.25726 0.0775 48	0.27539 0.0582 48



Creating a Distribution Analysis

Overview

Use the Distribution Analysis transformation to generate distribution analysis data in a target table and on the **Output** tab of the Job Editor. The target receives data only for the columns that are involved in the analysis. You can control many aspects of how data is generated, including choosing the type of analysis and which columns are analyzed.

The Distribution Analysis transformation is based on the UNIVARIATE procedure, which is documented in the "The UNIVARIATE Procedure" section in *Base SAS Procedures Guide: Statistical Procedures*.

The UNIVARIATE procedure provides the following:

- descriptive statistics based on moments (including skewness and kurtosis), quantiles or percentiles (such as the median), frequency tables, and extreme values
- histograms and comparative histograms. These can also be fitted with probability density curves for various distributions and with threaded kernel density estimates.
- quantile-quantile plots (Q-Q plots) and probability plots. These plots facilitate the comparison of a data distribution with various theoretical distributions.
- goodness-of-fit tests for a variety of distributions including the normal
- the ability to inset summary statistics on plots produced on a graphics device
- the ability to analyze data sets with a frequency variable
- the ability to create output data sets containing summary statistics, histogram intervals, and parameters of fitted curves

You can use the UNIVARIATE procedure, together with the VAR statement, to compute summary statistics. In addition, you can use the following statements to request plots:

- the HISTOGRAM statement for creating histograms, the QQPLOT statement for creating Q-Q plots, and the PROBPLOT statement for creating probability plots.
- the CLASS statement together with the HISTOGRAM, QQPLOT, and PROBPLOT statement for creating comparative histograms, Q-Q plots, and probability plots.
- the INSET statement with any of the plot statements for enhancing the plot with an inset table of summary statistics. The INSET statement is applicable only to plots produced on graphics devices.

You can specify grouping columns in the Distribution Analysis transformation. Doing so causes a SAS BY statement to order target rows according to the values in the grouping columns. The Distribution Analysis transformation requires that grouping columns be sorted in ascending order in the source. If you specify grouping columns, you can sort those columns before the Distribution Analysis transformation by using a SAS Sort transformation.

Problem

You want to generate a distribution analysis.

Solution

You can use Distribution Analysis transformation as an interface to the UNIVARIATE procedure in a job that generates a distribution analysis and creates an ODS document that contains its results. For example, you can create a job similar to the sample job featured in this topic. This sample job generates a distribution analysis that is based on a table of data about home loans. The output for this job is sent to a target table, the **Output** tab in the Job Editor window, and an ODS document that is configured in the job. The sample job includes the following tasks:

- "Create and Populate the Job" on page 372
- "Configure Analytical Options" on page 372

- "Configure Reporting Options" on page 374
- "Run the Job and View the Output" on page 375

Tasks

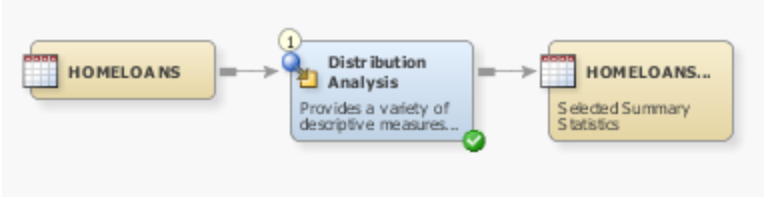
Create and Populate the Job

Perform the following steps to create and populate the job:

1. Create an empty SAS Data Integration Studio job.
2. Select and drag a Distribution Analysis transformation from the **Analysis** folder in the Transformations tree. Then, drop it in the empty job on the **Diagram** tab in the Job Editor window.
3. Select and drag the source table out of the Inventory tree. Then, drop it before the Distribution Analysis transformation on the **Diagram** tab.
4. Drag the cursor from the source table to the input port of the Distribution Analysis transformation. This action connects the source to the transformation.
5. Right-click the Distribution Analysis transformation, and click **Add Output Port** from the **Ports** option in the drop-down menu. This step enables you to add an output port to the transformation.
6. Select and drag the source table from the Inventory tree. Then, drop it after the Distribution Analysis transformation on the **Diagram** tab.
7. Drag the cursor from the Distribution Analysis transformation output port to the target table. This action connects the target to the transformation.

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

Display 18.9 *Sample Process Flow*



Note that the source table for the sample job is named HOMELOANS, and the target table is named HomeLoans_out.

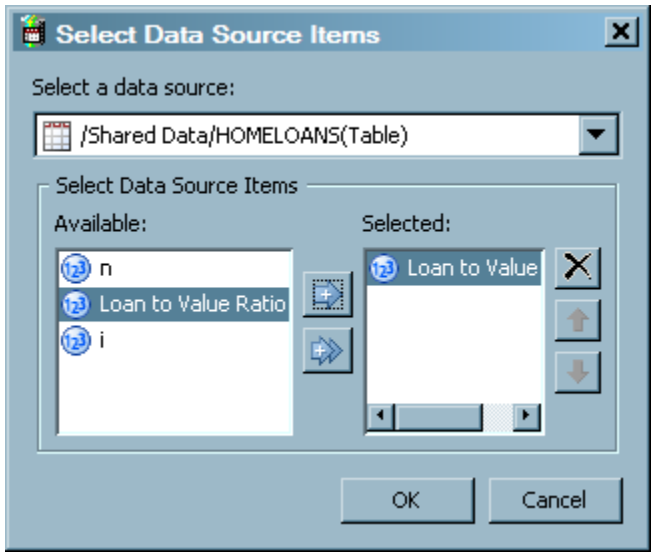
Configure Analytical Options

Use the **Options** tab in the properties window for the Distribution Analysis transformation to configure the output for your analysis. Note that the **Options** tab is divided into two parts, with a list of categories on the left-hand side and the options for the selected category on the right-hand side. Perform the following steps to set the options that you need for your job:

1. Open the properties window for the Distribution Analysis transformation on the **Diagram** tab in the Job Editor window. Then, click the **Options** tab.
2. Click **Assign columns** to access the Assign columns page. Use the column selection prompts to access the columns that you need for your job. For example, you can

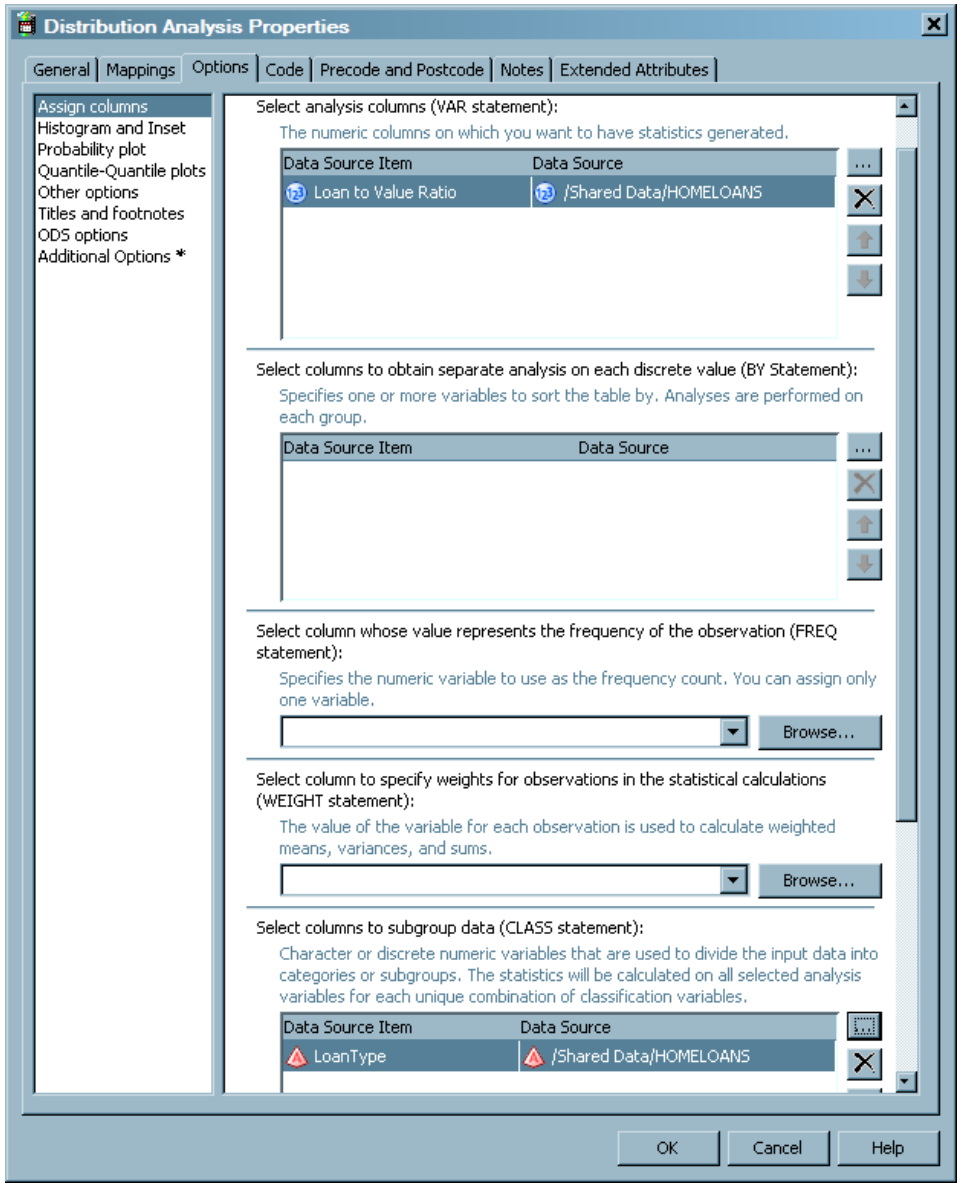
click ... for the **Select analysis columns (VAR statement)** field to access the Select Data Source Items window, as shown in the following display.

Display 18.10 *Sample Select Data Source Items Window*



In the sample job, the VAR statement column is Loan to Value Ratio. The column assignment options are shown in the following display.

Display 18.11 *Sample Options Properties*



3. Note that you must select the other columns that you need for your job, such as the Loan Type column in the CLASS statement required for the sample job.
4. Enter the other options that you need for your analysis. In the sample job, options are set in the Histogram and Inset page to generate a histogram for the analysis.

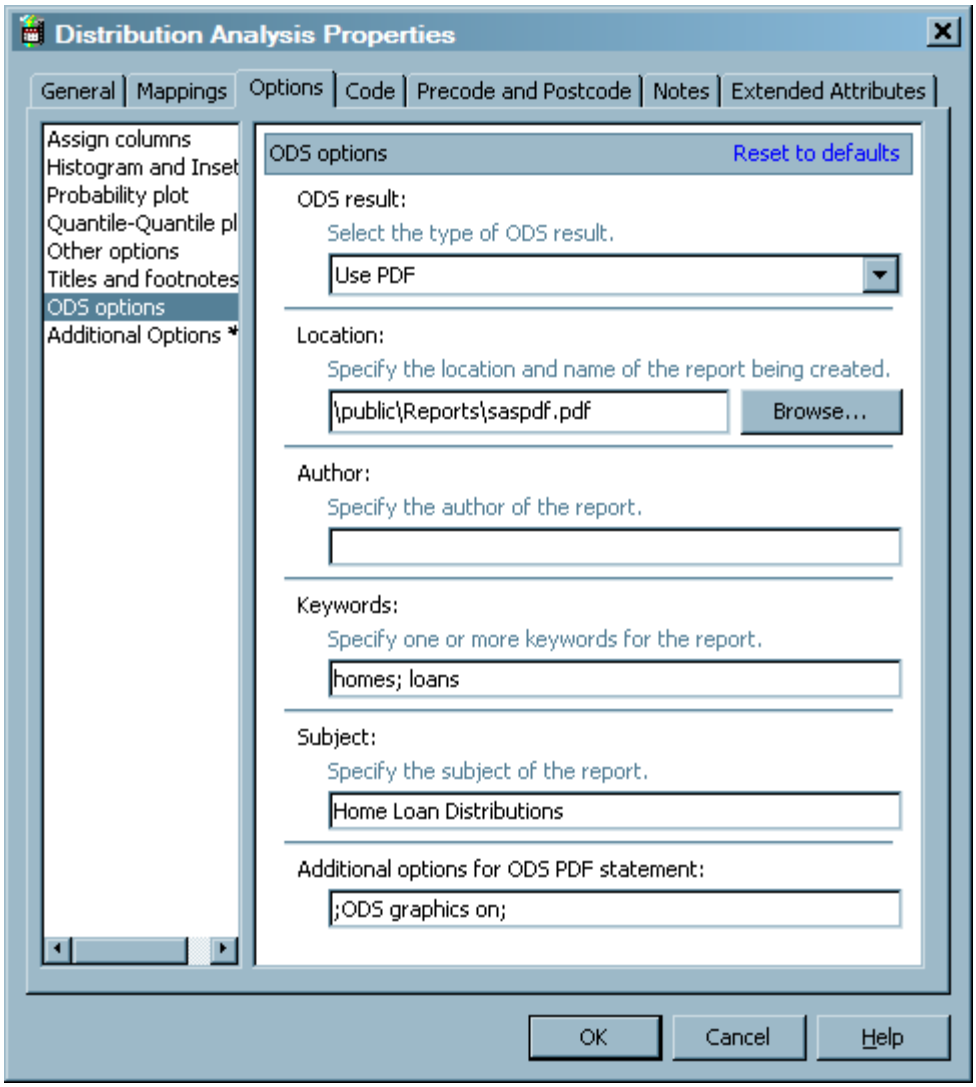
Configure Reporting Options

Use the remaining option pages to create and save a report based on the analysis conducted in the job. Perform the following steps to set the reporting options:

1. Click **Title and footnotes** to access the Title and footnotes page and enter up to three headings and two footnotes.
2. Click **ODS options** to access the ODS options page. You can choose between HTML, RTF, and PDF output and enter appropriate settings for each. The sample job uses PDF output. Therefore, a location, a set of keywords, the subject of the report, and code to enable ODS graphics are added to the fields that are displayed when **Use PDF** is selected in the **ODS Result** field. (The path specified in the

Location field is relative to the SAS Application Server that executes the job.) These fields are shown in the following display.

Display 18.12 *Sample ODS Options*



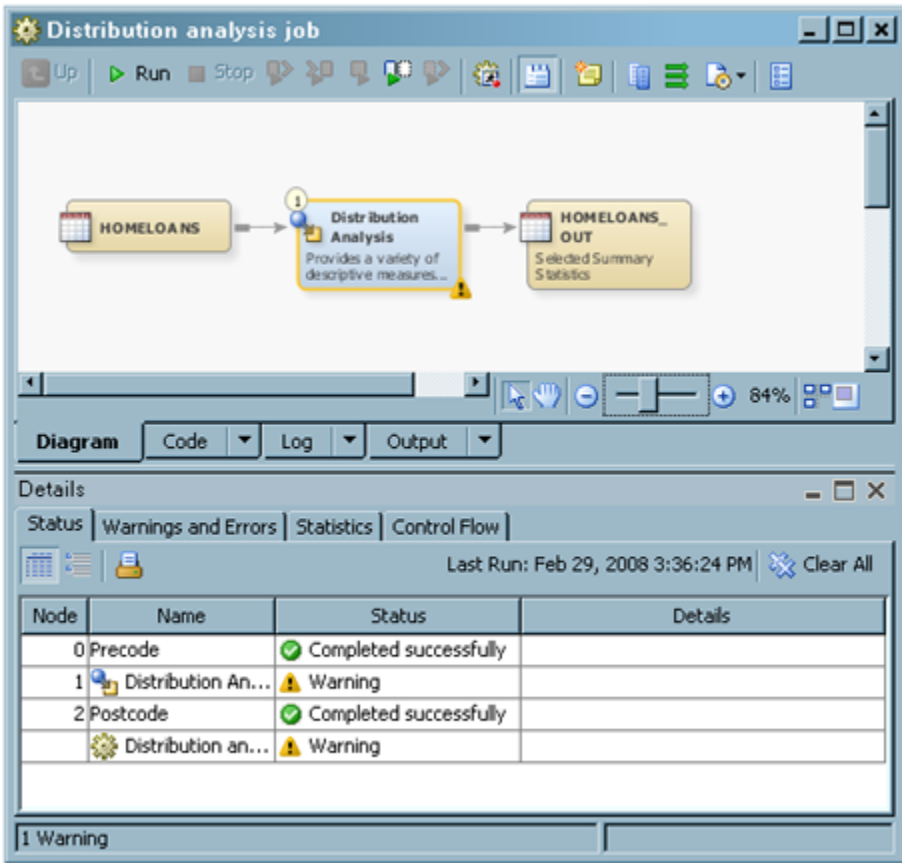
3. Click **OK** to save the settings for the **Options** tabs.

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. The following display shows a successful run of a sample job.

Display 18.13 *Sample Completed Job*



2. If error messages display on the **Status** tab, read and respond to the messages as needed. The sample jobs display warning messages because ODS graphics are experimental for this transformation. The expected output is still displayed on the **Output** tab and in the PDF report that is generated in the job.

3. To view the distribution analysis, click the **Output** tab in the Job Editor window. If the **Output** tab is not available, enable it at **Tools Options** ⇨ **Show Output tab** in the menu bar. The following display shows a portion of the analysis for the sample job.

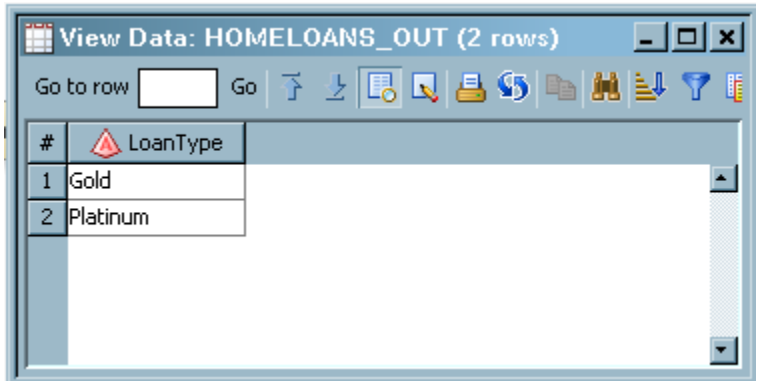
Display 18.14 *Sample Output in the Output Tab*

Basic Statistical Measures

Location		Variability	
Mean	0.400582	Std Deviation	0.14763
Median	0.366168	Variance	0.02180
Mode	0.350000	Range	1.09748
		Interquartile Range	0.11571

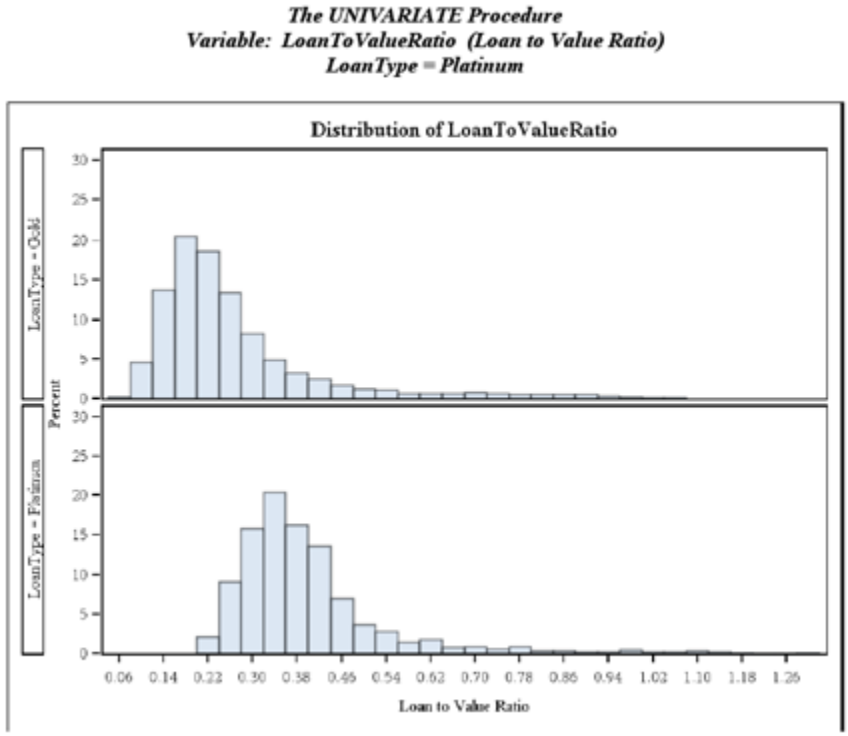
4. To view the target table, right-click the target and select **Open**. The following display shows the target table data for the sample job.

Display 18.15 *Target Table Data*



5. Open the PDF document that you created and saved earlier. The following display shows the histogram generated in a sample report based on the data.

Display 18.16 *Sample PDF Output*



Generating Forecasts

Overview

Use the Forecasting transformation to run the High-Performance Forecasting procedure (PROC HPF) against a warehouse data store. PROC HPF provides a quick and automatic way to generate forecasts for many sets of time series or transactional data.

The procedure can forecast millions of series at a time, with the series organized into separate variables or across BY groups. The Forecasting transformation provides a simple interface for entering values for various options that are associated with PROC HPF.

The Forecasting transformation can forecast either time series or transactional data:

- Time series data consists or observations that are equally spaced by a specific time interval, such as a month or week.
- Transactional data consists of observations that are not spaced with respect to any particular time interval. Typical examples of transactional data include information that is drawn from the Internet, inventory, and sales. For transactional data, the data is accumulated based on a specified time interval to form a procedure reference. The transformation can also perform trend and seasonal analysis on this transactional data.

The following prerequisites apply to the Forecasting transformation:

- SAS High-Performance Forecasting software must be installed on the SAS Application Server that executes a job that includes the Forecasting transformation.
- If you use plot options in the transformation, you will need to have the SAS/GRAPH component installed on the SAS Application Server that executes the job.

Problem

You want to generate a forecast in the context of a SAS Data Integration Studio job.

Solution

You can use the Forecasting transformation. The transformation runs the High-Performance Forecasting procedure (PROC HPF) against a warehouse data store. The options that are included in the Forecasting transformation give you the flexibility to tailor the output to meet your business needs.

PROC HPF provides a quick and automatic way to generate forecasts for many sets of time series or transactional data. Note that SAS High-Performance Forecasting software must be installed on the SAS Application Server that executes a job that includes the Forecasting transformation. Perform the following tasks:

- "Create and Populate the Job" on page 379
- "Set HPF Statement Options" on page 379
- "Set BY VARIABLE Statement Options" on page 380
- "Set ID Statement Options" on page 380
- "Set FORECAST Statement Options" on page 381
- "Set Target Table Options" on page 382
- "Configure the Report Output" on page 382
- "Run the Job" on page 383
- "View the Output" on page 383

Tasks

Create and Populate the Job

Perform the following steps to create and populate the job:

1. Create an empty job.
2. Drop the source table onto the **Diagram** tab of the Job Editor window.
3. Select and drag a Forecasting transformation from the Analysis folder in the Transformations tree in the Job Editor window. The following display shows a sample process flow for a forecasting job:

Display 18.17 *Sample Forecasting Process Flow*



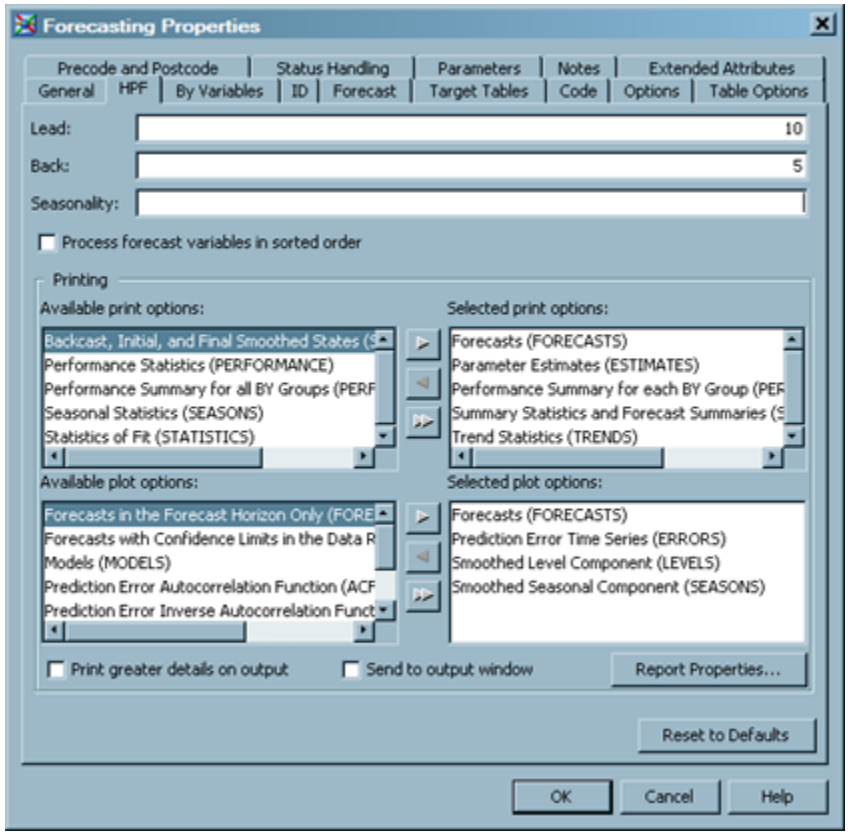
Note that the source table is named PRICEDATA. The output document and tables are created during the configuration of the Forecasting transformation.

Set HPF Statement Options

The **HPF** tab in the properties window of the Forecasting transformation enables you to set options in the HPF statement in PROC HPF. Perform the following steps to set HPF statement options:

1. Open the **HPF** tab in the properties window for the **Forecasting** transformation.
2. Enter the HPF statement options that you need to generate your forecast. The following display shows the HPF options for a sample job:

Display 18.18 *Sample HPF Options*



Note that the number of the periods preceding and following the forecast are set in the **Lead** and **Back** fields for this sample job. Appropriate print and plot options are also set. The print options specify the types of data that are printed in the output. The plot options specify the graphical plots that are included in the output. Use the arrow keys to move between the available options and selected options fields.

Set BY VARIABLE Statement Options

The **By Variables** tab provides an interface to the BY statement, which you can use to obtain separate analyses for groups of statements defined by the BY variables. Perform the following steps to set BY statement options:

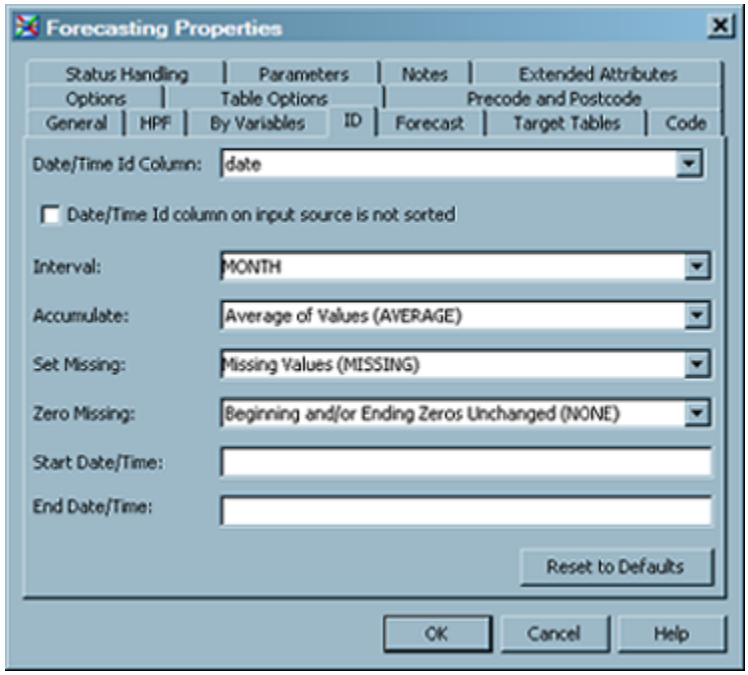
1. Click **By Variable**.
2. Select the appropriate columns from **Columns** field and move them to the **Sort by columns** field. For example, the values for the region and products columns are selected in the sample job. Keep the default **Ascending** sort order setting.

Set ID Statement Options

The **ID** tab provides an interface to the ID statement, which you can use to designate a numerical variable that identifies observations in the input and output data sets. Perform the following steps to set ID statement options:

1. Click **ID**.
2. Set appropriate values for the ID statement. The following display shows the ID options for a sample job:

Display 18.19 *Sample ID Options*



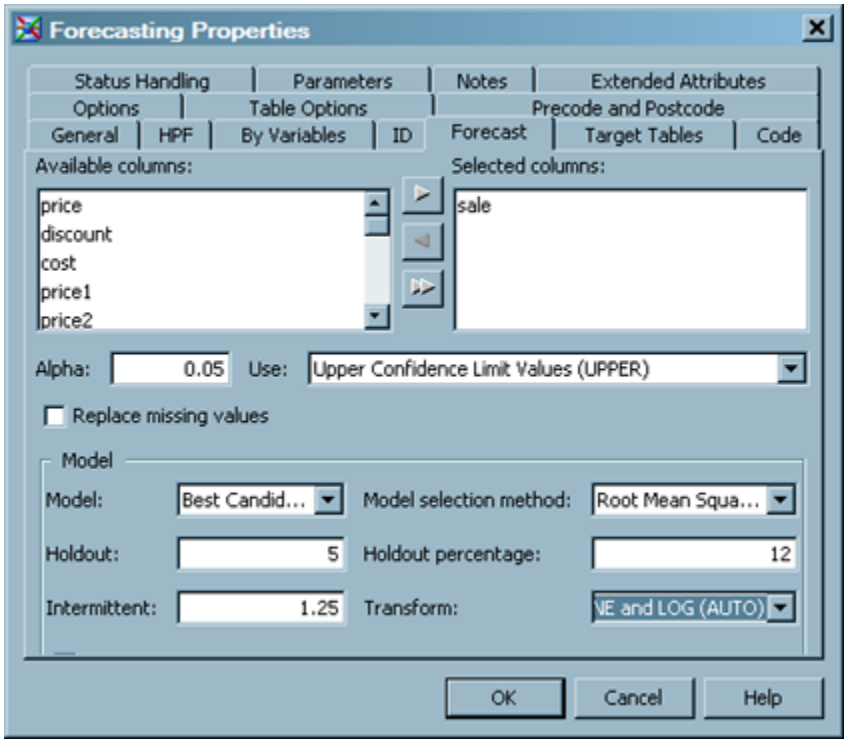
Note that **date** is selected in the **Data/Time Id Column** field for the sample job and that appropriate values are selected in the **Interval**, **Accumulate**, **Set Missing**, and **Zero Missing** fields. You can also specify start and end dates and times, if appropriate for your forecast.

Set FORECAST Statement Options

The **Forecast** tab provides an interface to the FORECAST statement, which you can use to list the numeric variables in the data set. The accumulated values in this data set represent the time series that is to be modeled and forecast. Perform the following steps to set FORECAST statement options:

1. Click **Forecast**.
2. Set appropriate values for the FORECAST statement. The following display shows the forecast options for a sample job:

Display 18.20 *Sample FORECAST Options*



Note that **sale** is selected in the **Selected columns** field for the sample forecast. In addition, *0.01* is entered in the **Alpha** field. This setting specifies the significance level to use in computing the confidence limits of the forecast. The default is ALPHA=0.05, which produces 95% confidence intervals. Similar settings are made in the **Use**, **Model**, **Model selection method**, **Intermittent**, and **Transform** fields to support the sample forecast.

Set Target Table Options

The **Target Tables** tab provides an interface for selecting the tables that are generated in the forecast output. You can select any combination of the tables that are listed on the tab. Perform the following steps to select your target tables:

1. Click **Target Tables**.
2. Select the appropriate values for your forecast. Note that the **Model Parameter Estimates** check box and the **Forecast Time Series Components** check box are selected in the sample job. Therefore, the Model Parameter Estimates and Forecast Time Series Components target tables are included in the output of the sample forecast.
3. Click the **OK** button to save the settings in the Forecasting properties window and return to the Job Editor window.

Configure the Report Output

This configuration ensures that the output is directed to the target tables directory and that the titles of the tables and the HTML document make sense to anyone who needs to review the forecast results. Perform the following steps to configure the output of the forecast HTML document:

1. Open the properties window for the output document in the forecasting job. Then, click the **Details** tab.

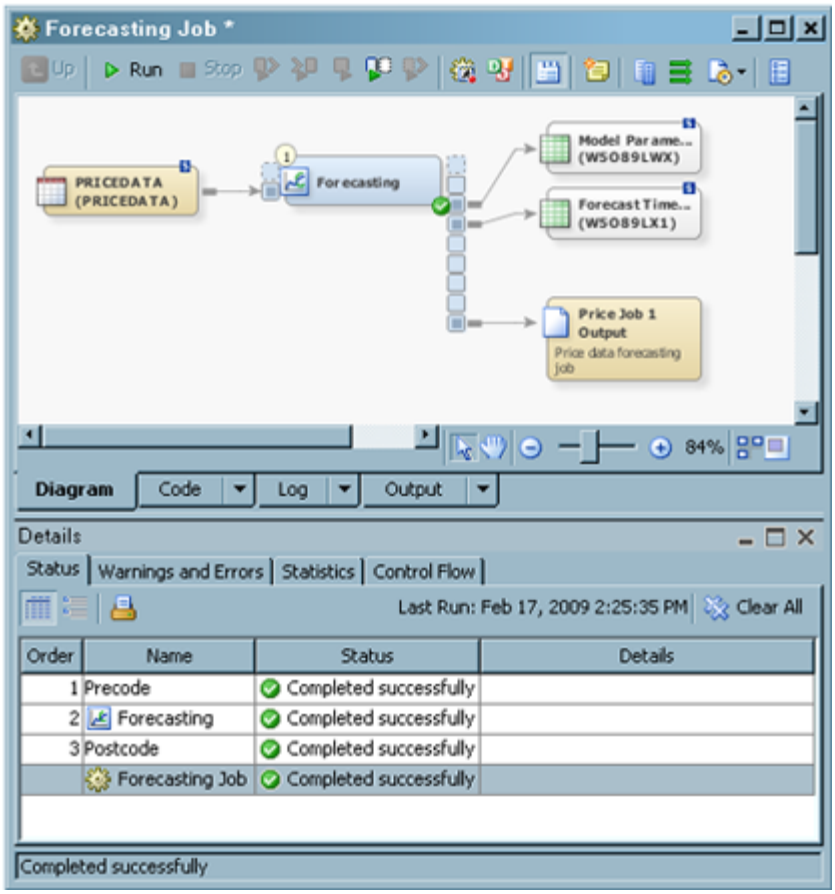
2. Enter the path to the directory where you store your target tables in the **Path** field.
3. Click **OK** to save the settings in the properties window and return to the **Diagram** tab of the Job Editor window.

Run the Job

Perform the following steps to run the forecasting job:

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. The following display shows a completed forecasting job:

Display 18.21 *Sample Completed Forecasting Job*



View the Output

Perform the following steps to verify that the job created the desired output:

1. Right-click the output document in the **Diagram** tab, and click **Open** in the pop-up menu. The output file in the sample job is named Price Job 1 Output. (You might be prompted to enter a user ID and password for the server that accesses the table.)
2. The HTML output of the forecast is displayed in your default Web browser. This file contains two types of output: tabular data and graphical plots. The following display shows the sample tabular data:

Display 18.22 *Sample Forecast Table*

The SAS System

The HPF Procedure

Region ID=1 Product ID=1

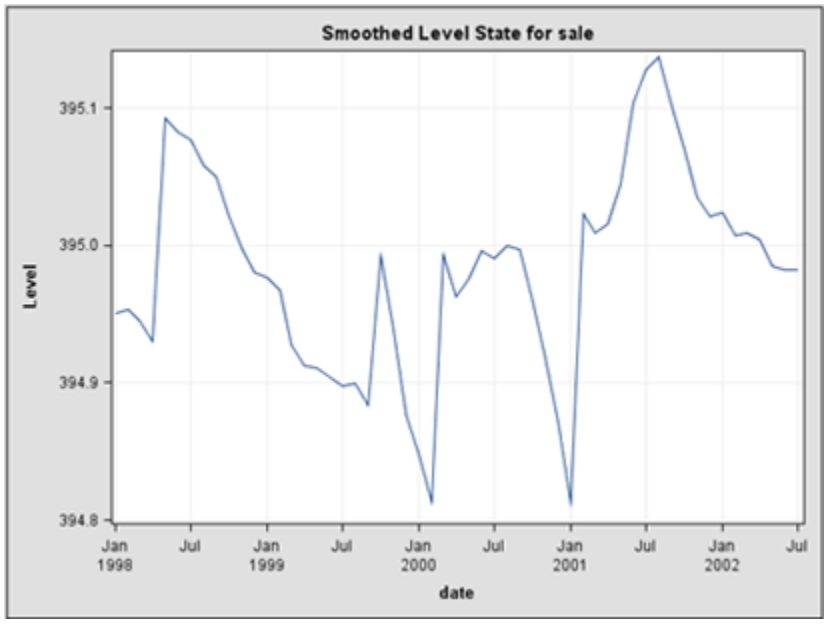
Variable Information	
Name	sale
Label	Unit Sale
First	JAN1998
Last	DEC2002
Number of Observations Read	60

Trend Statistics for Variable sale

Obs	Time	Number of Nonmissing	Minimum	Maximum	Sum	Mean	Standard Deviation
1	JAN1998	1	355.0000	355.0000	355.00000	355.0000	.
2	FEB1998	1	398.0000	398.0000	398.00000	398.0000	.
3	MAR1998	1	387.0000	387.0000	387.00000	387.0000	.
4	APR1998	1	380.0000	380.0000	380.00000	380.0000	.

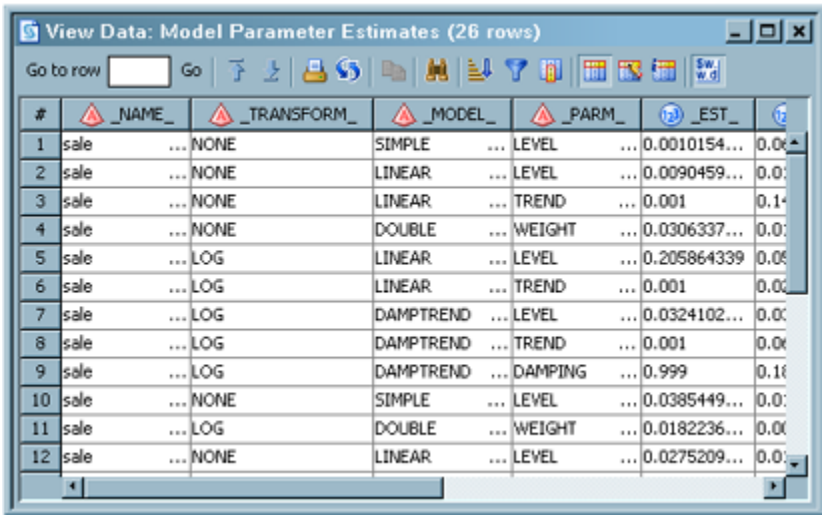
The following display shows a sample graphical plot:

Display 18.23 *Sample Forecast Plot*



3. Right-click the first target table on the **Diagram** tab, and click **Open** in the pop-up menu. The following display shows the data for the target table in the View Data window:

Display 18.24 *Sample Target Table Data*



#	_NAME_	_TRANSFORM_	_MODEL_	_PARM_	_EST_	
1	sale	NONE	SIMPLE	LEVEL	0.0010154...	0.0
2	sale	NONE	LINEAR	LEVEL	0.0090459...	0.0
3	sale	NONE	LINEAR	TREND	0.001	0.1
4	sale	NONE	DOUBLE	WEIGHT	0.0306337...	0.0
5	sale	LOG	LINEAR	LEVEL	0.205864339	0.0
6	sale	LOG	LINEAR	TREND	0.001	0.0
7	sale	LOG	DAMPTREND	LEVEL	0.0324102...	0.0
8	sale	LOG	DAMPTREND	TREND	0.001	0.0
9	sale	LOG	DAMPTREND	DAMPING	0.999	0.1
10	sale	NONE	SIMPLE	LEVEL	0.0385449...	0.0
11	sale	LOG	DOUBLE	WEIGHT	0.0182236...	0.0
12	sale	NONE	LINEAR	LEVEL	0.0275209...	0.0

Note: The target tables in the sample job are temporary output tables that are not preserved when the SAS session is ended. If you need permanent target tables, right-click the target tables and click **Replace** in the pop-up menu.

4. Repeat the process for the other target tables in your forecast.

Frequency of Eye Color By Hair Color Crosstabulation

Overview

Use the Frequency transformations to produce one-way to n-way frequency and contingency (crosstabulation) tables. The Frequency transformations are based on the FREQ procedure, which generates frequency statistics. For more information about this procedure, see "The FREQ Procedure" section in *Base SAS Procedures Guide*.

There are two Frequency transformations: Frequency and One-Way Frequency. The Frequency transformation uses PROC FREQ to compute statistics for complex tests, measures of association, and stratified analysis of one-way to *n*-way tables. The One-Way Frequency transformation is used for simpler PROC FREQ analysis on one-way tables to examine the relationship between two classification variables. It can also be used to compute statistics for equal proportions, specified proportions, or the binomial proportion. The One-Way Frequency transformation also has a subset of the options available for the Frequency transformation.

Both Frequency transformations control many aspects of the analysis, including the following:

- grouping of rows by the values in one or more columns
- how the rows appear in the report
- which column or columns are analyzed

You can use the Frequency transformations to generate frequency statistics in a target and on the **Output** tab of the Job Editor. ODS output in the form of HTML, PDF, or

RTF can be sent to a folder on the SAS Application Server that executes the job. ODS output can also be sent to any folder with access to that SAS Application Server.

The target receives data only for the source columns that are involved in the analysis. The target requires two columns that either Frequency transformation populates: **Count** receives the total number of occurrences in a category, and **Percent** receives the percentages for each category.

You can specify grouping columns in the Frequency transformations. When you do this, a SAS BY statement orders target rows according to the values in the grouping columns. The Frequency transformations require that grouping columns be sorted in ascending order in the source. If you specify grouping columns, you can sort those columns before the Frequency transformation using a SAS Sort transformation.

For examples of how you can use the Frequency transformations, see the Frequency of Eye Color By Hair Color Crosstabulation at "Frequency of Eye Color By Hair Color Crosstabulation" on page 385 and the One-Way Frequency transformation example at "One-Way Frequency of Eye Color By Region" on page 398.

Problem

You want to generate frequency statistics.

Solution

You can use the Frequency transformation in a SAS Data Integration Studio job to produce one-way to n-way frequency and contingency (crosstabulation) tables. For example, you can create a job similar to the sample job featured in this topic. This sample job generates a list of the numbers of individuals with particular combinations of hair and eye color by geographical region. The frequency statistics are sent to a target and to the **Output** tab in the Job Editor window. The sample job includes the following tasks:

- "Create and Populate the Job" on page 386
- "Configure Analytical Options" on page 387
- "Configure Reporting Options" on page 393
- "Run the Job and View the Output" on page 394

Tasks

Create and Populate the Job

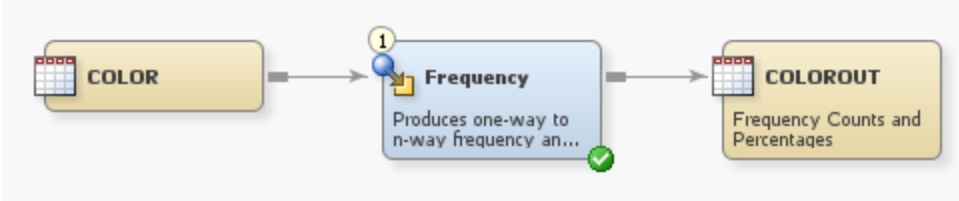
Perform the following steps to create and populate the job:

1. Create an empty SAS Data Integration Studio job.
2. From the **Analysis** folder in the Transformations tree, select and drag a Frequency 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 Frequency transformation on the **Diagram** tab.
4. Drag the cursor from the source table to the input port of the Frequency transformation. This action connects the transformation to the source.

5. Right-click the Frequency transformation, and click **Add Output Port** from the **Ports** option in the drop-down menu. This step enables you to add an output port to the transformation.
6. Select and drag the source table from the Inventory tree. Then, drop it after the Frequency transformation on the **Diagram** tab.
7. Drag the cursor from the Frequency transformation output port to the target table. This action connects the target to the transformation.

The following display shows a sample process flow diagram for a job that contains the Frequency transformation:

Display 18.25 *Sample Process Flow*



Note that the source table for the sample job is named COLOR, and the target table is named COLOROUT.

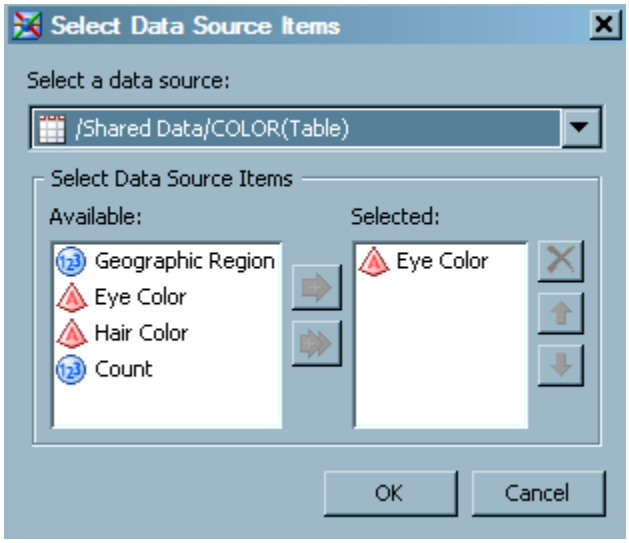
Configure Analytical Options

Use the **Options** tab in the properties window for the Frequency transformation to configure the output for your analysis. Note that the **Options** tab is divided into two parts, with a list of categories on the left side and the options for the selected category on the right side.

Perform the following steps to set the options that you need for your job:

1. In the **Mappings** tab, add the column Eye Color to the target table.
2. In the **Diagram** tab of the Job Editor window, open the properties window for the Frequency transformation. Then, click the **Options** tab.
3. Click **Assign columns** to access the Assign columns page. Use the column selection prompts to access the columns that you need for your job. For example, you can click  beside the **Select columns for frequency distribution** field to access the Select Data Source Items window, as shown in the following display:

Display 18.26 *Sample Select Data Source Items Window*



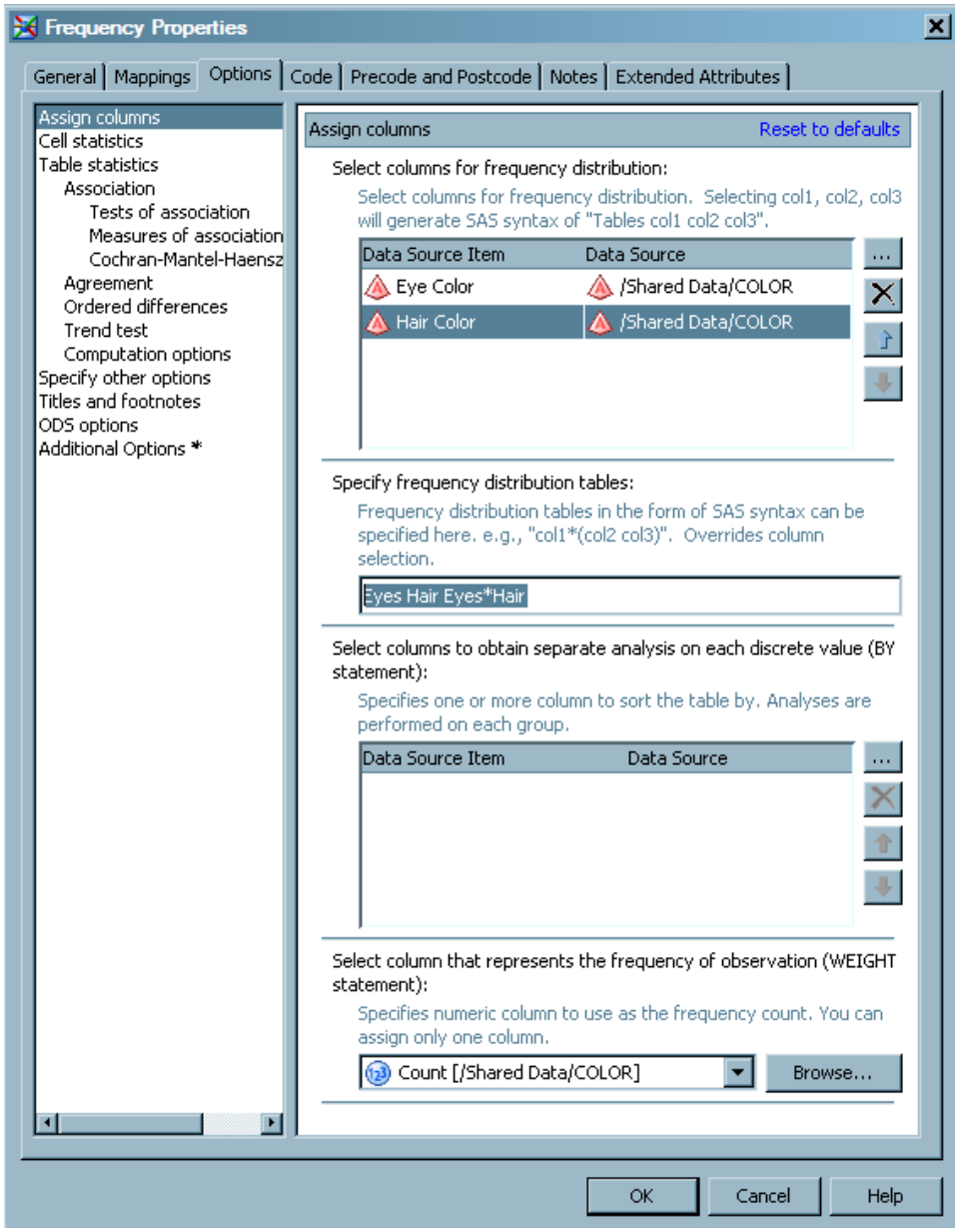
In the sample job, the following column options are set in the Assign columns window:

- In the **Select columns for frequency distribution** field, select the values of **Eye Color** and **Hair Color**.
- To create a crosstabulation table, enter the value of *Eyes Hair Eyes*Hair* in the **Select frequency distribution tables** field. The *Eyes*Hair* specification produces a crosstabulation table with eye color defining the table rows and hair color defining the table columns.

  Note: Any entry in the **Select frequency distribution tables** field overrides the values in the **Select columns for frequency distribution** field.
- In the **Select column that represents the frequency of observation (WEIGHT statement)** field, select **Count**.

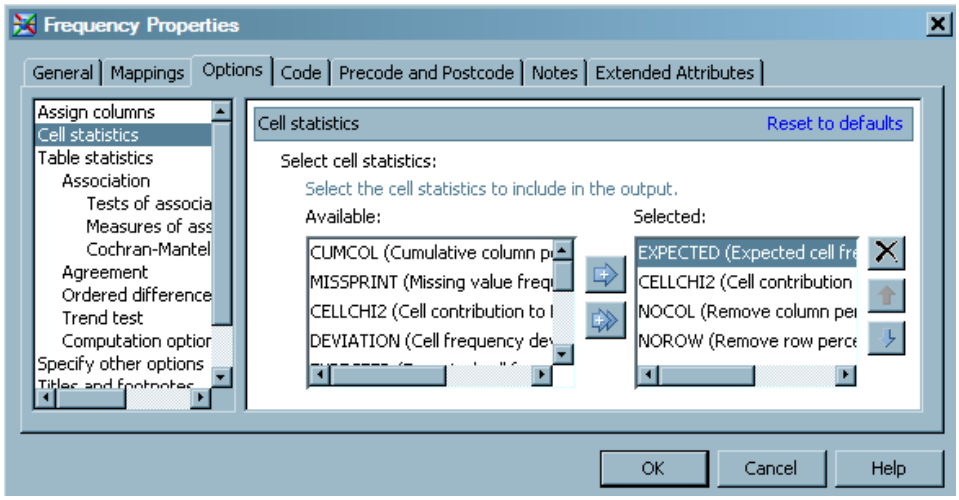
These fields are shown in the following display:

Display 18.27 *Frequency Column Options*



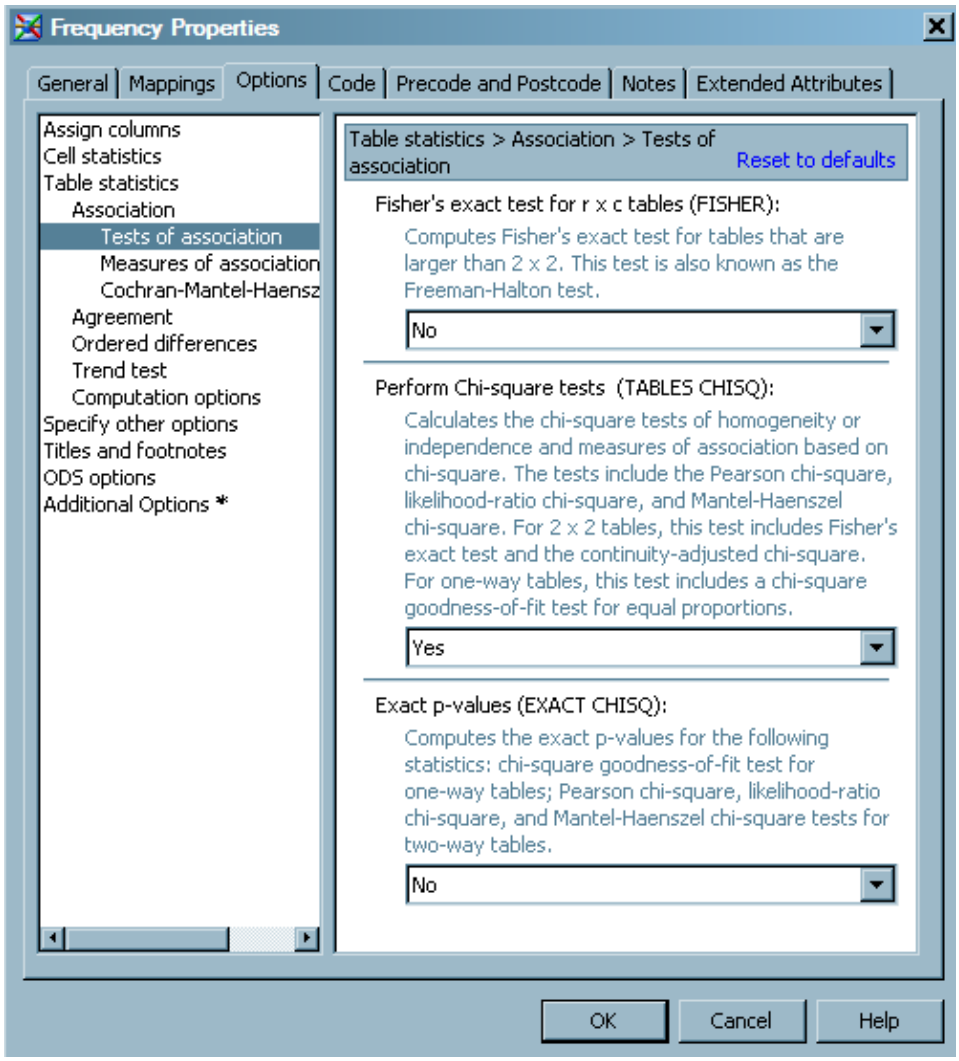
4. Set the Cell statistics to include in the output. In this example, the CHISQ option is used to produce chi-square tests. The selected cell statistics include the EXPECTED option, which displays expected cell frequencies in the table, and the CELLCHI2 option, which displays the cell contribution to the chi-square. The NOROW and NOCOL options suppress the display of row and column percentages in the table. These items are selected as shown in the following display:

Display 18.28 *Cell statistics*

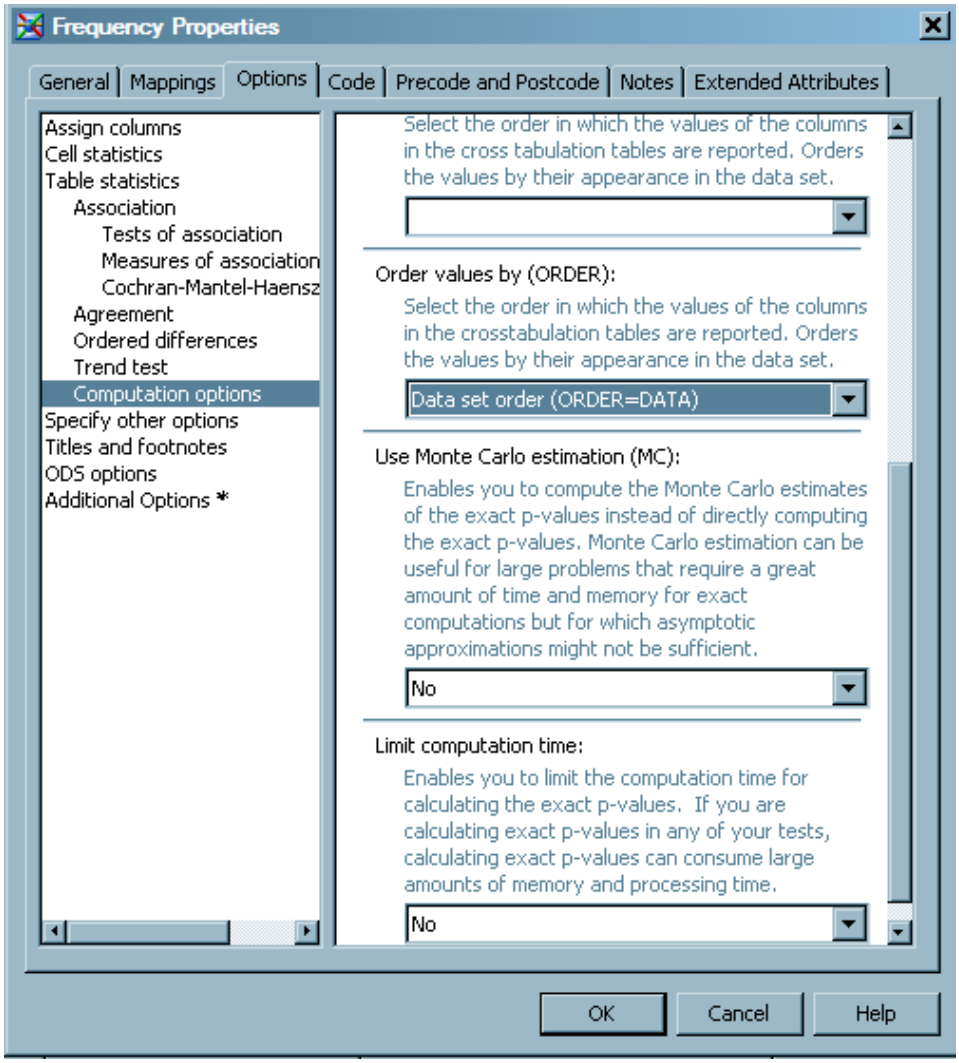


5. Set the options for the Table statistics in the appropriate fields. For this example, the settings for **Perform Chi-square tests (TABLES CHISQ)** and **Order by values (ORDER)** are set in the windows as shown in the following displays:

Display 18.29 *Table Statistics Example*



Display 18.30 *Computation Options*

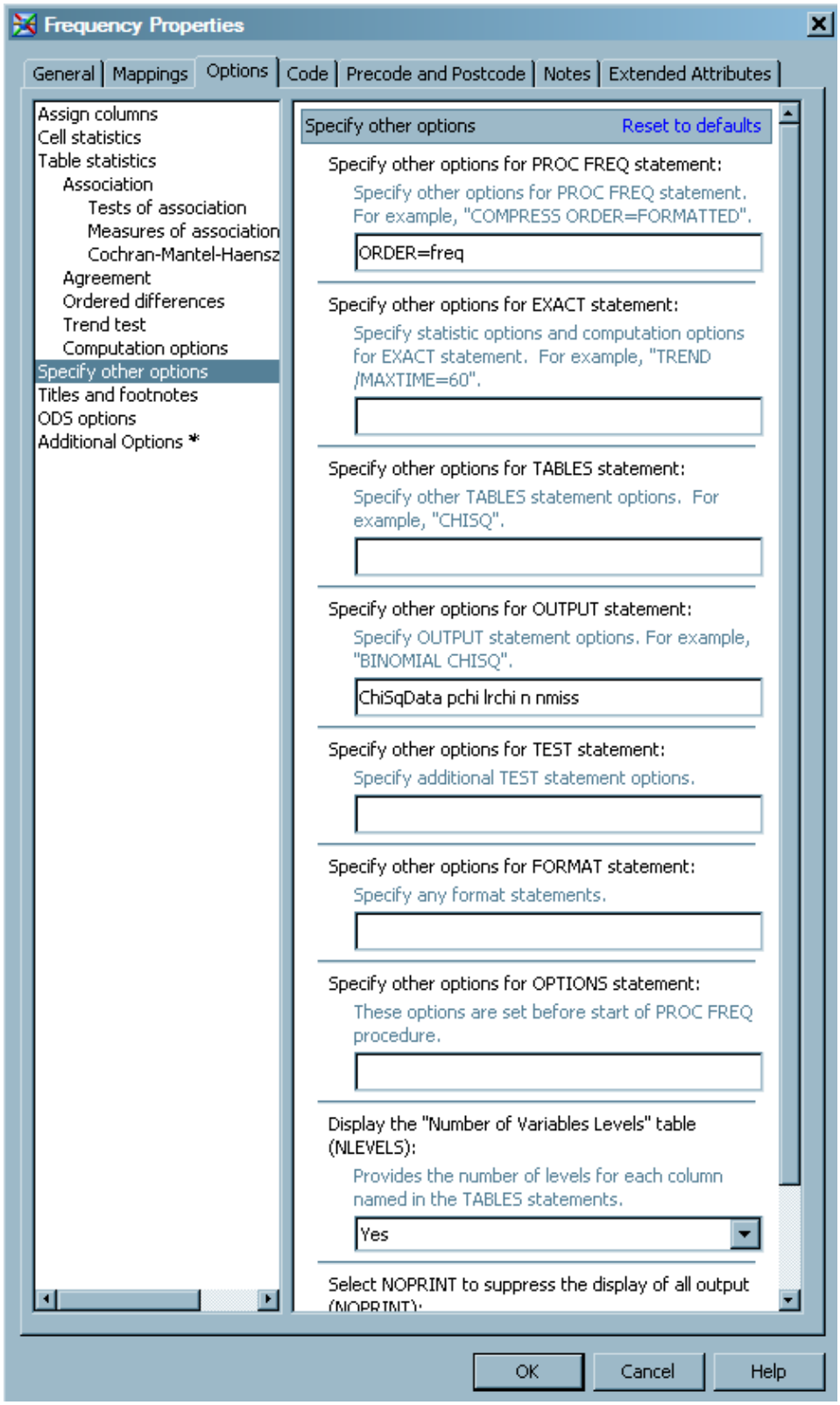


6. Set the options for your analysis in the appropriate fields. Note that these frequency options are set for the sample job in the Specify other options window:

   - Enter a value of *ORDER=FREQ* in the **Specify other options for PROC FREQ statement** field.
   - Enter a value of *ChiSqData pchi lrchi n nmiss* in the **Specify other options for OUTPUT statement** field. The OUTPUT statement creates the ChiSqData data set with eight variables: the N option stores the number of non-missing observations; the NMISS option stores the number of missing observations; and the PCHI and LRCHI options store Pearson and likelihood-ratio chi-square statistics, respectively, together with their degrees of freedom and p-values.
   - Select a value of **Yes** in the **Display the "Number of Variables Levels" table (NLEVELS)** field.

   These fields are shown in the following display:

Display 18.31 *Frequency Options*



Note: In the sample job, the COLOR source table is already sorted in ascending order according to the values of the Geographical Region column. The Frequency transformation requires sorting by grouping columns. If COLOR is not sorted

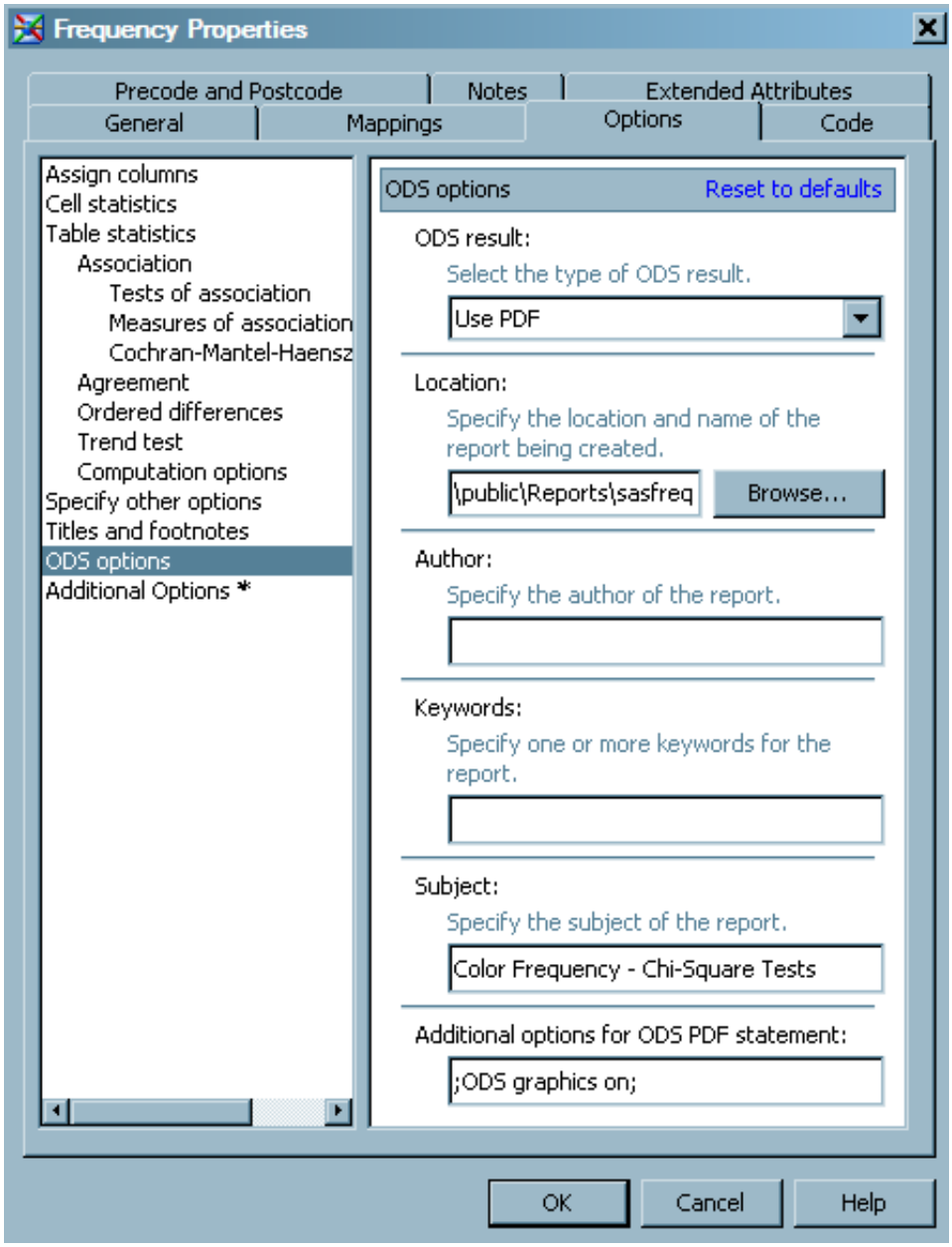
appropriately, then a SAS Sort transformation can be added to the job before the Frequency transformation.

Configure Reporting Options

Use the remaining option pages to create and save a report based on the analysis conducted in the job. Perform the following steps to set the reporting options:

1. Click **Titles and footnotes** to access the Titles and footnotes page and enter up to three headings and two footnotes.
2. Click **ODS options** to access the ODS options page. You can choose between HTML, RTF, and PDF output and enter appropriate settings for each. The sample job uses PDF output. When **Use PDF** is selected in the **ODS Result** field, new fields are displayed. These include **Location**, **Author**, **Keywords**, **Subject**, and **Additional options for ODS PDF statement**. (The path specified in the **Location** field is relative to the SAS Application Server that executes the job.) These fields are shown in the following display:

Display 18.32 *Sample ODS Options*

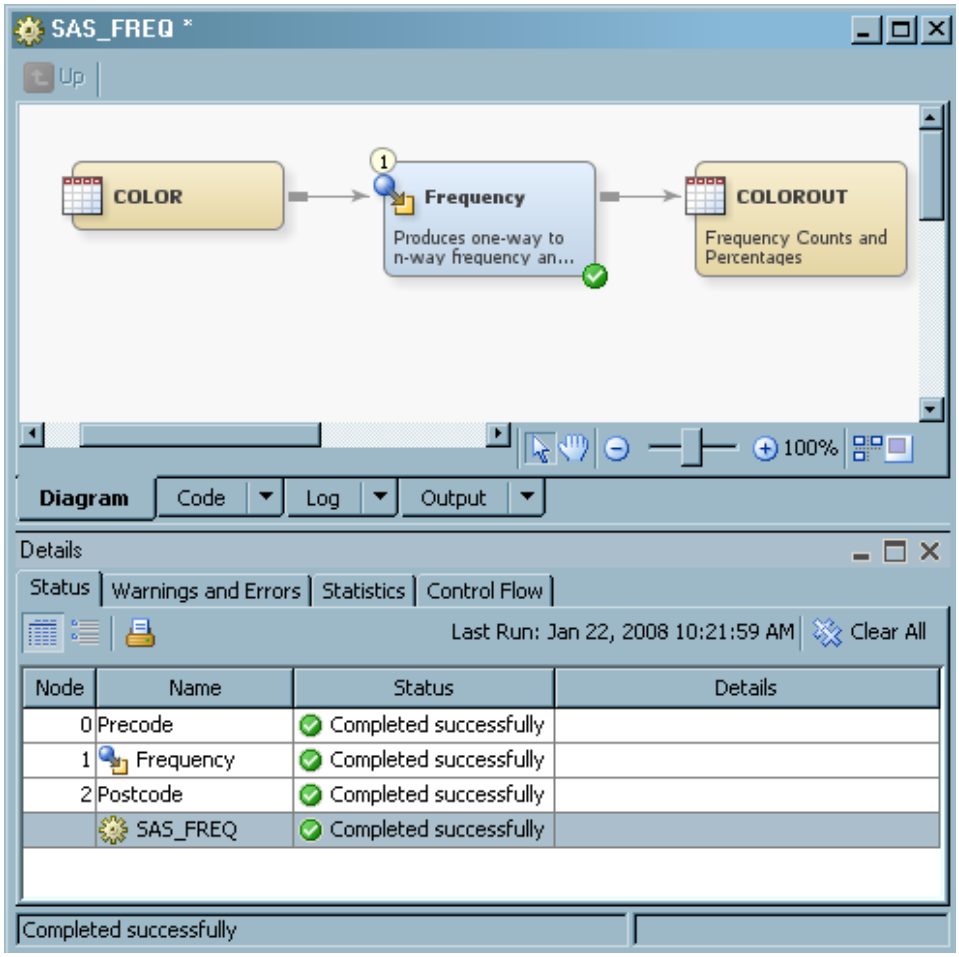


Run the Job and View the Output

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. The following display shows a successful run of a sample job:

Display 18.33 *Successfully Completed Sample Job*



2. If error messages are displayed on the **Status** tab, read and respond to the messages as needed.

3. To view the frequency analysis, click the **Output** tab in the Job Editor window. The following display shows the analysis for the sample job:

Display 18.34 *Sample Output in the Output Tab*

The FREQ Procedure

Number of Variable Levels

Variable	Label	Levels
Eyes	Eye Color	3
Hair	Hair Color	5

Eye Color

Eyes	Frequency	Percent	Cumulative Frequency	Cumulative Percent
blue	222	29.13	222	29.13
green	199	26.12	421	55.25
brown	341	44.75	762	100.00

Chi-Square Test for Equal Proportions

Chi-Square	45.7402
DF	2
Pr > ChiSq	<.0001

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 18.35 *Sample Target Table Data*

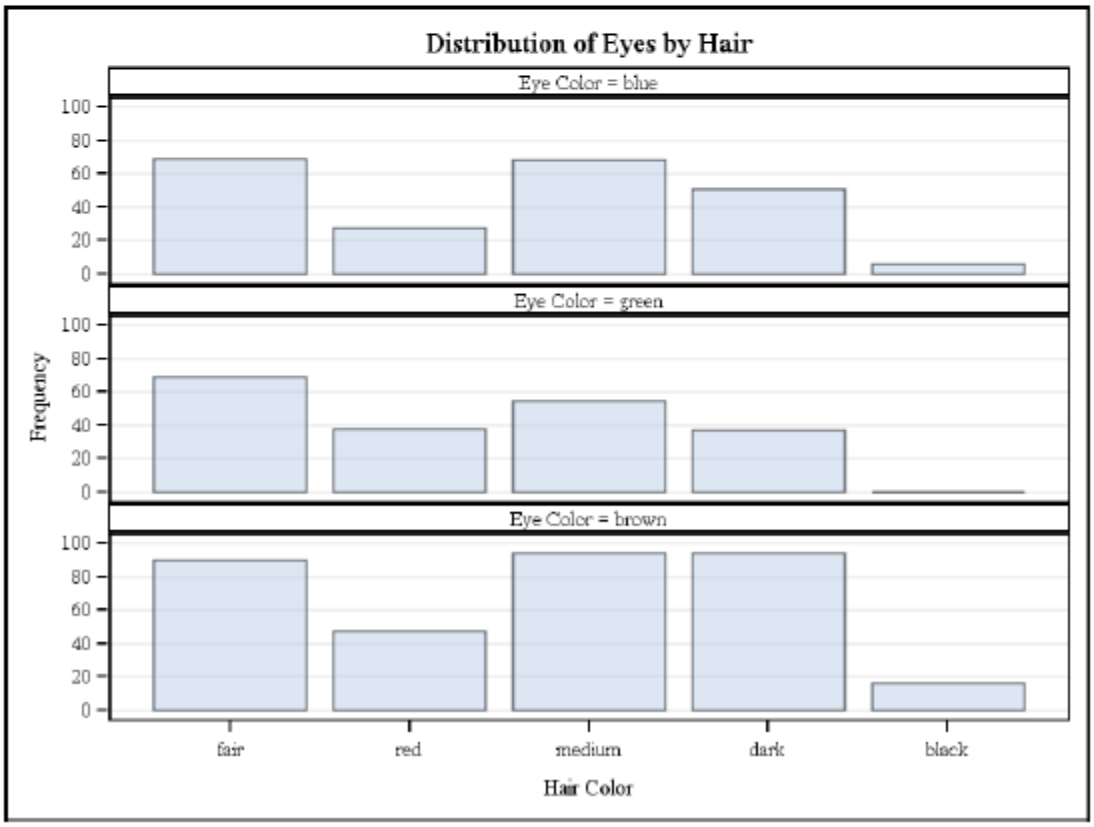
#	Region	Eyes	Count	PERCENT
1	1	brown	123	50
2	1	blue	65	26.422764228
3	1	green	58	23.577235772
4	2	brown	218	42.248062016
5	2	blue	157	30.426356589
6	2	green	141	27.325581395

5. Open the PDF document that you created and saved earlier. A portion displays the same as the One-Way Frequency example. The following display illustrates a sample report based on the frequency data that is not available to the One-Way Frequency:

Display 18.36 *Sample PDF Output*

The FREQ Procedure

Frequency Expected Cell Chi-Square Percent	Table of Eyes by Hair					
Eyes(Eye Color)	Hair(Hair Color)					
	fair	red	medium	dark	black	Total
blue	69 66.425 0.0998 9.06	28 32.921 0.7357 3.67	68 63.22 0.3613 8.92	51 53.024 0.0772 6.69	6 6.4094 0.0262 0.79	222 29.13
green	69 59.543 1.5019 9.06	38 29.51 2.4422 4.99	55 56.671 0.0492 7.22	37 47.53 2.3329 4.86	0 5.7454 5.7454 0.00	199 26.12
brown	90 102.03 1.4187 11.81	47 50.568 0.2518 6.17	94 97.109 0.0995 12.34	94 81.446 1.935 12.34	16 9.8451 3.8478 2.10	341 44.75
Total	228 29.92	113 14.83	217 28.48	182 23.88	22 2.89	762 100.00



One-Way Frequency of Eye Color By Region

Overview

Use the Frequency transformations to produce one-way to n-way frequency and contingency (crosstabulation) tables. The Frequency transformations are based on the FREQ procedure, which generates frequency statistics. For more information about this procedure, see "The FREQ Procedure" section in *Base SAS Procedures Guide*.

There are two Frequency transformations: Frequency and One-Way Frequency. The Frequency transformation uses PROC FREQ to compute statistics for complex tests, measures of association, and stratified analysis of one-way to *n*-way tables. The One-Way Frequency transformation is used for simpler PROC FREQ analysis on one-way tables to examine the relationship between two classification variables. It can also be used to compute statistics for equal proportions, specified proportions, or the binomial proportion. The One-Way Frequency transformation also has a subset of the options available for the Frequency transformation.

Both Frequency transformations control many aspects of the analysis, including the following:

- grouping of rows by the values in one or more columns
- how the rows appear in the report
- which column or columns are analyzed

You can use the Frequency transformations to generate frequency statistics in a target and on the **Output** tab of the Job Editor. ODS output in the form of HTML, PDF, or RTF can be sent to a folder on the SAS Application Server that executes the job. ODS output can also be sent to any folder with access to that SAS Application Server.

The target receives data only for the source columns that are involved in the analysis. The target requires two columns that either Frequency transformation populates: **Count** receives the total number of occurrences in a category, and **Percent** receives the percentages for each category.

You can specify grouping columns in the Frequency transformations. When you do this, a SAS BY statement orders target rows according to the values in the grouping columns. The Frequency transformations require that grouping columns be sorted in ascending order in the source. If you specify grouping columns, you can sort those columns before the Frequency transformation using a SAS Sort transformation.

For examples of how you can use the Frequency transformations, see the Frequency of Eye Color By Hair Color Crosstabulation at "Frequency of Eye Color By Hair Color Crosstabulation" on page 385 and the One-Way Frequency transformation example at "One-Way Frequency of Eye Color By Region" on page 398.

Problem

You want to generate simple frequency statistics.

Solution

You can use the One-Way Frequency transformation in a SAS Data Integration Studio job to produce one-way frequency and crosstabulation (contingency) tables. For example, you can create a job similar to the sample job featured in this topic. This sample job generates a list of the numbers of individuals with particular combinations of hair and eye color by geographical region. The frequency statistics are sent to a target and to the **Output** tab in the Job Editor window. The sample job includes the following tasks:

- "Create and Populate the Job" on page 399
- "Configure Analytical Options" on page 400
- "Configure Reporting Options" on page 402
- "Run the Job and View the Output" on page 403

Tasks

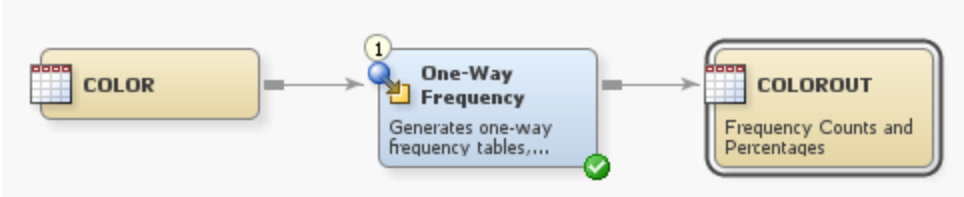
Create and Populate the Job

Perform the following steps to create and populate the job:

1. Create an empty SAS Data Integration Studio job.
2. Select and drag a One-Way Frequency 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.
3. Select and drag the source table from its folder and drop it before the One-Way Frequency transformation on the **Diagram** tab.
4. Drag the cursor from the source table to the input port of the One-Way Frequency transformation. This action connects the source to the transformation.
5. Right-click the One-Way Frequency transformation, and click **Add Output Port** from the **Ports** option in the drop-down menu. This step enables you to add an output port to the transformation.
6. Select and drag the source table from the Inventory tree. Then, drop it after the One-Way Frequency transformation on the **Diagram** tab.
7. Drag the cursor from the One-Way Frequency transformation output port to the target table. This action connects the target to the transformation.

The following display shows a sample process flow diagram for a job that contains the One-Way Frequency transformation:

Display 18.37 *Sample Process Flow*



Note that the source table for the sample job is named COLOR, and the target table is named COLOROUT.

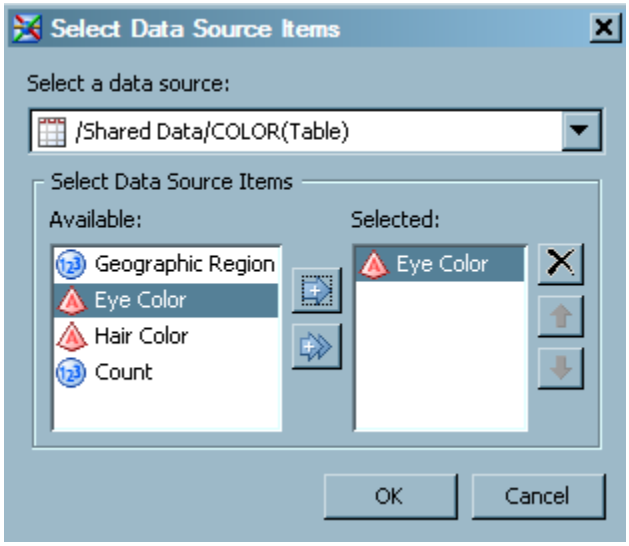
Configure Analytical Options

Use the **Options** tab in the properties window for the One-Way Frequency transformation to configure the output for your analysis. Note that the **Options** tab is divided into two parts, with a list of categories on the left side and the options for the selected category on the right side.

Perform the following steps to set the options that you need for your job:

1. In the **Mappings** tab, add the column Eye Color to the target table.
2. In the **Diagram** tab of the Job Editor window, open the properties window for the One-Way Frequency transformation. Then, click the **Options** tab.
3. Click **Assign columns** to access the Assign columns page. Use the column selection prompts to access the columns that you need for your job. For example, you can click ... beside the **Select columns for frequency distribution** field to access the Select Data Source Items window, as shown in the following display:

Display 18.38 *Sample Select Data Source Items Window*

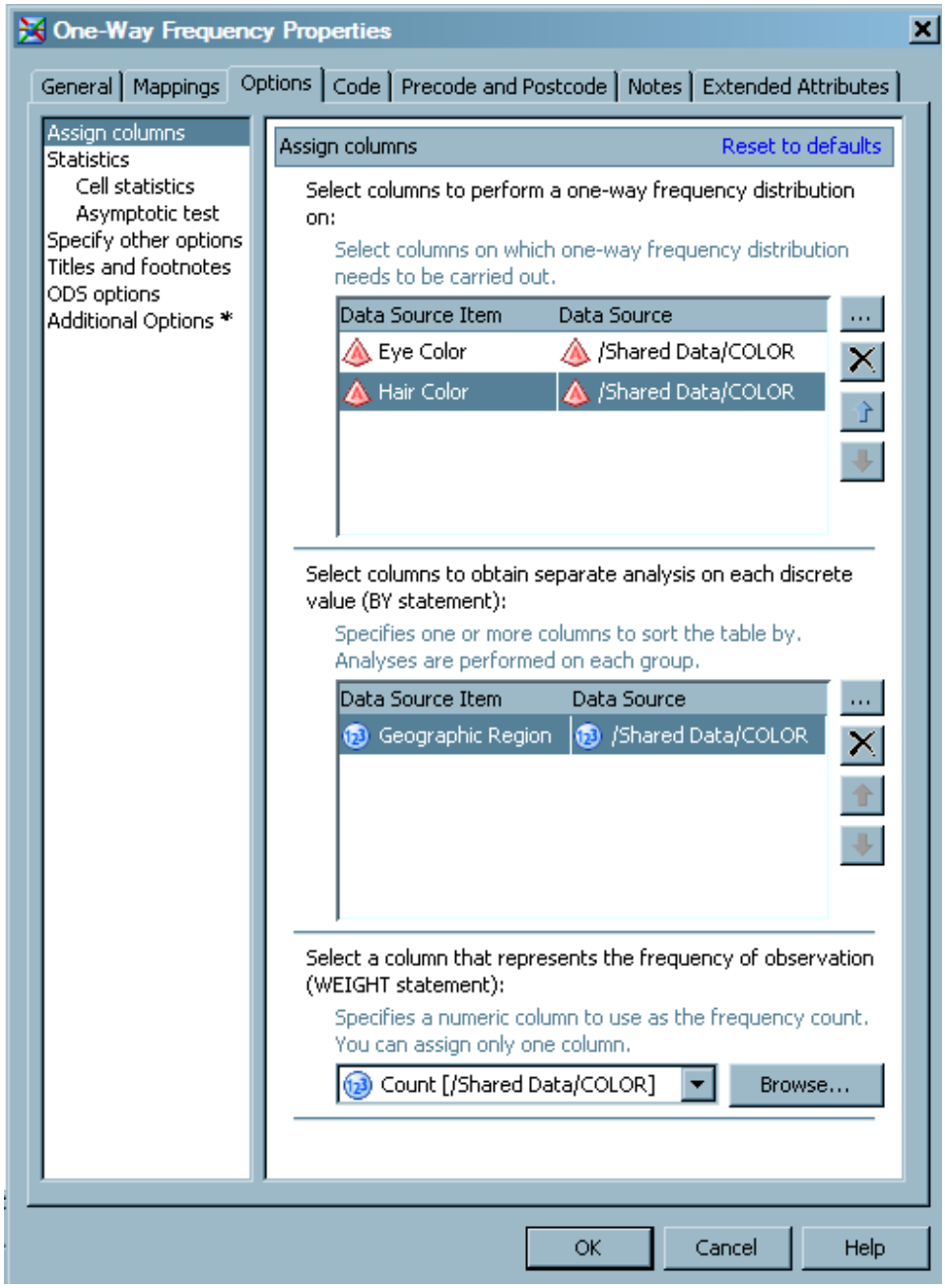


In the sample job, the following column options are set in the Assign columns window:

- In the **Select columns for frequency distribution** field, select the values of **Eye Color** and **Hair Color**.
- In the **Select columns to obtain separate analysis on each discrete value (BY statement)** field, select **Geographic Region**.
- In the **Select column that represents the frequency of observation (WEIGHT statement)** field, select **Count**.

These fields are shown in the following display:

Display 18.39 *Frequency Column Options*

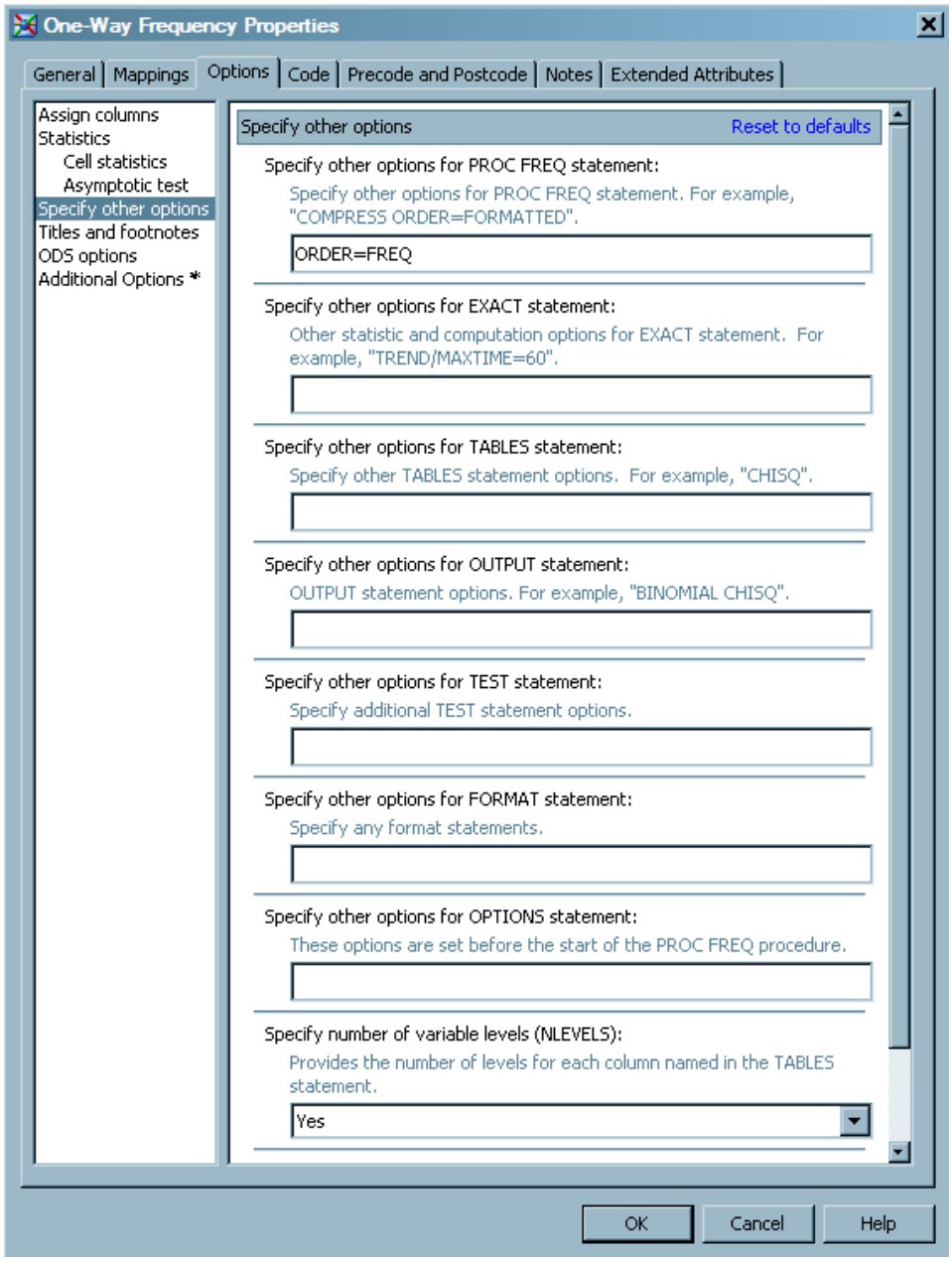


4. Set the options for your analysis in the appropriate fields. Note that these frequency options are set for the sample job:

   - In the Specify other options window, enter a value of *ORDER=FREQ* in the **Specify other options for PROC FREQ statement** field.
   - In the Specify other options window, select a value of *Yes* in the **Specify number of variables levels (NLEVELS)** field.

   These fields are shown in the following display:

Display 18.40 *One-Way Frequency Options*



Note: In the sample job, the COLOR source table is already sorted in ascending order according to the values of the Geographical Region column. The One-Way Frequency transformation requires sorting by grouping columns. If COLOR is not sorted appropriately, then a SAS Sort transformation can be added to the job before the Frequency transformation.

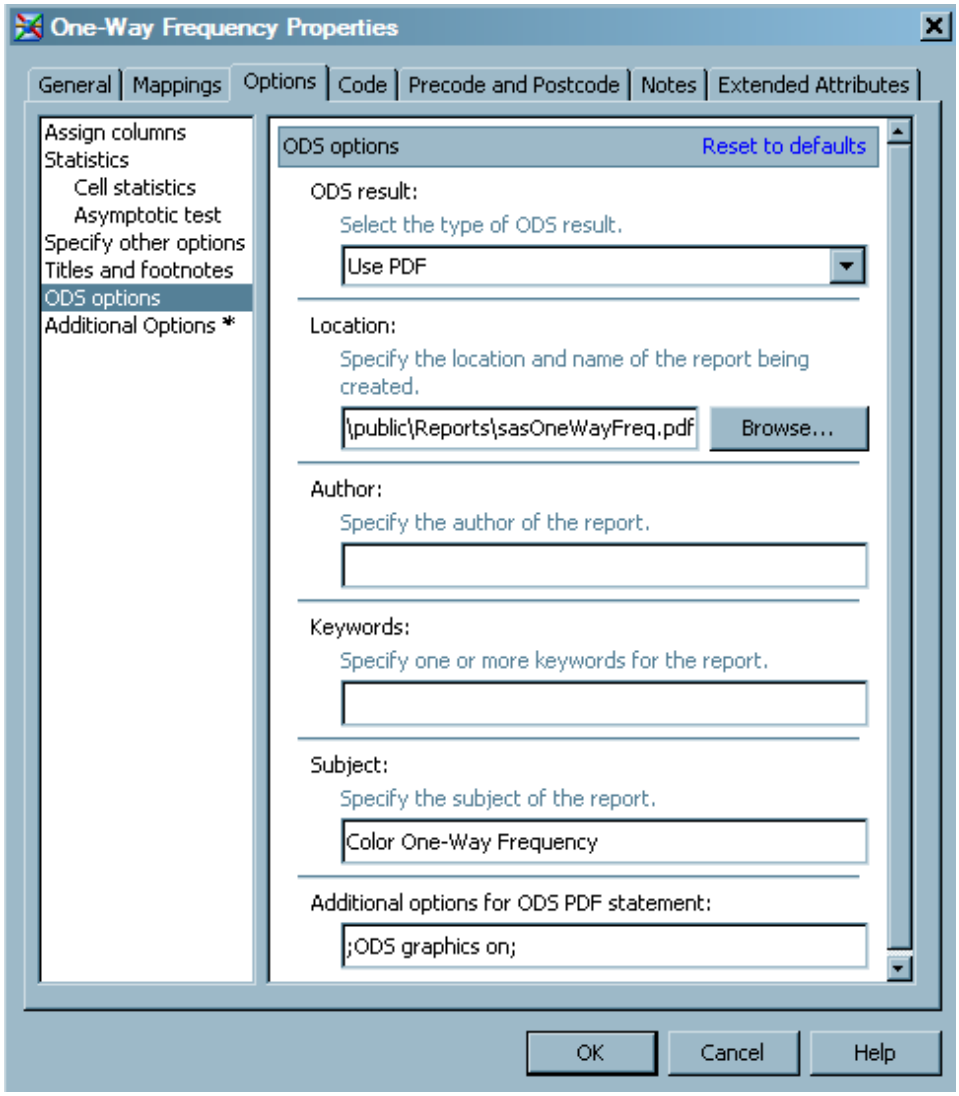
Configure Reporting Options

Use the remaining option pages to create and save a report based on the analysis conducted in the job. Perform the following steps to set the reporting options:

1. Click **Titles and footnotes** to access the Titles and footnotes page and enter up to three headings and two footnotes.

2. Click **ODS options** to access the ODS options page. You can choose between HTML, RTF, and PDF output and enter appropriate settings for each. The sample job uses PDF output. When **Use PDF** is selected in the **ODS result** field, new fields are displayed. These include **Location**, **Author**, **Keywords**, **Subject**, and **Additional options for ODS PDF statement**. (The path specified in the **Location** field is relative to the SAS Application Server that executes the job.) These fields are shown in the following display:

Display 18.41 *Sample ODS Options*

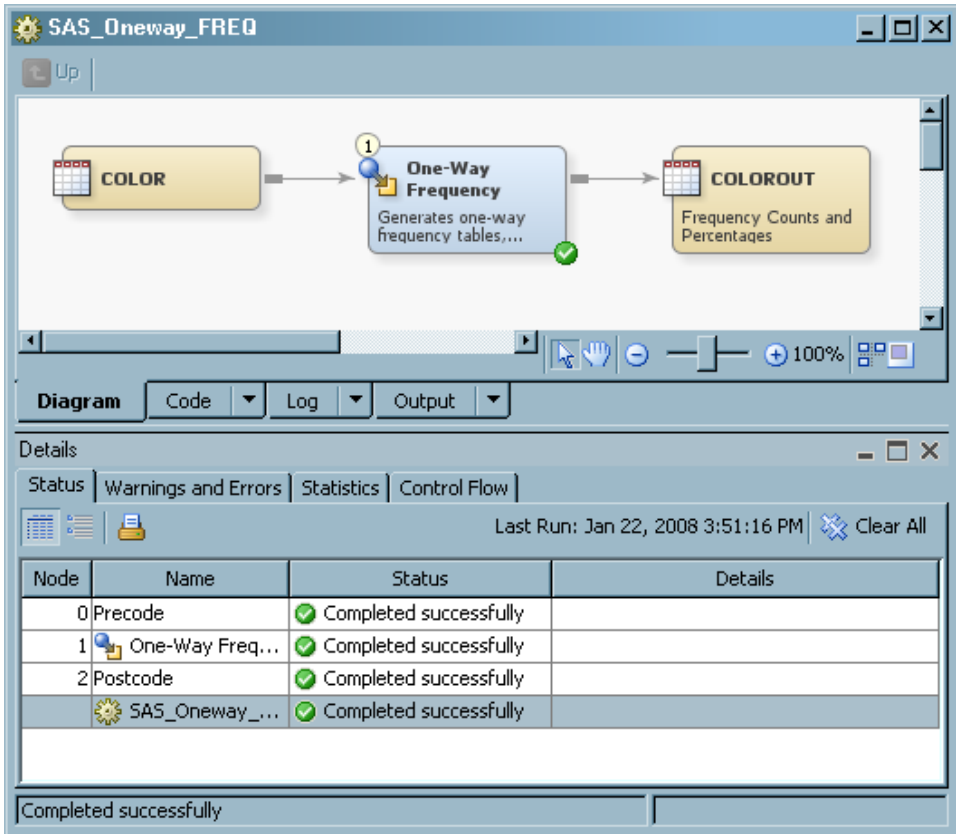


Run the Job and View the Output

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. The following display shows a successful run of a sample job:

Display 18.42 *Successfully Completed Sample Job*



2. If error messages are displayed on the **Status** tab, read and respond to the messages as needed.

3. To view the frequency analysis, click the **Output** tab in the Job Editor window. The following display shows the analysis for the sample job:

Display 18.43 *Sample Output in the Output Tab*

The FREQ Procedure

Number of Variable Levels

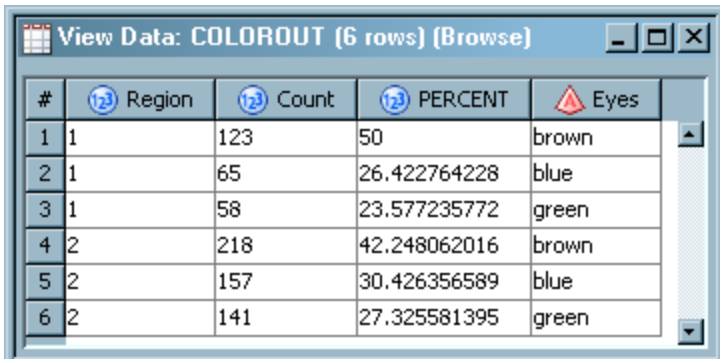
Variable	Label	Levels
Eyes	Eye Color	3

Eye Color

Eyes	Frequency	Percent	Cumulative Frequency	Cumulative Percent
brown	218	42.25	218	42.25
blue	157	30.43	375	72.67
green	141	27.33	516	100.00

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 18.44 *Sample Target Table Data*



View Data: COLOROUT (6 rows) (Browse)

#	Region	Count	PERCENT	Eyes
1	1	123	50	brown
2	1	65	26.422764228	blue
3	1	58	23.577235772	green
4	2	218	42.248062016	brown
5	2	157	30.426356589	blue
6	2	141	27.325581395	green

5. Open the PDF document that you created and saved earlier. The following display illustrates a sample report based on the frequency data:

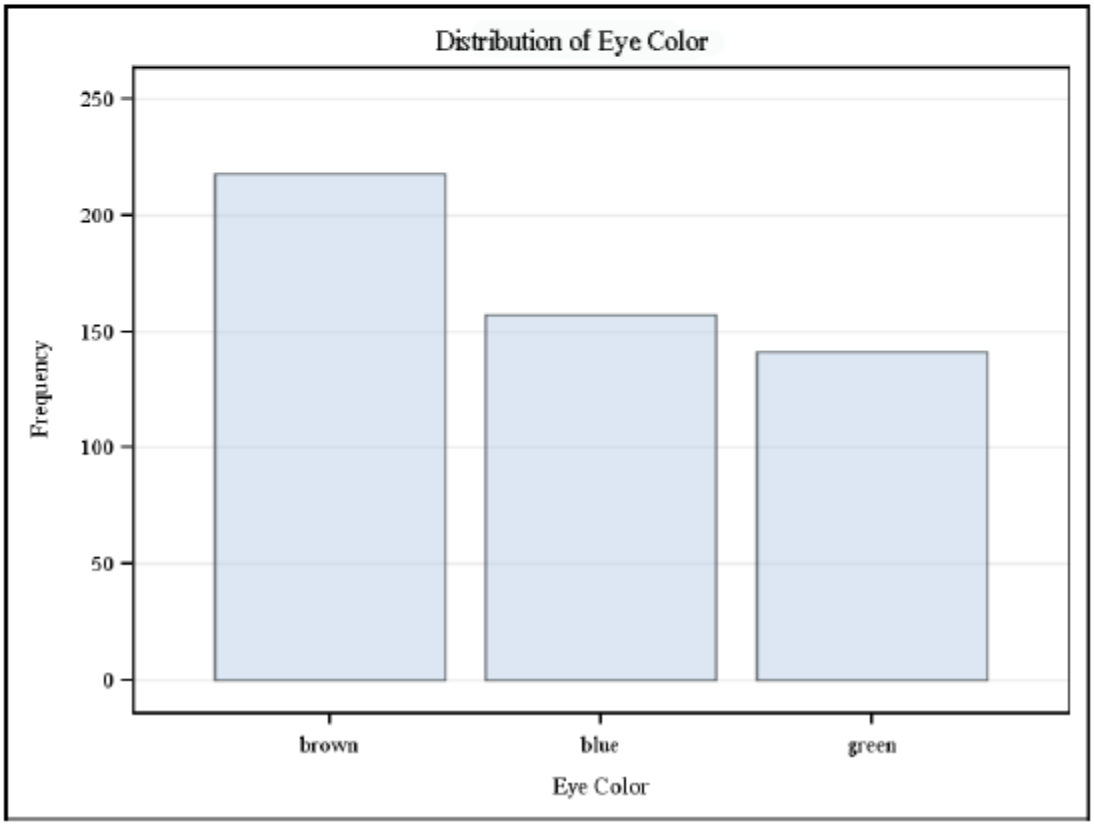
Display 18.45 *Sample PDF Output*

The FREQ Procedure

Geographic Region=2

Number of Variable Levels		
Variable	Label	Levels
Eyes	Eye Color	3
Hair	Hair Color	5

Eye Color				
Eyes	Frequency	Percent	Cumulative Frequency	Cumulative Percent
brown	218	42.25	218	42.25
blue	157	30.43	375	72.67
green	141	27.33	516	100.00



Creating Summary Statistics for a Table

Overview

The Summary Statistics transformation provides an interface to the MEANS procedure. The MEANS procedure provides data summarization tools to perform the following tasks:

- compute descriptive statistics for variables across all observations and within groups of observations
- calculate descriptive statistics based on moments
- estimate quantiles, which includes the median
- calculate confidence limits for the mean
- identify extreme values
- perform a t test

By default, the MEANS procedure displays output. You can also use the OUTPUT statement to store the statistics in a SAS data set. You can use the MEANS procedure to generate a statistical summary. Data is sent to a target table and to the **Output** tab of the Job Editor. You can also create ODS output.

You can control many aspects of how the target table is created, including the following:

- the type of analysis
- analysis options
- which columns are analyzed

The target table receives data only for the columns that are involved in the analysis. The target requires three columns that the Summary Statistics transformation populates:

TYPE
contains the type of statistic.

FREQ
contains the frequency.

STAT
contains the name of the statistic.

You can specify grouping columns in the Summary Statistics transformation. Doing so causes a SAS BY statement to order target rows according to the values in the grouping columns. The Summary Statistics transformation requires that grouping columns be sorted in ascending order in the source. If you specify grouping columns, you can sort those columns before the Summary Statistics transformation using a SAS Sort transformation.

Problem

You want to generate summary statistics for a table.

Solution

You can use the Summary Statistics transformation in a job that generates summary statistics and creates an ODS document that contains the results. This transformation uses the MEANS procedure to compute descriptive statistics for variables across all observations and within groups of observations. For example, you can create a job similar to the sample job featured in this topic. This sample job generates summary statistics from a source table that contains demographic data about a classroom of students. Note that the output for this job is sent to the **Output** tab in the Job Editor window and an ODS document that is configured in the job. The sample job includes the following tasks:

- "Create and Populate the Job" on page 408
- "Configure Analytical Options" on page 409
- "Configure Reporting Options" on page 410
- "Run the Job and View the Output" on page 411

Tasks

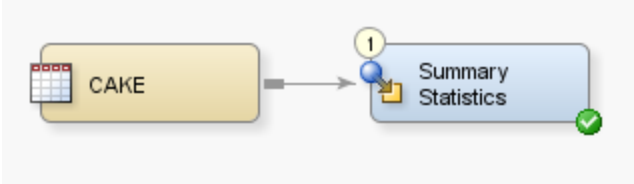
Create and Populate the Job

Perform the following steps to create and populate the job:

1. Create an empty SAS Data Integration Studio job.
2. Select and drag a Summary Statistics transformation from the **Analysis** folder in the Transformations tree. Then, drop it in the empty job on the **Diagram** tab in the Job Editor window.
3. Select and drag the source table out of the Inventory tree. Then, drop it before the Summary Statistics transformation on the **Diagram** tab.
4. Drag the cursor from the source table to the input port of the Summary Statistics transformation. This action connects the source to the transformation.
5. Right-click the Summary Statistics transformation, and click **Add Output Port** from the **Ports** option in the drop-down menu. This step enables you to add an output port to the transformation.
6. Select and drag the source table from the Inventory tree. Then, drop it after the Summary Statistics transformation on the **Diagram** tab.
7. Drag the cursor from the Summary Statistics transformation output port to the target table. This action connects the target to the transformation.

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

Display 18.46 *Sample Process Flow*



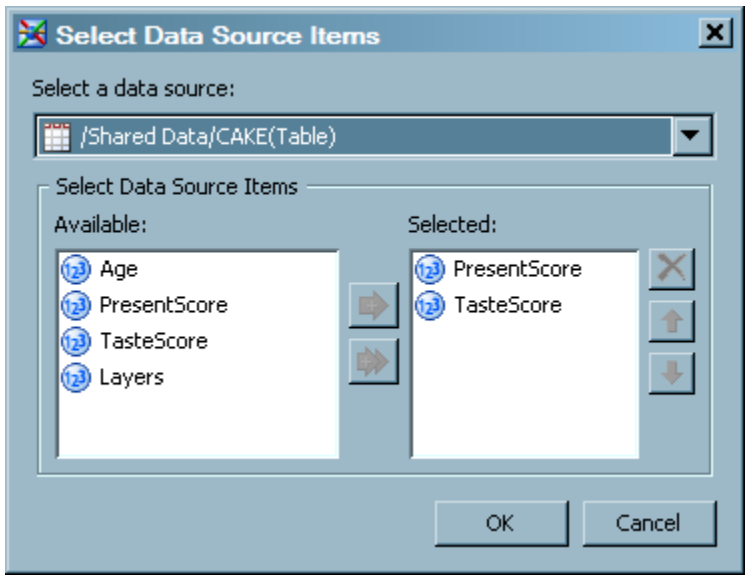
Note that the source table for the sample job is named CAKE.

Configure Analytical Options

Use the **Options** tab in the properties window for the Summary Statistics transformation to configure the SAS tables that are generated in the job and shape the output of your analysis. Note that the **Options** tab is divided into two parts, with a list of categories on the left-hand side and the options for the selected category on the right-hand side. Perform the following steps to set the options that you need for your job:

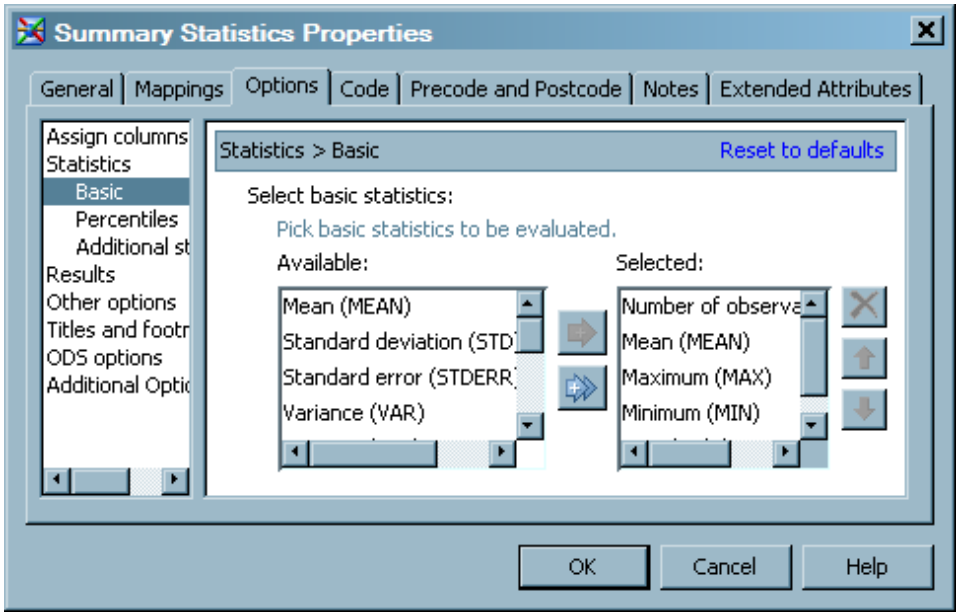
1. Open the properties window for the Summary Statistics transformation in the **Diagram** tab in the Job Editor window. Then, click the **Options** tab.
2. Click **Assign columns** to access the Assign columns page. Use the column selection prompts to access the columns that you need in the SAS tables generated in your job. For example, you can click ... for the **Select analysis columns (VAR statement)** to access the Select Data Source Items window, as shown in the following display.

   Display 18.47 *Sample Select Data Source Items Window*

   In the sample job, the VAR statement columns are PresentScore and TasteScore.
3. Click **Basic** to access the Statistics > Basic page to set the basic statistical options for the analysis conducted in the job. In the sample job, the **Number of observations (N)**, **Mean (MEAN)**, **Maximum (MAX)**, **Minimum (MIN)**, **Range (RANGE)**, and **Standard deviation (STD)** options are moved to the **Selected** field. The statistical options for the sample job are shown in the following display.

Display 18.48 *Sample Basic Statistical Options*



4. Set additional analytical options as needed. For example, the sample job uses a field width of eight, which limits the output width. This setting is made in the **Other PROC MEANS options** field on the Additional Options page, as follows:

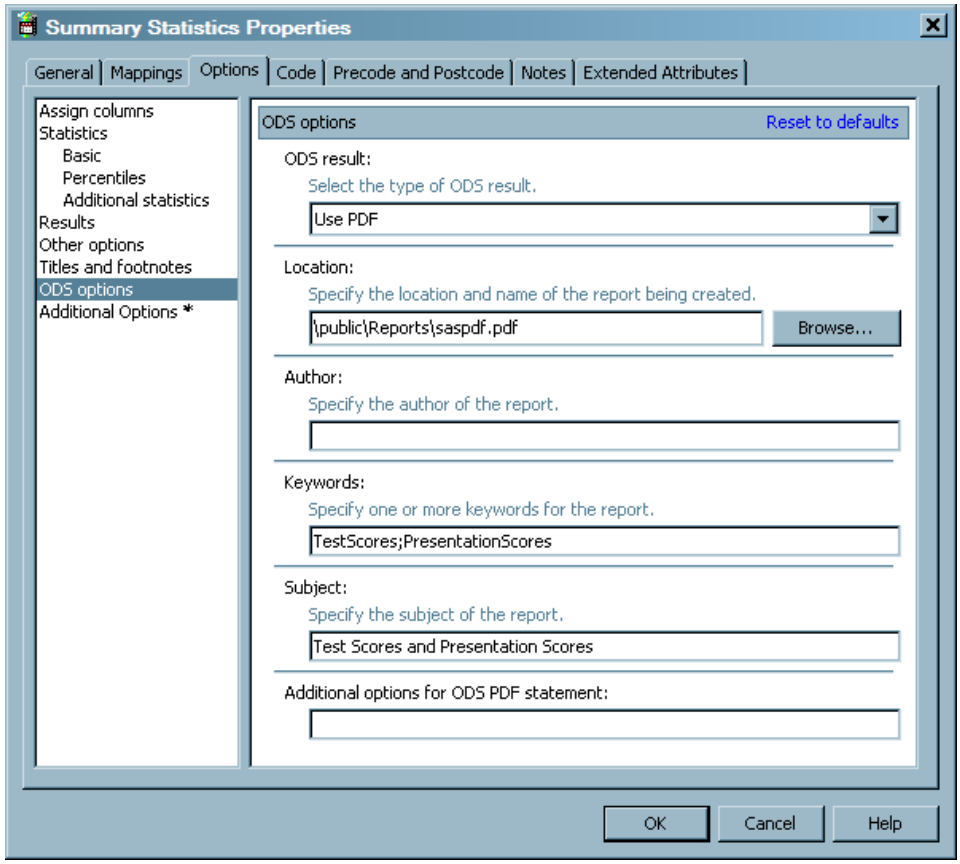
```
fw=8
```

Configure Reporting Options

Use the remaining option pages to create and save a report based on the analysis conducted in the job. Perform the following steps to set the reporting options:

1. Click **Title and footnotes** to access the Title and footnotes page and enter up to three headings and two footnotes.
2. Click **ODS options** to access the ODS options page. You can choose between HTML, RTF, and PDF output and enter appropriate settings for each. The sample job uses PDF output. Therefore, a location, a set of keywords, the subject of the report, and code to enable ODS graphics are added to the fields that are displayed when **Use PDF** is selected in the **ODS Result** field. (The path specified in the **Location** field is relative to the SAS Application Server that executes the job.)

Display 18.49 *Sample ODS Options*



Note: You can set additional reporting and formatting options in the **Specify other options for OPTIONS statement** field on the Other options page. For example, the following options are set for the sample job:

```
options nodate pageno=1 linesize=80 pagesize=60
```

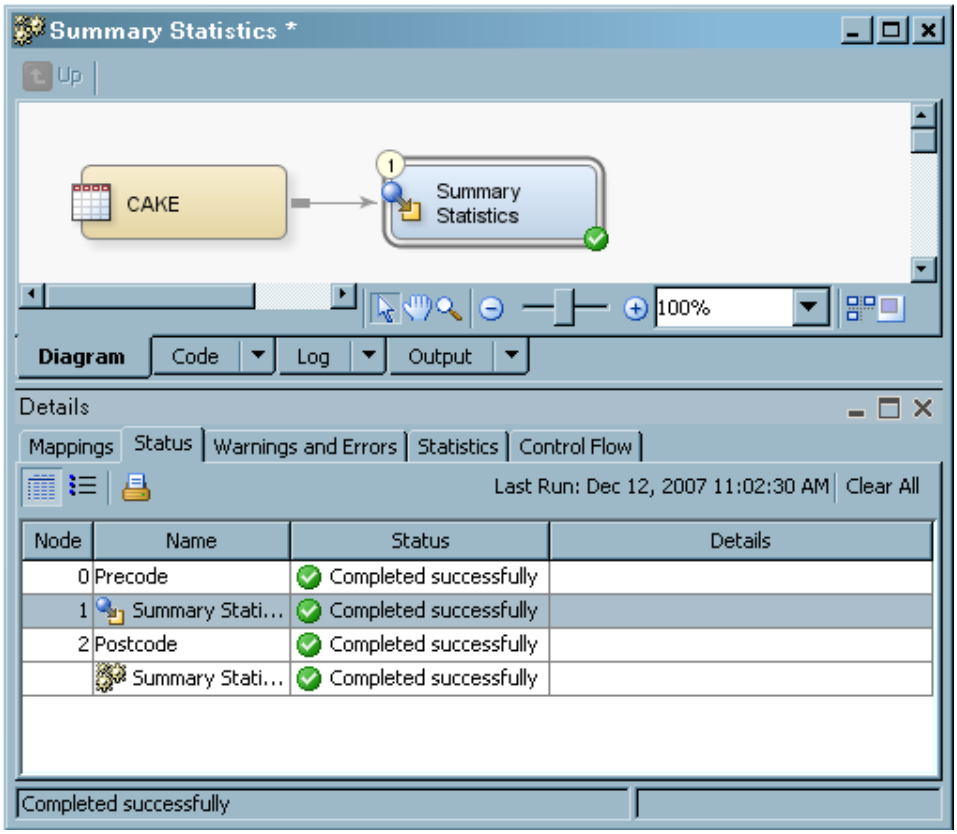
3. Click **OK** to save the settings for the **Options** tab.

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. The following display shows a successful run of a sample job.

Display 18.50 *Successfully Completed Sample Job*



2. If error messages are displayed on the **Status** tab, read and respond to the messages as needed.
3. To view the summary statistics, click the **Output** tab in the Job Editor window. The following display shows the analysis for the sample job.

Display 18.51 *Sample Output*

```
                          The MEANS Procedure

Variable        N            Mean         Maximum         Minimum         Std Dev
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
PresentScore    20      76.1500000      93.0000000      56.0000000       9.3767630
TasteScore      20      81.3500000      94.0000000      72.0000000       6.6115927
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

                          The SAS System                                1
```

4. Open the PDF document that you created and saved earlier. The following display illustrates a sample report based on the summary statistics generated by the sample job.

Display 18.52 *Sample PDF Output*

The SAS System

The MEANS Procedure

Variable	N	Mean	Maximum	Minimum	Std Dev
PresentScore	20	76.1500000	93.0000000	56.0000000	9.3767630
TasteScore	20	81.3500000	94.0000000	72.0000000	6.6115927

Creating a Summary Tables Report from Table Data

Overview

You can use a Summary Tables transformation as an interface to the TABULATE procedure. The TABULATE procedure displays descriptive statistics in tabular format, using some or all of the variables in a data set. You can create a variety of tables ranging from simple to highly customized. It computes many of the same statistics that are computed by other descriptive statistical procedures such as MEANS, FREQ, and REPORT.

The TABULATE procedure provides the following:

- simple but powerful methods to create tabular reports
- flexibility in classifying the values of variables and establishing hierarchical relationships between the variables
- mechanisms for labeling and formatting variables and procedure-generated statistics

It displays descriptive statistics in tabular format, using some or all of the variables in a data set. You can create a variety of tables ranging from simple to highly customized.

Problem

You want to print a tabular report of summary data from a data table.

Solution

You can use the Summary Tables transformation in a job that generates a tabulated data and creates an ODS document that contains the results. This transformation uses the TABULATE procedure to display descriptive statistics in tabular format, using some or all of the variables in a data set. For example, you can create a job similar to the sample job featured in this topic. This sample job creates a table that contains summary information about energy consumption. Note that the output for this job is sent to the

Output tab in the Job Editor window and an ODS document that is configured in the job. The sample job includes the following tasks:

- "Create and Populate the Job" on page 414
- "Configure Analytical Options" on page 414
- "Configure Reporting Options" on page 416
- "Run the Job and View the Output" on page 417

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. Select and drag a Summary Tables transformation from the Analysis folder in the Transformations tree. Then, drop it in the empty job on the **Diagram** tab in the Job Editor window.
3. Select and drag the source table out of the Inventory tree. Then, drop it before the Summary Tables transformation on the **Diagram** tab.
4. Drag the cursor from the source table to the input port of the Summary Tables transformation. This action connects the source to the transformation.
5. Right-click the Summary Tables transformation, and click **Add Output Port** from the **Ports** option in the drop-down menu. This step enables you to add an output port to the transformation.
6. Select and drag the source table from the Inventory tree. Then, drop it after the Summary Tables transformation on the **Diagram** tab.
7. Drag the cursor from the Summary Tables transformation output port to the target table. This action connects the target to the transformation.

Display 18.53 *Sample Process Flow*



Note that the source table for the sample job is named ENERGY.

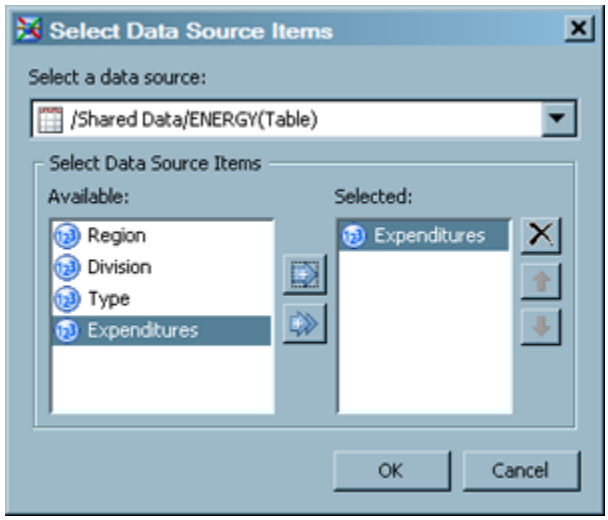
Configure Analytical Options

Use the **Options** tab in the properties window for the Summary Tables transformation to configure the SAS tables that are generated in the job and shape the output of your analysis. Note that the **Options** tab is divided into two parts, with a list of categories on the left-hand side and the options for the selected category on the right-hand side. Perform the following steps to set the options that you need for your job:

1. Open the properties window for the Summary Tables transformation in the **Diagram** tab in the Job Editor window. Then, click the **Options** tab.
2. Click **Assign columns** to access the Assign columns page. Use the column selection prompts to access the columns that you need in the SAS tables generated in your job.

For example, you can click ... for the **Select analysis columns (VAR statement)** to access the Select Data Source Items window, as shown in the following display.

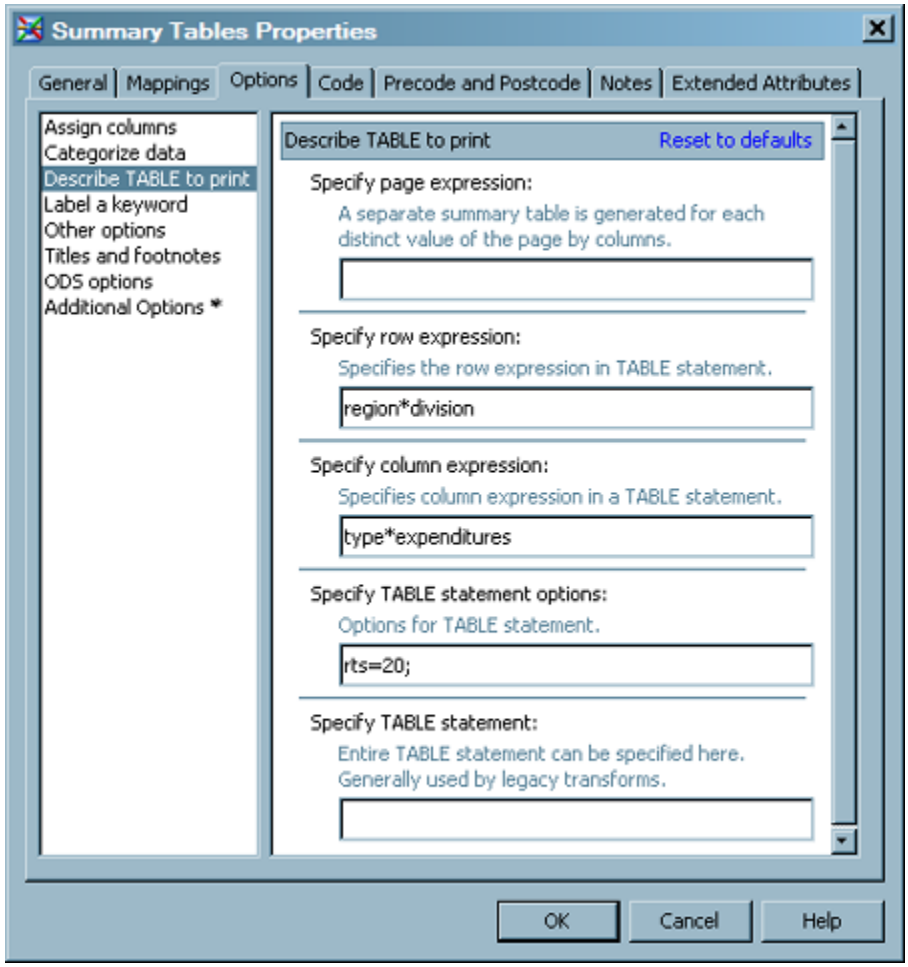
Display 18.54 *Sample Select Data Source Items Window*



In the sample job, the VAR statement column is Expenditures.

3. Set additional analytical options as needed. For example, the sample job has three CLASS statement columns, which are Region, Division, and Type. These columns are specified in the **Select columns to subgroup data (CLASS statement)** field on the Categorize data page. The TABLE statement options are set on the Describe TABLE to print page, as shown in the following display:

Display 18.55 *Sample TABLE Statement Options*



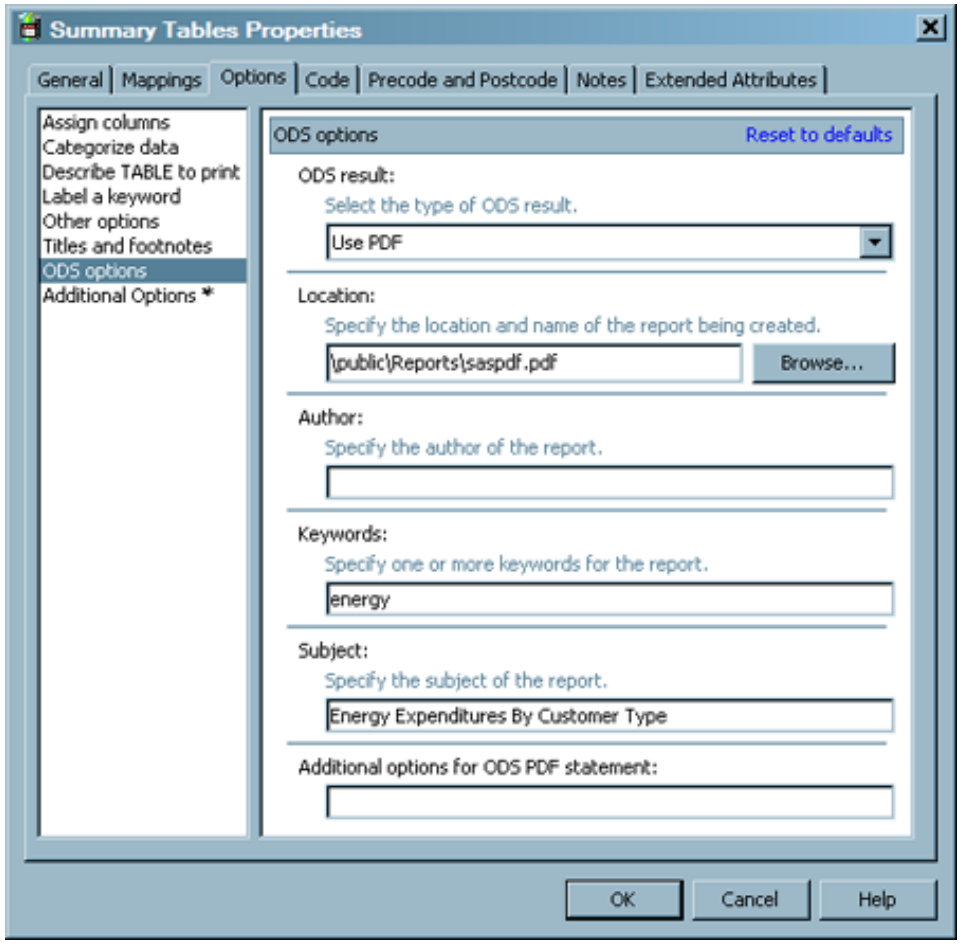
Note that separate options are set for the row expression, the column expression, and the TABLE statement as a whole. Taken together, these options define the table that is generated by the job and control how it is formatted.

Configure Reporting Options

Use the remaining option pages to create and save a report based on the analysis conducted in the job. Perform the following steps to set the reporting options:

1. Click **Title and footnotes** to access the Title and footnotes page and enter up to three headings and two footnotes.
2. Click **ODS options** to access the ODS options page. You can choose between HTML, RTF, and PDF output and enter appropriate settings for each. The sample job uses PDF output. Therefore, a location, a set of keywords, the subject of the report, and code to enable ODS graphics are added to the fields that are displayed when **Use PDF** is selected in the **ODS Result** field. (The path specified in the **Location** field is relative to the SAS Application Server that executes the job.)

Display 18.56 *Sample ODS Options*



Note: You can set additional reporting and formatting options in the **Specify other options for OPTIONS statement** field on the Other options page. For example, the following options are set for the sample job:

```
options nodate pageno=1 linesize=64 pagesize=40;
```

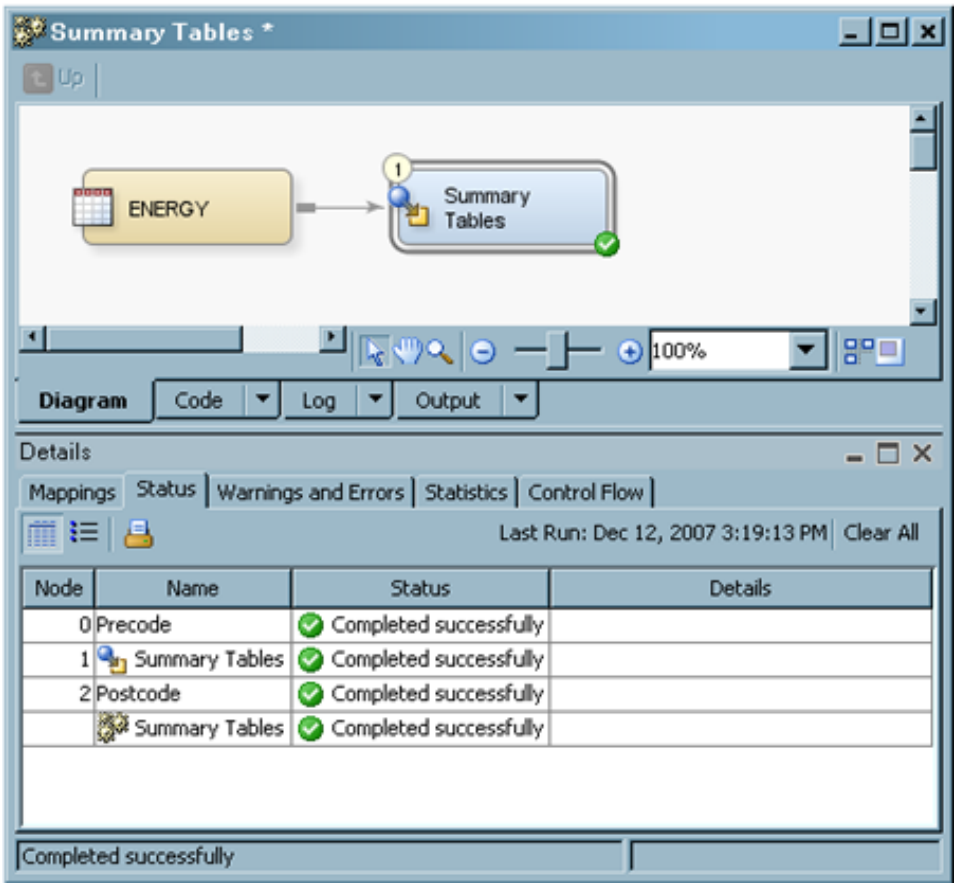
3. Click **OK** to save the settings for the **Options** tab.

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. The following display shows a successful run of a sample job.

Display 18.57 *Successfully Completed Sample Job*



2. If error messages are displayed on the **Status** tab, read and respond to the messages as needed.
3. To view the summary table created in the job, click the **Output** tab in the Job Editor window. The following display shows the analysis for the sample job.

Display 18.58 *Sample Output*

```
                    The SAS System

,□□□□□□□□□□□□□□□□□□...□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□†
,                    ,            Type            ,
,                    ‡□□□□□□□□□□□...□□□□□□□□□□□□□□□‰
,                    ,      1      ,      2       ,
,                    ‡□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□‰
,                    ,Expenditures,Expenditures,
,                    ‡□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□‰
,                    ,     Sum     ,     Sum      ,
‡□□□□□□□□...□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□‰
,Region  ,Division ,             ,             ,
‡□□□□□□□□□□□□□□□□□□‰             ,             ,
,1       ,1        ,     7477.00,      5129.00,
,        ‡□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□‰
,        ,2        ,    19379.00,     15078.00,
‡□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□‰
,4       ,3        ,     5476.00,      4729.00,
,        ‡□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□‰
,        ,4        ,    13959.00,     12619.00,
Š□□□□□□□□‹□□□□□□□□□‹□□□□□□□□□□□□‹□□□□□□□□□□□□□E
```

4. Open the PDF document that you created and saved earlier. The following display shows the summary table generated by the sample job.

Display 18.59 *Sample PDF Output*

The SAS System

		Type	
		1	2
		Expenditures	Expenditures
		Sum	Sum
Region	Division		
1	1	7477.00	5129.00
	2	19379.00	15078.00
4	3	5476.00	4729.00
	4	13959.00	12619.00

Chapter 19

Working with Loader Transformations

About Loader Transformations **421**

About the SPD Server Table Loader Transformation **422**

Teradata Table Loader Transformation **423**
Teradata Table Loader 423
Teradata Indexes 423
Teradata Custom Restart 423

About the Table Loader Transformation **424**

About the Oracle Bulk Table Loader Transformation **425**

About the DB2 Bulk Table Loader **426**

Setting Table Loader Transformation Options **427**
Problem 427
Solution 427
Tasks 427

Selecting a Load Technique in the Table Loader **429**
Problem 429
Solution 429
Tasks 429

Removing Non-Essential Indexes and Constraints during a Load **432**
Problem 432
Solution 432

Considering a Bulk Load **433**
Problem 433
Solution 433
Tasks 433

About Loader Transformations

SAS Data Integration Studio provides seven 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 and appends, and they can be used to maintain constraints.

SAS Data Integration Studio provides the following transformations for loading data into permanent output tables:

- The SCD Type 1 and Type 2 Loader transformations load source data into a dimension table, detect changes between source and target rows, update change tracking columns, and apply generated key values. These transformations implement slowly changing dimensions. For more information, see "Transformations That Support Slowly Changing Dimensions" on page 523.
- 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 422.
- The Teradata Table Loader transformation is specifically designed to optimally load Teradata tables. It provides different load options depending on whether the source table is in the same Teradata database as the target table. For more information, see "Teradata Table Loader Transformation" on page 423.
- 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 424.
- The Oracle Bulk Table Loader transformation can be used to bulk load SAS and most DBMS source tables to an Oracle target table. For more information, see "About the Oracle Bulk Table Loader Transformation" on page 425.
- The DB2 Bulk Table Loader transformation can be used to bulk load SAS and most DBMS source tables to a DB2 target table. For more information, see "About the DB2 Bulk Table Loader" on page 426.

All loader transformations write to a table that is registered to a library. For more information about registering tables and libraries, see the appropriate sections in the "Connecting to Common Data Sources" chapter of the *SAS Intelligence Platform: Data Administration Guide*.

For additional information, see "Usage Notes for Loaders" on page 669.

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 **Table Options** tab. Other loader options can be set in the **Options** tab. Additional table options not specified in these tabs can be set in the **Additional data table options** field located in the **Loader** window on the **Options** tab. 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.

All loader transformations write to a table that is registered to a library. For more information about registering SPD Server tables and libraries, see the "Establishing Connectivity to a Scalable Performance Data Server" section in the "Connecting to Common Data Sources" chapter of the *SAS Intelligence Platform: Data Administration Guide*.

Teradata Table Loader Transformation

Teradata Table Loader

The Teradata Table Loader transformation can be added to a process flow when a Teradata table is used as a target. The Teradata Table Loader also has a unique **Load Technique** tab that provides different load options depending on whether the source table is in the same Teradata database as the target table.

All loader transformations write to a table that is registered to a library. For more information about registering tables and libraries, see the "Overview of SAS/ACCESS Connections to RDBMS" section in the "Connecting to Common Data Sources" chapter of the *SAS Intelligence Platform: Data Administration Guide*.

You can specify a variety of table options unique to Teradata tables on the **Table Options** tab. Other loader options can be set on the **Options** tab.

The Teradata Table Loader transformation also supports the pushdown feature that enables you to process relational database tables directly on the appropriate relational database server. For more information, see "Pushing ELT Job Code Down to a Database" on page 198.

Teradata Indexes

Teradata indexes differ from other database indexes and require special handling. These differences apply to all uses of the Teradata tables, not just when using the Teradata Table Loader. Specifically, primary indexes cannot be dropped or removed for existing tables. They have to be created when the table is created. You can query for the Teradata Primary Index (PI) and give it a name if it does not have one. You can register this PI using the Register Tables function on the **File** menu. Once the PI is registered, go to the **Index** tab on the Teradata table's properties. A check box will show which index is the PI. All Teradata tables have a single primary index that cannot be changed once the table is registered unless it is dropped or recreated.

Teradata Custom Restart

Teradata custom restart allows a step to be restarted where the load stopped rather than being started from the beginning of the step. Teradata custom restart is available when loading from a SAS or other DBMS source that is not on the same server as the Teradata target. Custom restart is not available when Upsert is selected.

When custom restart is supported, the step determines the last good checkpoint, and the row number is saved as the restart number. After the error condition is fixed by an administrator (for example, the database size has been extended), the next run of that job will start loading the target table where it stopped.

The load styles that are available on the Teradata loader for SAS to Teradata loads are:

- **Append (Multiload)**

- **Determine load technique at runtime (Multiload/Fastload)**
- **Replace (Fastload)**
- **Replace (Multiload)**
- **Trickle Feed Append (TPUMP)**
- **Upsert (Multiload/Upsert)**

When restart is used, and when the **Determine load technique at runtime** option is selected, the same technique that was used in the first run is used during the restart.

Using the **Use TPT Utilities** option with the **Determine load technique at runtime** option provides a more seamless restart because Fastload without TPT does not support restart through the access engine, which calls the TPUMP functionality multistatement. Other TPT and load style combinations result in the following manner:

- When TPT is set, Multiload will generate CHECKPOINT=xxx.
- When TPT is not set, Multiload will generate ML_CHECKPOINT=xxx.
- Regardless of the TPT setting, CHECKPOINT=xxx will be generated when Fastload is used.

The **Use TPT Utilities** check box is located on the **Load Technique** tab of the Teradata Table Loader's properties window. This check box is available if the source table is not in the same Teradata database as the target table and if one of the load styles for the Teradata loader, except **Upsert (Multiload/Upsert)**, is selected. When this check box is selected, SAS Data Integration Studio uses the Teradata parallel transporter (TPT) API for loading data. You can select additional TPT options in the TPT window located on the **Teradata Options** tab on the **Table Options** tab in the Teradata Table Loader properties window.

For more information about restarting jobs, see "About Restarting Jobs" on page 201.

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.
- supporting the pushdown feature that enables you to process relational database tables directly on the appropriate relational database server. For more information, see "Pushing ELT Job Code Down to a Database" on page 198.

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=#.

About the Oracle Bulk Table Loader Transformation

The Oracle Bulk Table Loader transformation can be added to a process flow to take large amounts of data from a SAS or Oracle source file and bulk load it to an Oracle target.

The Oracle Bulk Table Loader contains several tabs to define the bulk loading method to use. The **Load Technique** tab and **Table Options** tab are specific to Oracle. Other loader options can be set on the **Options** tab.

The Oracle Bulk Table Loader functions like other loaders, but it also provides additional options available on the **Load Technique** tab. These options enable users to select the best method to load their data. The default bulk load method is Insert, and other options include Append, Replace, and Truncate. Additional options on the **Load Technique** tab allow users to drop and recreate indexes and constraints and to gather table statistics after the table has been bulk loaded.

In order for the Oracle Bulk Table Loader functionality to work properly, follow these data and configuration considerations:

- Oracle does not support table names with spaces in the name, so any table created in metadata with this name will not load properly.
- When an index is dropped or created, the index must be unique to the target table. The index cannot be used on any other table without causing a failure when trying to create the index because the index already exists in the database.
- The SQL loader must be installed as part of the Oracle client.

About the DB2 Bulk Table Loader

The DB2 Bulk Table Loader transformation can be added to a process flow to take large amounts of data from SAS and most DBMS source tables and bulk load it to a DB2 target. The DB2 Bulk Table Loader functions like other loaders. However, it loads only UDB (Linux, UNIX, and Windows), not z/OS. Note that it does not support ODBC to DB2 or OLE/DB to DB2.

The DB2 Bulk Table Loader contains several tabs to define the bulk loading method to use. These tabs include the **Load Technique** tab, the **Table Options** tab, and the Loader pane in the **Options** tab.

The options on the **Load Technique** tab enable users to gather table statistics after the table has been bulk loaded and to select the best method to load their data. The default bulk load method is CliLoad, and other options include Import, Load, and CliLoad with truncate.

The other bulk load methods require certain privileges. To use the Load or CliLoad method, a user must have system administrator authority, database administrator authority, or load authority on the database. The user must also have Insert privileges on the table being loaded. The Import method does not offer the same level of performance as the Load method. However, it is available to all users who have Insert privileges on the tables being loaded.

After the bulk load is processed, code is saved by the DB2 loader to retain statistics for quicker execution the next time the table is loaded. The user can set a value for the number of frequent values that are used in the generated code. This value is entered on the **Options** tab of the Properties window.

Note: If indexes or constraints exist in metadata for a table that does not already exist at load time, then indexes registered in metadata will be used at create time. This is the only time that the metadata is read when creating indexes for a table using the DB2 Bulk Table Loader.

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 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:

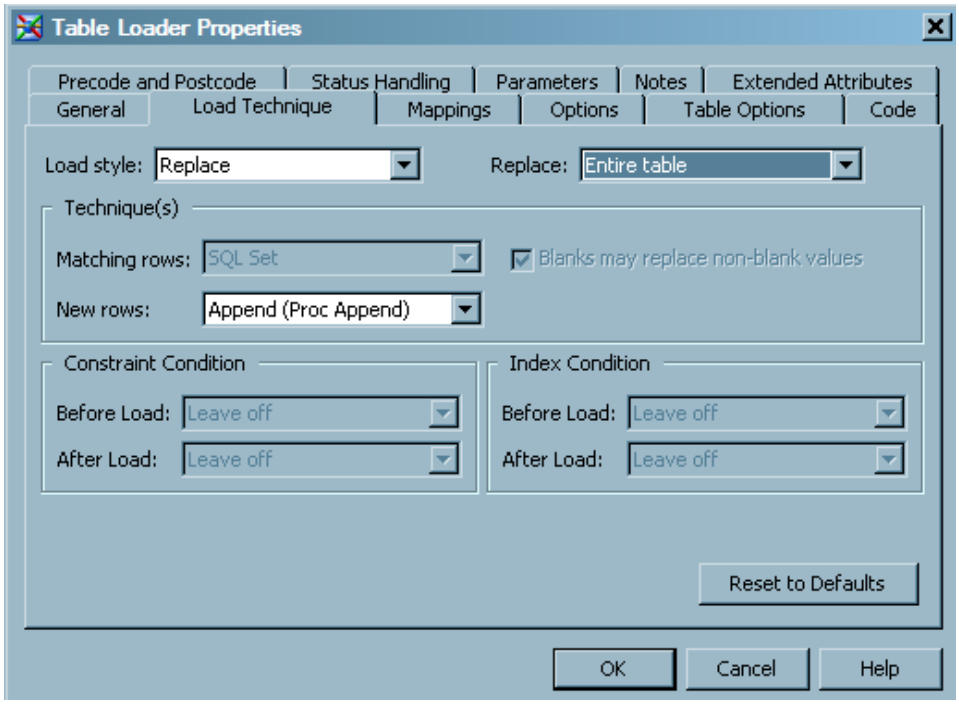
1. Create a job in SAS Data Integration Studio and give it an appropriate name.
2. Drop the Table Loader transformation from the **Process** tab onto the Job Editor window. Drag and drop a source table and a target table from the **Inventory** or **Folders** tab to the appropriate sides of the Table 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 19.1 *Sample of the Table Loader Flow*



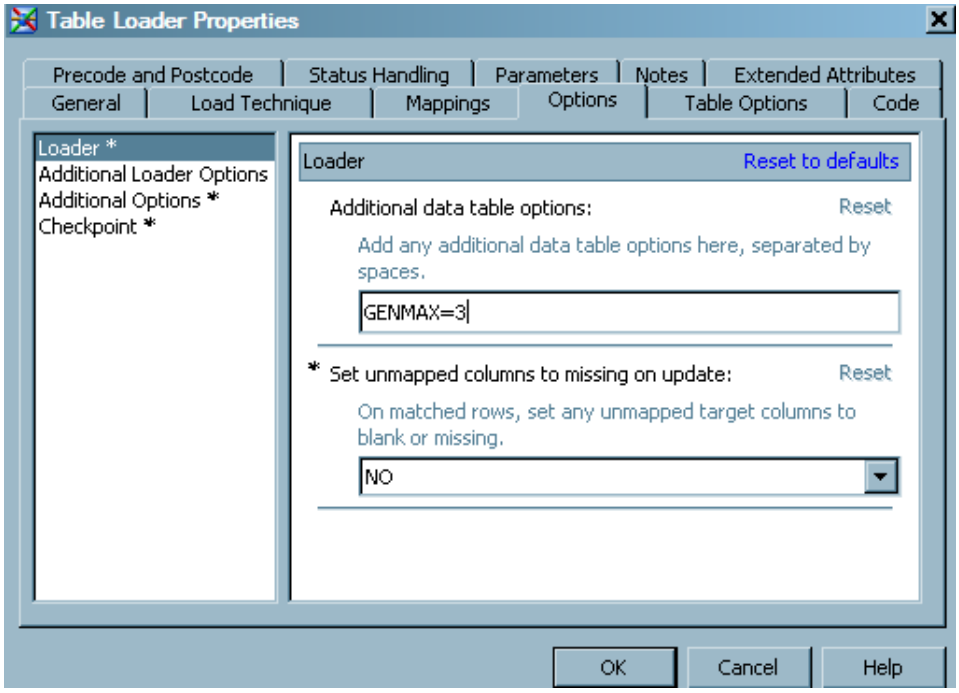
3. Set the Load Technique by right-clicking on the Table Loader transformation to open the **Properties** window. Select the **Load Technique** tab. Here you can set the load style, the technique to be used, and the constraints or indexes. For this example, which uses a SAS table, the selections are shown in the following display.

Display 19.2 *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 19.3 *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 in the Table Loader

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.

Note: All table loaders have similar **Load Technique** tabs, but this example is specific to the Table Loader Transformation. For specific instructions about other loaders, see the Help topics for the other loaders.

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 429
- "Add New Rows" on page 430
- "Match and Update Rows" on page 431

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 433
- "Set the BULKLOAD Option for a DBMS Table" on page 434

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 **Options** tab.
2. Click on the **Table Options** tab.
3. Enter **`BULKLOAD=YES`** in the field labeled **Additional Table options**.

Chapter 20

Working with SAS Sort Transformations

About Sort Transformations **435**

Optimizing Sort Performance **435**
Problem 435
Solution 436

Creating a Table That Contains the Sorted Contents of a Source **438**
Problem 438
Solution 438
Tasks 438

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 435. 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 438.

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 white paper that is available from **http://support.sas.com/resources/papers/tnote/tnote_performance.html**.

Table 20.1 *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: • Minimize row width. • Drop unnecessary columns. • Minimize pad bytes.
Direct sort utility files to fast storage devices	Use the WORK invocation option, the UTILLOC invocation option, or both options to direct SORT procedure utility files to fast, less-utilized storage devices. Some procedure utility files are accessed heavily, and separating them from other active files might improve performance.
Distribute sort utility files across multiple devices	Distribute SORT procedure utility files across multiple fast, less-utilized devices. Direct the SORT procedure utility file of each job to a different device. Use the WORK invocation option, the UTILLOC invocation option, or both options.
Pre-sort explicitly on the most common sort key	SAS Data Integration Studio might arrange a table in sort order, one or multiple times. For large tables in which sort order is required multiple times, look for a common sort order. Use the MSGLEVEL=I option to expose information that is in the SAS log to determine where sorts occur.
Change the default SORTSIZE value	For large tables, set SORTSIZE to 256 MB or 512 MB. For extremely large tables (a billion or more wide rows), set SORTSIZE to 1 GB or higher. Tune these recommended values further based on empirical testing or based on in-depth knowledge of your hardware and operating system.
Change the default MEMSIZE value	Set MEMSIZE at least 50% larger than SORTSIZE.
Set the NOSORTEQUALS system option	In an ETL process flow, maintaining relative row order is rarely a requirement. If maintaining the relative order of rows with identical key values is not important, set the system option NOSORTEQUALS to save resources.

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: • input data • SORT procedure utility file • output data
Size input data	Because sorting is so I/O intensive, it is important to start with only the rows and columns that are needed for the sort. The SORT procedure WORK files and the output file are dependent on the input file size.
Size SORT procedure utility files	Consider a number of factors to size the SORT procedure utility files: • sizing information of the input data • any pad bytes added to character columns • any pad bytes added to short numeric columns • pad bytes that align each row by 8 bytes (for SAS data sets) • 8 bytes per row overhead for EQUALS processing • per-page unused space in the SORT procedure utility files • multi-pass merge: doubling of SORT procedure utility files (or sort failure)
Size of output data	To size the output data, apply the sizing rules of the destination data store to the columns that are produced by the sort.

Creating a Table That 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 438
- "Specify How to Sort Information in the Target" on page 439
- "Run the Job and View the Output" on page 439

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 150.)

The following example shows the sample process flow. The source table is named ALL_EMP and the permanent target table is named EMPSORT.

Display 20.1 *Sample Sort Process Flow Diagram*



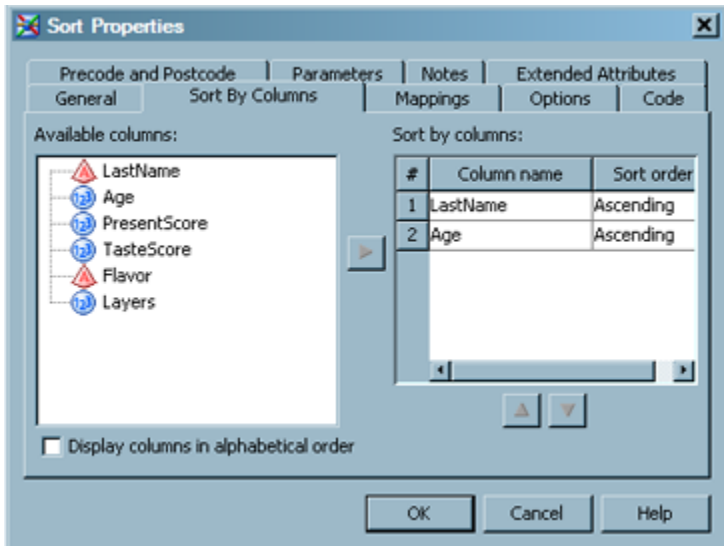
Specify How to Sort Information in the Target

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 20.2 *Completed Sort Tab for Sample Job*



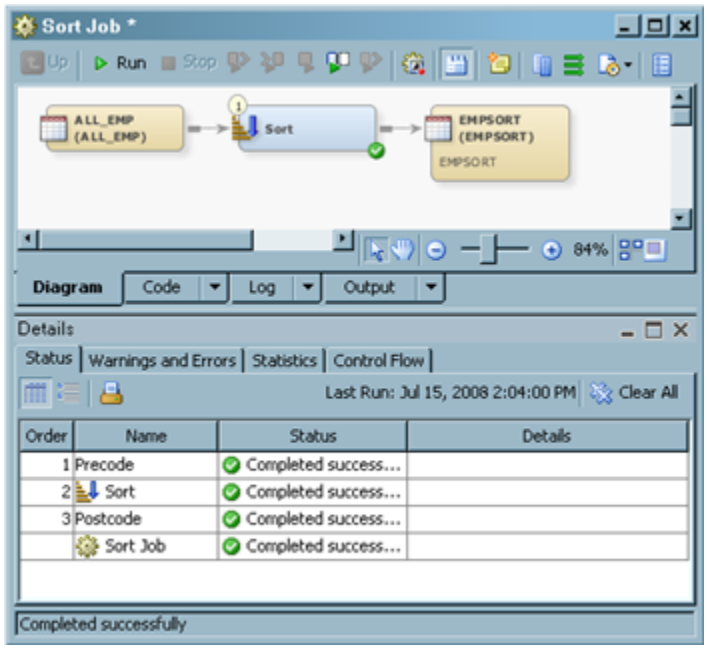
Note: Additional sorting options can be specified on the **Options** tab.

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.

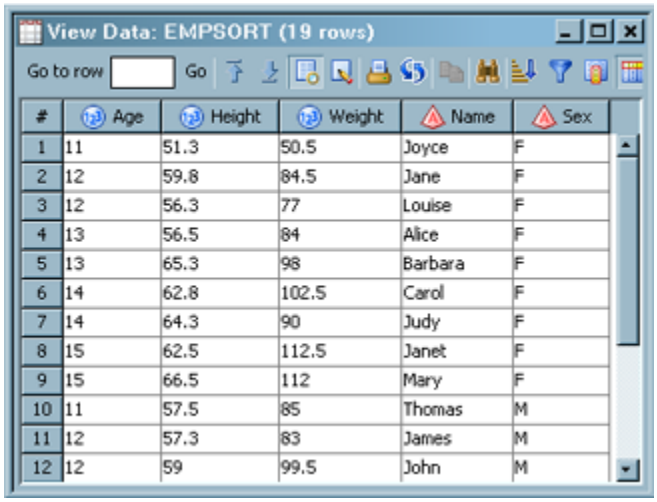
Display 20.3 *Successfully Completed Sample Job*



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 20.4 *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 21

Working with SQL Join Transformations

About Join Transformations **443**
Overview 443

Using the Designer Window **443**
Problem 443
Solution 444
Tasks 444

Reviewing and Modifying Clauses, Joins, and Tables in an SQL Query **445**
Problem 445
Solution 445
Tasks 445

Understanding Automatic Joins **447**
The Autojoin Process 447
A Sample Auto-Join Process 448

Selecting the Join Type **450**
Problem 450
Solution 450
Tasks 450

Adding User-Written SQL Code **452**
Problem 452
Solution 452
Additional Information 453

Debugging an SQL Query **453**
Problem 453
Solution 454
Tasks 454

Adding a Column to the Target Table **455**
Problem 455
Solution 455
Tasks 455

Adding a Join to an SQL Query on the Designer Tab **455**
Problem 455
Solution 456
Tasks 456

Creating a Simple SQL Query **457**
Problem 457
Solution 457
Tasks 457

Configuring a SELECT Clause **459**
Problem 459
Solution 459
Tasks 460

Adding a CASE Expression **461**
Problem 461
Solution 462
Tasks 462

Creating or Configuring a WHERE Clause **463**
Problem 463
Solution 463
Tasks 464

Adding a GROUP BY Clause and a HAVING Clause **465**
Problem 465
Solution 466
Tasks 466

Adding an ORDER BY Clause **468**
Problem 468
Solution 468
Tasks 469

Adding Subqueries **470**
Problem 470
Solution 470
Tasks 470

Validating or Submitting an SQL Query **475**
Problem 475
Solution 475
Tasks 475

Joining a Table to Itself **476**
Problem 476
Solution 476
Tasks 476

Using Parameters with an SQL Join **477**
Problem 477
Solution 477

Constructing a SAS Scalable Performance Data Server Star Join **478**
Problem 478
Solution 478
Tasks 478

Optimizing SQL Processing Performance **479**
Problem 479
Solution 479

Performing General Data Optimization **480**
Problem 480
Solution 480
Tasks 480

Influencing the Join Algorithm **481**
Problem 481
Solution 481
Tasks 481

Setting the Implicit Property for a Join **482**
Problem 482
Solution 482

Enabling Explicit Pass-Through Processing for SQL Join Transformations **484**
Problem 484
Solution 484
Tasks 484

Using Properties Window Options to Optimize SQL Processing Performance . . **486**
Problem 486
Solution 486
Tasks 486

About Join Transformations

Overview

The **SQL** folder in the Transformation tree contains a number of transformations that enable you to add SQL processing to jobs. This chapter is about the Join transformation.

The Join transformation provides an interface for building the statements and clauses that constitute queries. The transformation supports the SAS SQL procedure 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 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.

The Join transformation supports the pushdown feature that enables you to process relational database tables directly on the appropriate relational database server. For information about pushdown, see "Pushing ELT Job Code Down to a Database" on page 198.

See the SQL-related usage notes in "General Usage Notes" on page 645. For information about other SQL transformations, see "Working with Other SQL Transformations" on page 489.

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 21.1 Designer Tab Tasks

Task	Location	Action
Select and manipulate an object that displays in the **Diagram** tab.	Navigate pane	Click the object that you need to access.
Add SQL clauses to the flow shown on the **Diagram** tab.	SQL Clauses pane	Double-click the clause or drop it on the **Diagram** tab.
Review the list of columns in the source table and the target table. Note that you can specify alphabetic display of the columns by selecting **Display columns in alphabetical order**.	Tables pane	Click **Select**, **Where**, **Having**, **Group by**, or **Order by** in the SQL Clauses pane.
Display and update the main properties of an object that is selected on the **Diagram** tab. The title of this pane changes to match the object selected in the Navigate pane.	Properties pane	Click an object on the **Diagram** 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.	**Diagram** tab	Click **SQL Join**, **Create**, or **From** in the Navigate pane.
View the SAS code generated for the query.	**Code** tab	Click **Code** at the bottom of the **Diagram** tab.
View the log of a SAS program, such as the code that is executed or validated for the SQL query.	**Log** tab	Click **Log** at the bottom of the **Diagram** 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 445
- "Modify Properties of Clauses and Tables" on page 446

Tasks

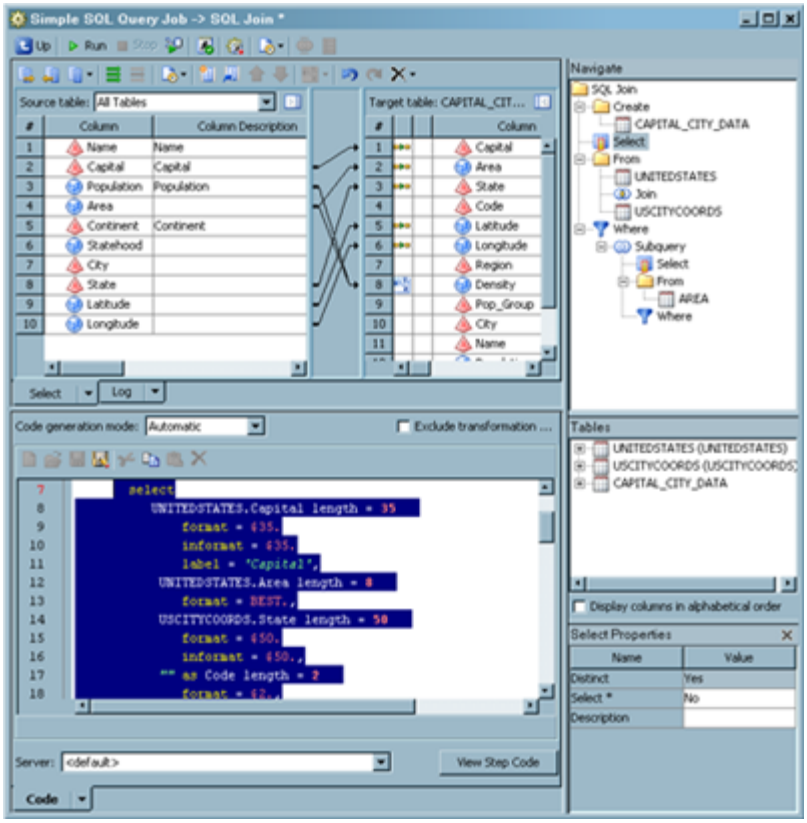
Review Clauses, Join, and Tables

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 21.1 *Information about a Select Clause on a Designer Tab*



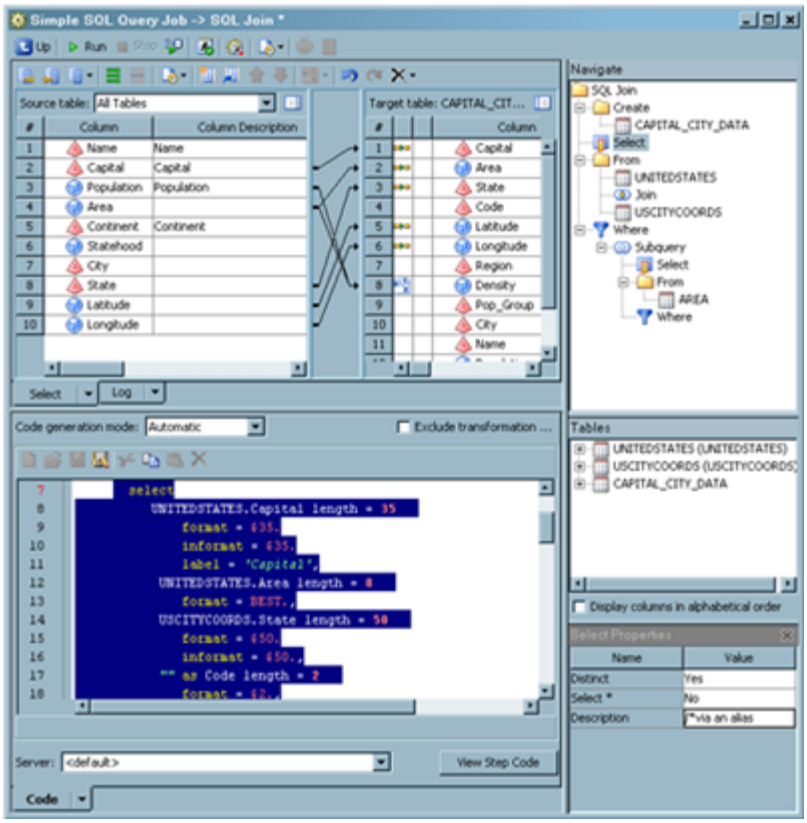
Note that **Select** is highlighted on the Navigate pane, and the SQL code for the SELECT clause is highlighted on the **Code** tab. To highlight the code for a query object, right-click the object in the Navigate pane and click **Find In**. Then, click **Code** in the submenu. Also note that the **Select** 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 21.2 *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.

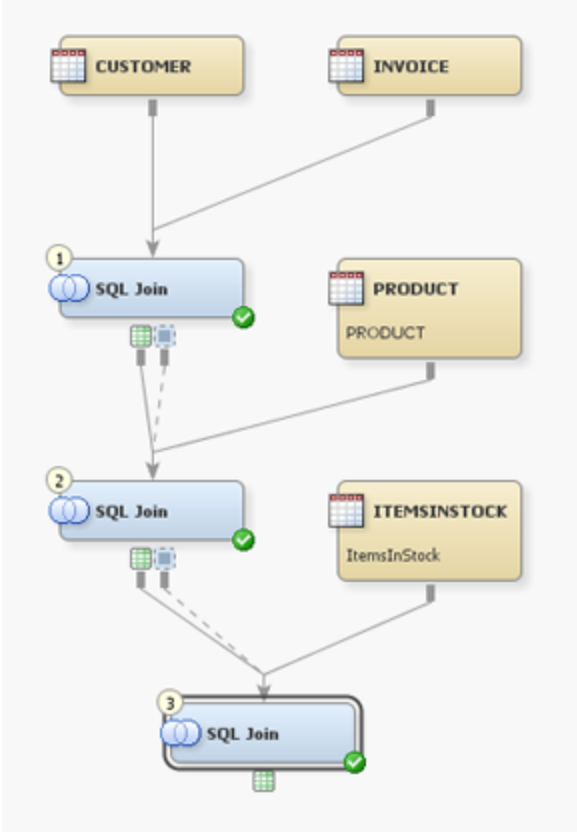
A Sample Auto-Join Process

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 21.3 *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

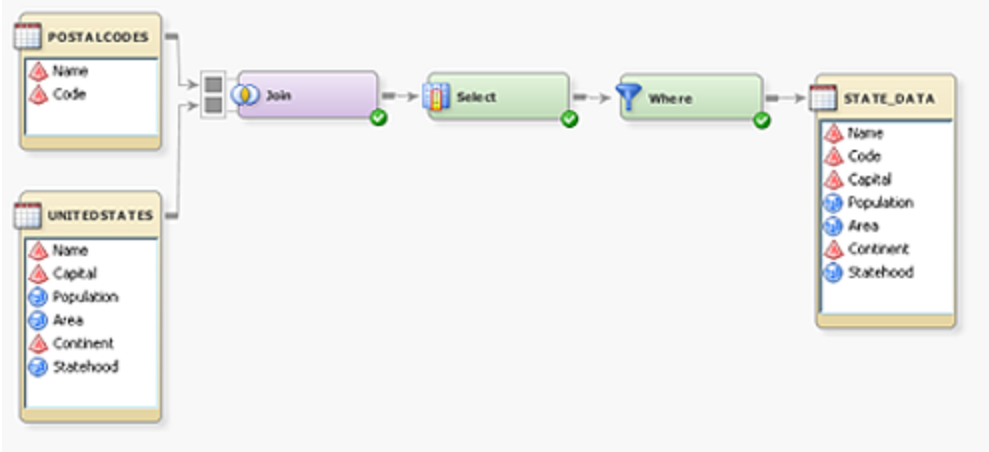
Examine a sample SQL query in a SAS Data Integration Studio job to see the effects of changing the join types that are used in the query. The sample query contains the tables and columns that are listed in the following table:

Table 21.2 *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 21.4 *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:

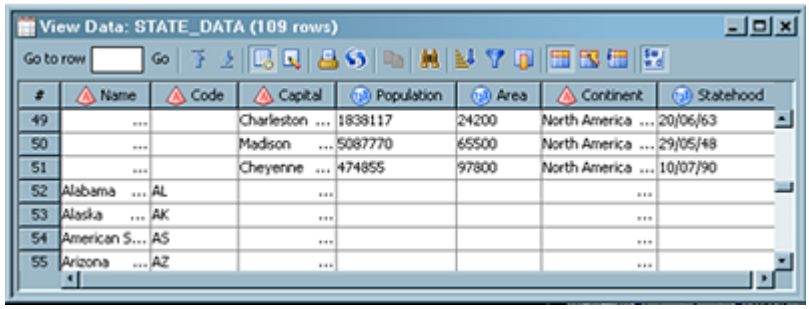
Table 21.3 *Results By Join Type*

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 21.5 *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

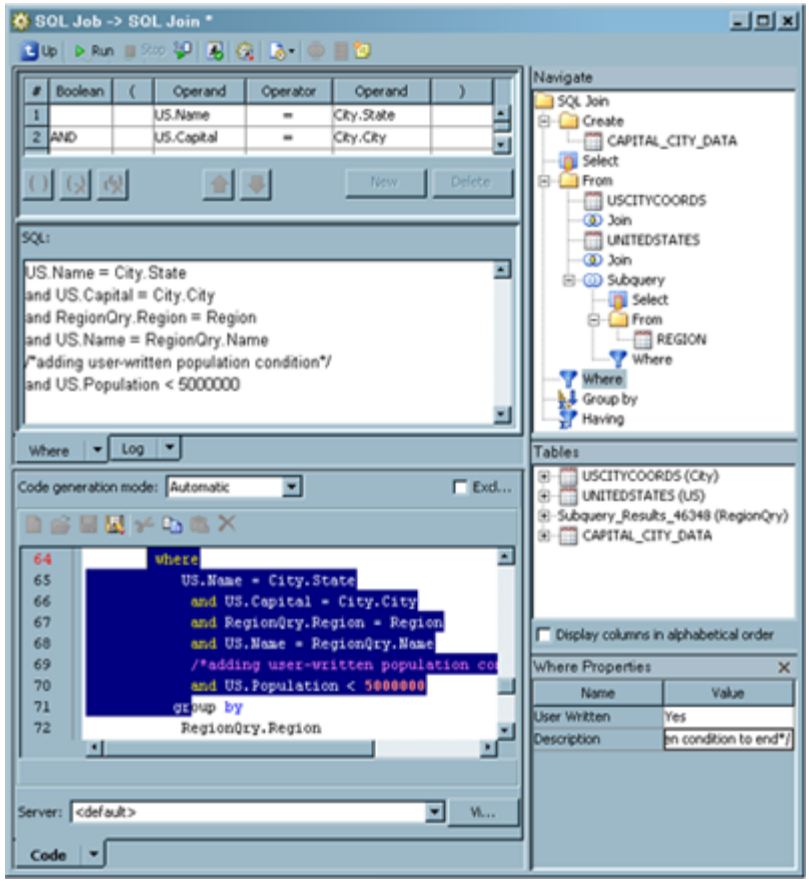
You want to add user-written code to an SQL query that is used in a SAS Data Integration Studio job. This user-written code can consist of SQL code that is added to a WHERE, HAVING, or JOIN clause. It can also overwrite the entire DATA step for the SQL Join transformation.

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 21.6 *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 271.

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 454
- "Examine Some Sample Method Traces" on page 454

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 21.4 *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 21.7 *Initial Job Flow*



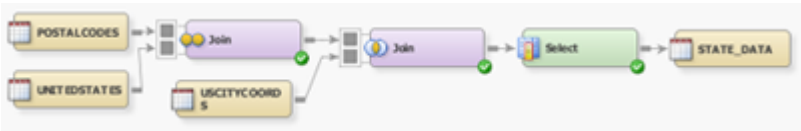
3. Move the new join to an appropriate location. Then, complete the following actions:
   - Connect the original join to one input port of the new join.

     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 21.8 *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 457
- "Create the SQL Query" on page 458

Tasks

Create and Populate the Job

Perform the following steps to create and populate the job:

1. Create an empty job.
2. Select and drag an SQL Join transformation from the **SQL** folder in the Transformations tree. Then, drop it in the empty job on the **Diagram** tab in the Job Editor window.
3. 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.
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 150.)

   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 21.9 *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

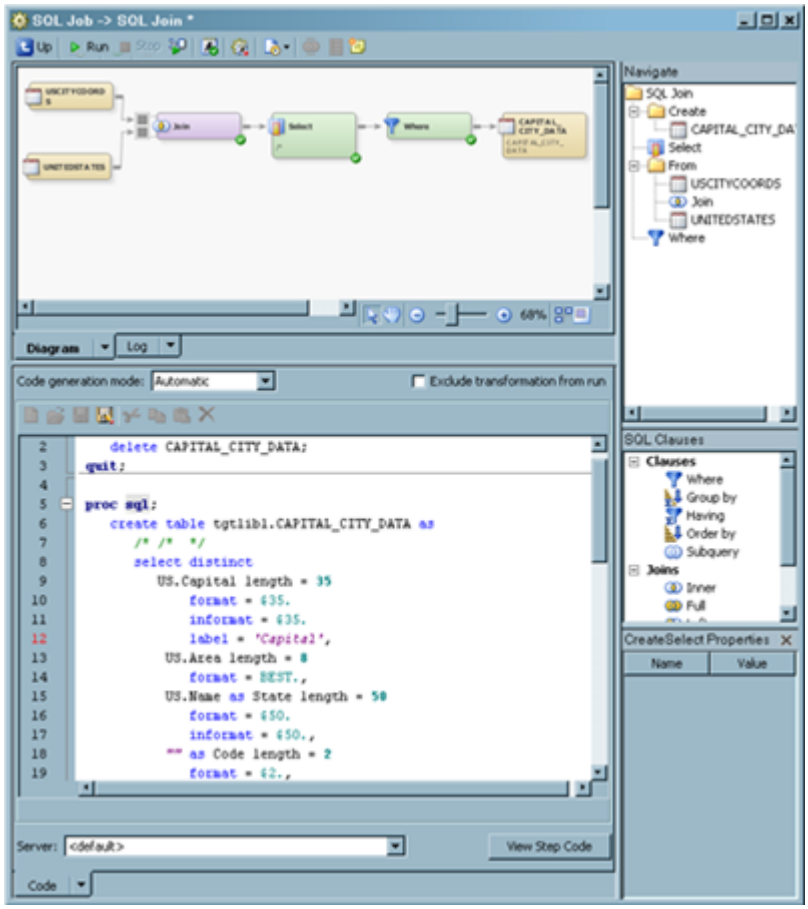
Perform the following steps to create the SQL query that populates the target table:

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 443.

   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 21.10 *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

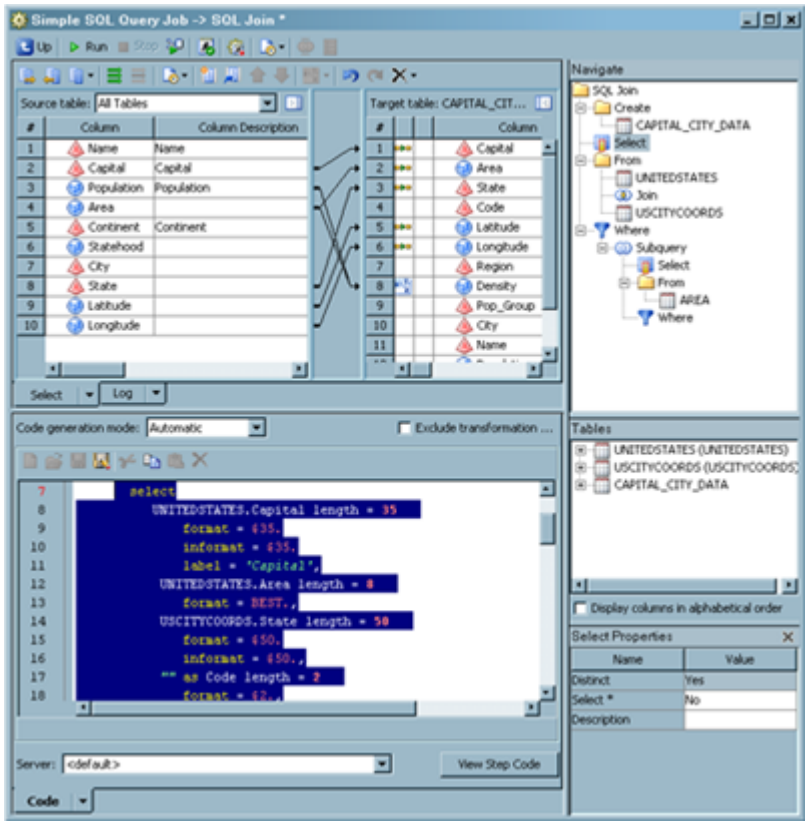
Configure the SELECT Clause with the Select Tab

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 620.
   - 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 21.11 *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.

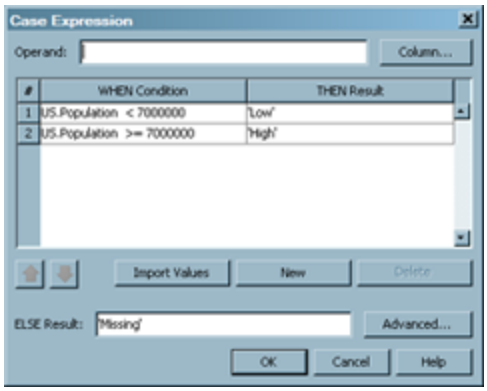
Tasks

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:

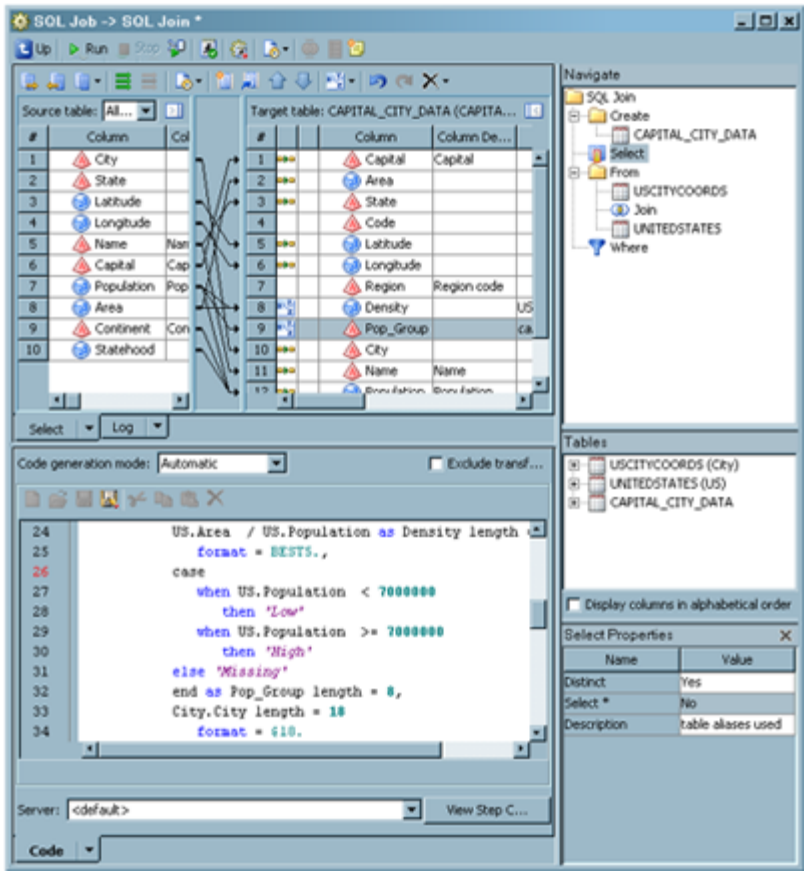
1. Access the CASE Expression window. To perform this task, click **CASE** in the drop-down menu for an Operand in a WHERE, HAVING, or JOIN condition. You can also access the **CASE** option in the Expression column for any column that is listed in the **Target table** field on the **Select** tab.
2. 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 21.12 *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 21.13 *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.

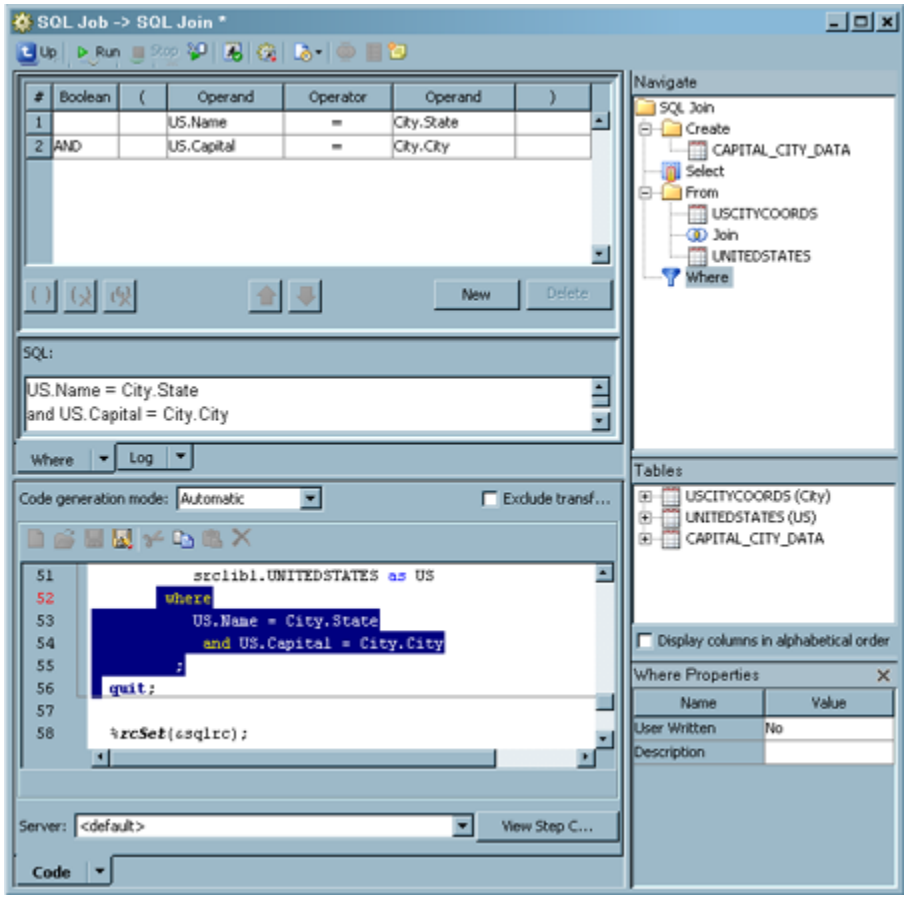
Tasks

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 21.14 *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.

Note that WHERE conditions are not optimized for these types of conditions:

- arithmetic operators
- variable-to-variable condition
- sounds-like operator
- any function other than SUBSTR and TRIM

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 466
- "Add a HAVING Clause to an SQL Query in the Diagram Tab" on page 467

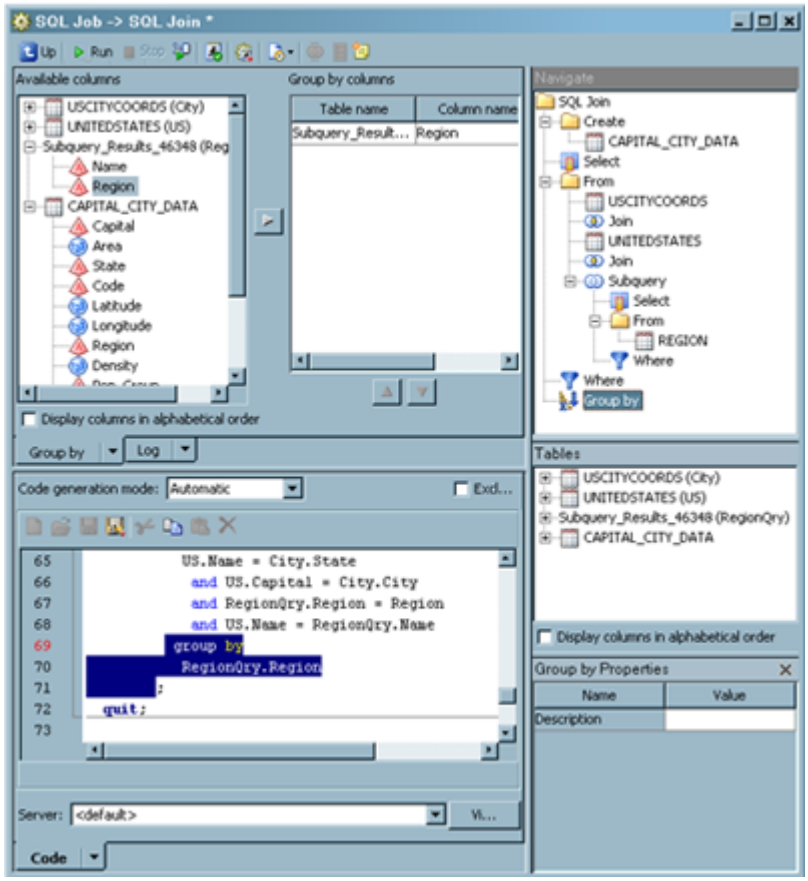
Tasks

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 21.15 *Sample SQL Query Grouped with a GROUP BY Clause*



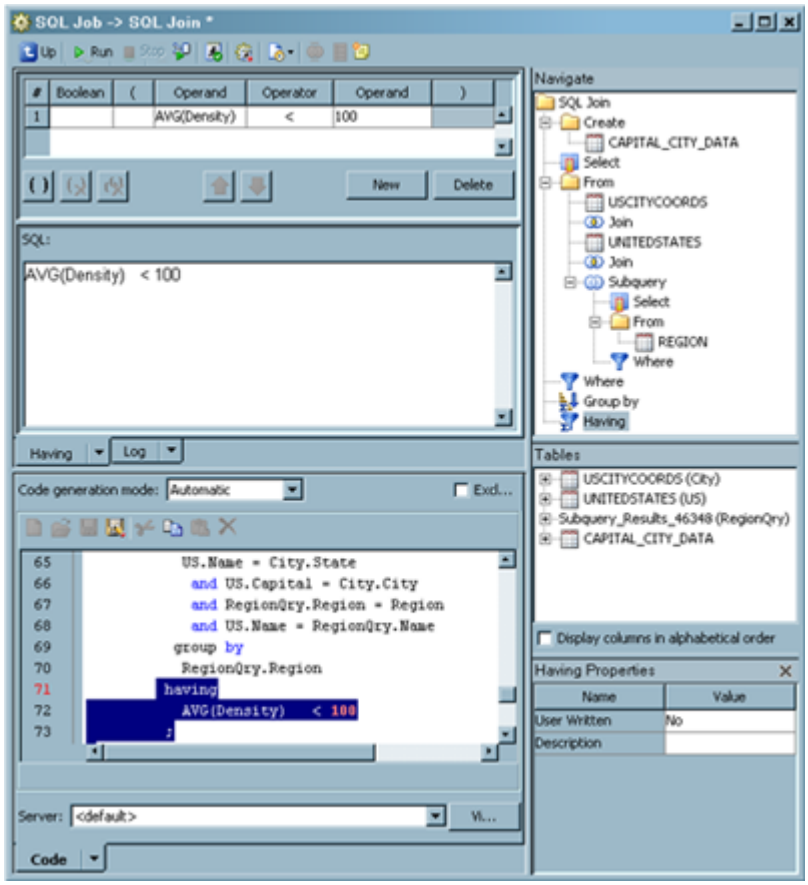
Note that the Group by column pane is set on the **Group by** tab, and the resulting SQL code is highlighted on the **Code** tab. The GROUP BY clause in the sample query groups the results of the query by the region of the United States.

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 21.16 *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.

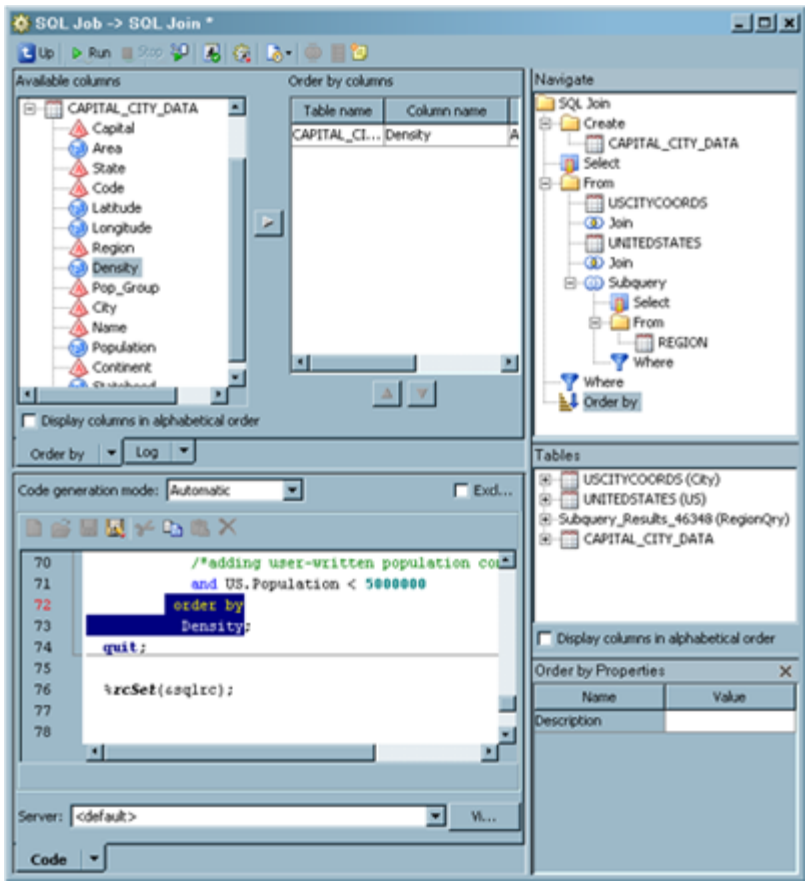
Tasks

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 21.17 *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 470 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 473 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 470
- "Add a Subquery to an SQL Clause" on page 473

Note: You can specify SQL subqueries in many different transformations in SAS Data Integration Studio.

For example, you could open the properties window for an SQL Merge transformation. Click the **Source** tab. Select **Subquery** in the **Source** control to display the Subquery Builder. Then you could click the **Filter and Sort** tab to specify a filter for the subquery. In general, the steps for creating SQL subqueries in SAS Data Integration Studio are similar to these steps that are described in this topic.

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 21.18 *Initial Subquery on Inner Join*

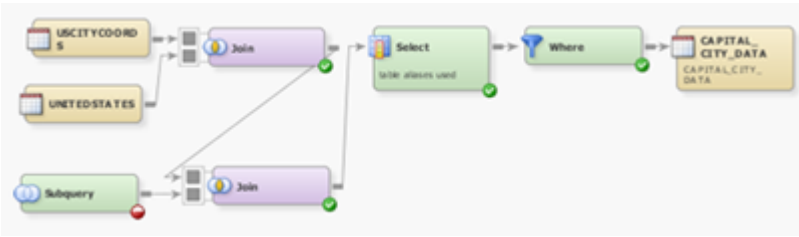


4. Move the subquery and the new join to appropriate locations. Then, complete the following actions:

   - 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.

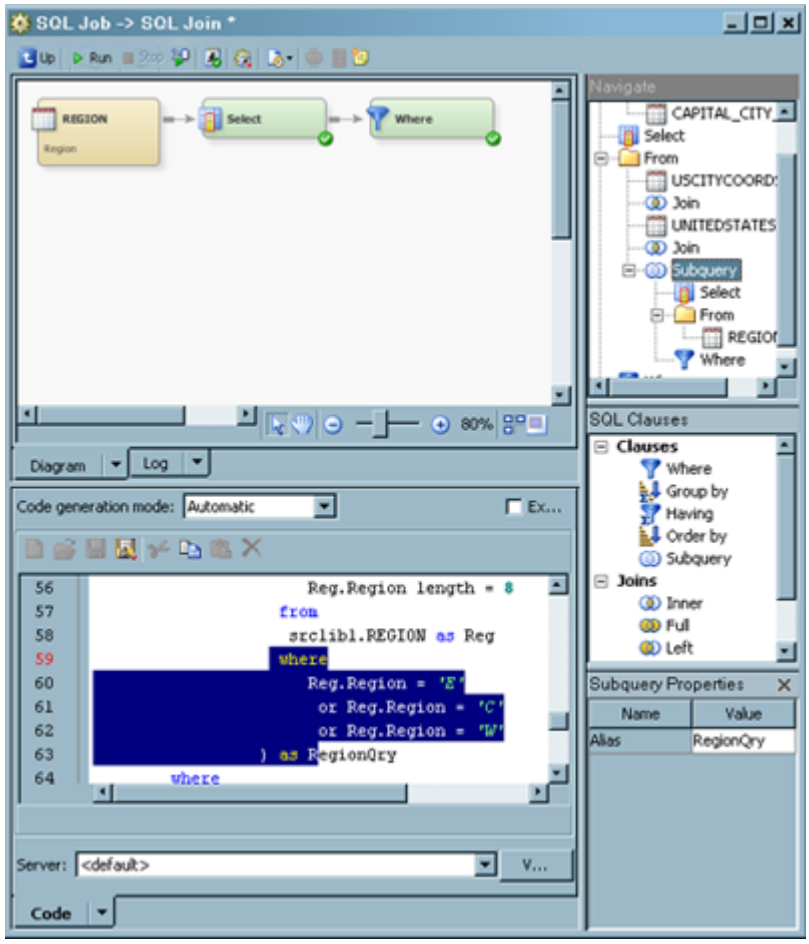
   A sample subquery on an inner join is shown in the following display.

Display 21.19 *Connected Subquery on Inner Join*



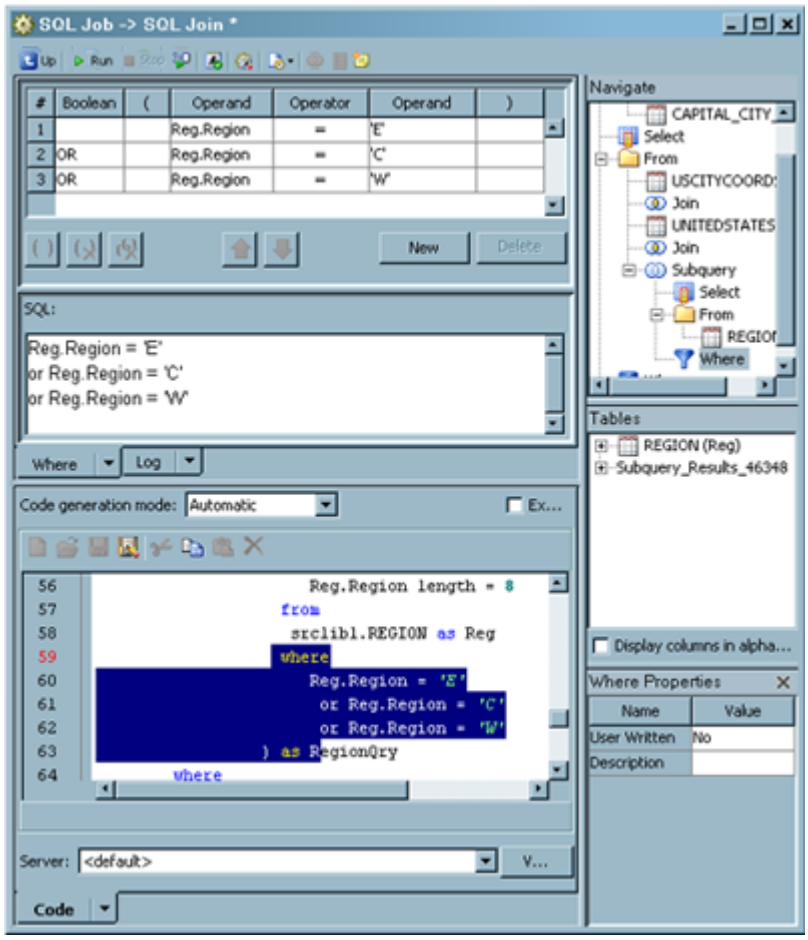
5. Click the **SubQuery** object. Note that the SubQuery Properties pane displays. Enter an appropriate value in the **Alias** field. (`RegionQry` was entered in the sample job.) If you do not enter an alias here, then the subquery fails. The system-generated name for the subquery results table is too 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 onto 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 21.20 *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 field displays the value **`r.Region`**.
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 21.21 *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
    ```

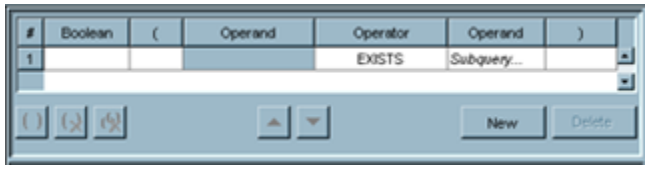
    This condition connects the inner join subquery to the main query.

Note: You can add a subquery to any place that you can add a table.

Add a Subquery to an SQL Clause

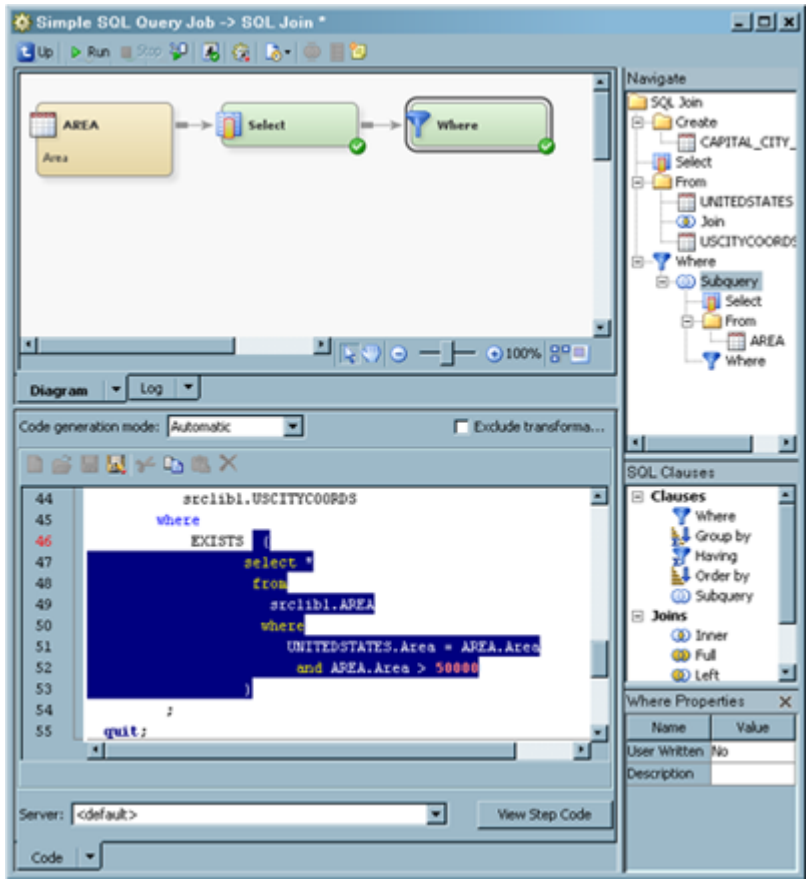
You can also add a subquery to SELECT, WHERE, HAVING clauses in SQL queries. The following display shows how a subquery can be added as a condition to a WHERE clause.

Display 21.22 *Add a Subquery to a WHERE Clause*



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 21.23 *Sample WHERE Clause Subquery*



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 are 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 475
- "Submit a Query As a Part of a SAS Data Integration Studio Job" on page 475

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.

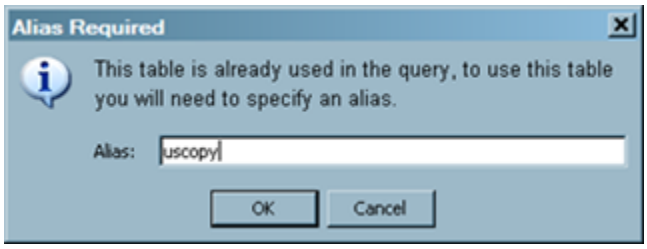
Tasks

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:

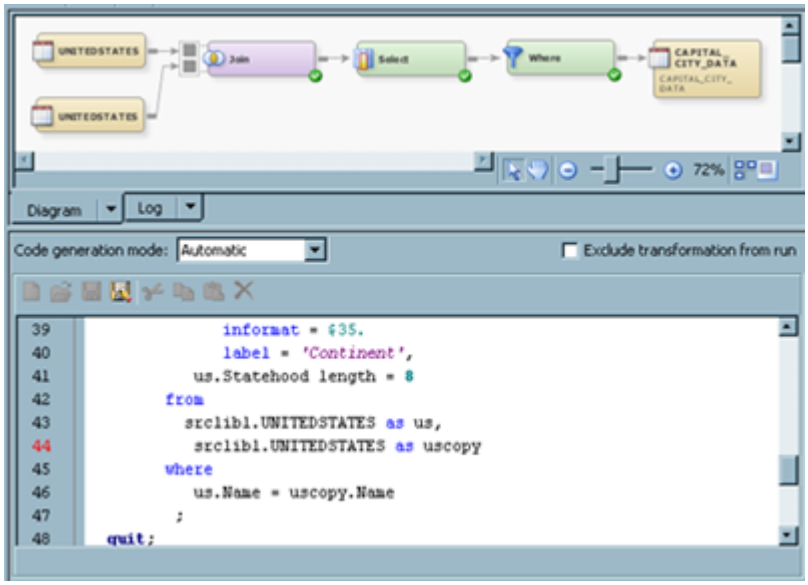
1. Create an SQL query in an empty job. The query should contain the SQL Join transformation, at least one source table, and a target table.
2. Open the **Designer** window for the SQL Join transformation. Click **Create** in the Navigate pane to access the **Diagram** tab and the SQL Clauses pane.
3. 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 21.24 *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 21.25 *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 509.
- 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 510.
- One or more parameters must be set for the parameterized job. See “Set Parameters for the Job” on page 511.

- The parameterized job must be embedded in an iterative job. See "About Iterative Jobs" on page 505.
- 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 512.

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 4.3 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 SAS SPD Server star join:

- 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: 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 453
- "Enabling Explicit Pass-Through Processing for SQL Join Transformations" on page 484

- "Influencing the Join Algorithm" on page 481
- "Performing General Data Optimization" on page 480
- "Understanding Automatic Joins" on page 447
- "Setting the Implicit Property for a Join" on page 482
- "Selecting the Join Type" on page 450
- "Using Properties Window Options to Optimize SQL Processing Performance" on page 486

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 480
- "Pre-Sort Data" on page 480

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 481
- "Index Joins" on page 481
- "Hash Joins" on page 482

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 buffer size 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 21.5 *Implicit and Explicit Properties for SQL Join Types*

Join Type	Join Condition Code

Inner	Can generate an implicit inner join condition in a WHERE statement near the end of the query:
	`where`
	`  POSTALCODES.Name = UNITEDSTATES.Name`
	You can use an implicit join only when the tables are joined with the equality operator. You can also generate an explicit inner join condition in a FROM clause in the SELECT statement:
	`from`
	`  srclib.POSTALCODES inner join`
	`  srclib.UNITEDSTATES`
	`  on`
	`  (`
	`  POSTALCODES.Name = UNITEDSTATES.Name`
	`  )`

Join Type	Join Condition Code

Full	Can generate an explicit join condition in a FROM clause in the SELECT statement:
	`from`
	`  srclib.POSTALCODES full join`
	`  srclib.UNITEDSTATES`
	`  on`
	`  (`
	`  POSTALCODES.Name = UNITEDSTATES.Name`
	`  )`

Left	Can generate an explicit join condition in a FROM clause in the SELECT statement:
	`from`
	`  srclib.POSTALCODES left join`
	`  srclib.UNITEDSTATES`
	`  on`
	`  (`
	`  POSTALCODES.Name = UNITEDSTATES.Name`
	`  )`

Right	Can generate an explicit join condition in a FROM clause in the SELECT statement:
	`from`
	`  srclib.POSTALCODES right join`
	`  srclib.UNITEDSTATES`
	`  on`
	`  (`
	`  POSTALCODES.Name = UNITEDSTATES.Name`
	`  )`

Cross	Can generate an explicit join condition in a FROM clause in the SELECT statement:
	`from`
	`  srclib.POSTALCODES cross join`
	`  srclib.UNITEDSTATES`

Union	Can generate an explicit join condition in a FROM clause in the SELECT statement:
	`from`
	`  srclib.POSTALCODES union join`
	`  srclib.UNITEDSTATES`

The **Implicit** property is disabled by default for all of the join types except the inner join.

Enabling Explicit Pass-Through Processing for SQL Join Transformations

Problem

You want to enable explicit pass-through processing for an SQL Join transformation.

Solution

Perform the following tasks:

- "Determine Whether Explicit Pass-Through Processing Is Possible" on page 484
- "Enable Explicit Pass-Through Processing" on page 485

Tasks

Determine Whether Explicit Pass-Through Processing Is Possible

Pass-through processing sends DBMS-specific statements to a database management system and retrieves the DBMS data directly. In some situations, explicit pass-through processing can improve the performance of SQL transformations in the context of a SAS Data Integration Studio job. However, 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:

- SAS formats
- SAS functions
- DATE or DATETIME literals or actual numeric values
- date arithmetic (usually does not work)
- INTO: macro variable
- data set options

Even if explicit pass-through is not enabled, the SAS SQL procedure 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. 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:

- 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.
- 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`**). For heterogeneous queries, try explicit pass-through. With the SQL Join transformation, 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 in the SQL Jon transformation can be used to create a homogeneous join, which can then enable an explicit pass-through operation. This property enables 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.

Enable Explicit Pass-Through Processing

To enable explicit pass-through processing by default for new instances of most SQL transformations, select **Tools** ⇨ **Options** ⇨ **Job Editor Tab**, and then select the pass-through check box in the **Automatic Settings** area. This setting affects SQL Join transformations and also any SQL transformation whose properties window includes a **Database pass-through** option on its **Options** tab. This includes SQL transformations such as Create Table, Insert Rows, Set Operators, Delete, and Update.

To enable pass-through processing for an SQL Join transformation, open the properties window for the transformation and specify **Yes** in the **Pass Through** property. 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 does not exist, it is created.

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 486
- "Optimize the SELECT Statement" on page 487
- "Set Buffering Options" on page 487
- "Use Threaded Reads" on page 487
- "Write User-Written Code" on page 488

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 cannot 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, know 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 cannot 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 22

Working with Other SQL Transformations

About Other SQL Transformations **489**
Overview 489
Query Builder Window 490

Inserting Rows into a Target Table **491**
Problem 491
Solution 492
Tasks 492

Using the SQL Set Operators Transformation **495**
Problem 495
Solution 495
Tasks 496

Enabling Explicit Pass-Through Processing for Other SQL Transformations . . **502**
Problem 502
Solution 502
Tasks 502

About Other SQL Transformations

Overview

The **SQL** folder in the Transformation tree contains a number of transformations that enable you to add SQL processing to jobs. This chapter is about the SQL transformations other than the Join transformation.

In addition to the Join transformation, the **SQL** folder contains the following transformations:

Table 22.1 *Other SQL Transformations*

Name	Description
Create Table	Provides a simple SQL interface for creating tables.

Name	Description
Delete	Generates a PROC SQL statement that deletes user-selected rows in a single target table. Supports delete, truncate, or delete with a WHERE clause. Also supports implicit and explicit pass-through.
Execute	Enables you to specify custom SQL code to be executed and provides SQL templates for supported databases.
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. For more information, see "Extracting Data from a Source Table" on page 712.
Insert Rows	Provides a simple SQL interface for inserting rows into a target table. For more information, see "Inserting Rows into a Target Table" on page 491.
Merge	Inserts new rows and updates existing rows using the SQL Merge DML command. The command was officially introduced in the SQL:2008 standard.
Set Operators	Enables you to use set operators to combine the results of table-based queries. For more information, see "Using the SQL Set Operators Transformation" on page 495.
Update	Updates user-selected columns in a single target table. The target columns can be updated by case, constant, expression, or subquery. Handles correlated subqueries.

Some functions in the Delete, Execute, Insert Rows, Merge, and Update transformations might work only when the table comes from a database management system that provides an implementation of an SQL command for which a SAS/ACCESS interface is available. One example is sort. You can use SAS tables and tables from database management systems that do not implement the SQL command, but these command-specific functions might not work.

You should enable explicit pass-through processing when you connect a database management system table to the Create Table transformation, Delete transformation, Insert Rows transformation, and Update transformation. For more information, see "Enabling Explicit Pass-Through Processing for Other SQL Transformations" on page 502.

See also the SQL-related usage notes in "General Usage Notes" on page 645. For information about the Join transformation, see "Working with SQL Join Transformations" on page 443.

Query Builder Window

The Query Builder window provides a convenient interface for creating SQL queries within transformations in SAS Data Integration Studio jobs. You can access the Query Builder or its components in the following transformations:

- Insert Rows — displays the Query Builder window when you click **Edit Query** on the **Insert** tab.
- Create Table — incorporates the tabs from the Query Builder window into its properties window.

- Additional SQL transformations that support subqueries include Delete, Update, and Merge. Note that the subquery version of the Query Builder window includes a **Name (ALIAS)** field. This field enables you to specify an alias for the subquery when it is used in a query.

The Query Builder window contains the following tabs:

Tabs	Description
Source	Identifies the tables used in a query. When multiple tables are selected, you can specify the join type and any applicable join conditions. Finally, you can create a subquery that you can use as the source of a query. *Note:* In the Subquery window, tables from all database management systems are handled in the same way. The interface in the window does not change to reflect the differences in how the various database management systems implement the SQL MERGE command. Therefore, it is possible to generate invalid SQL Merge code by using features that are not supported by a specific database management system. When you encounter SQL Merge errors, review the log for the SAS Data Integration Studio job. Also, see the documentation for the database management system for information about its implementation of the SQL MERGE command.
Result	Maps source tables to a target table. The tab uses the standard SAS Data Integration Studio mapping component.
Filter and Sort	Filters and sorts query results.
Group	Groups query results. You can also use the tab to filter the groups with a HAVING clause.
Code	Manages the code that is generated.

Inserting Rows into a Target Table

Problem

You want to insert rows into a target table that is included in a SAS Data Integration Studio job.

Solution

You can use the Insert Rows transformation to create an SQL query that will insert the rows into the target table.

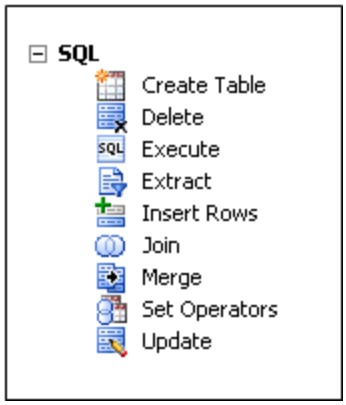
Perform the following tasks to insert the rows:

- "Create and Populate the Job" on page 492
- "Filter and Sort the Data" on page 493
- "Run the Job and Review the Results" on page 494

Insert Rows is one of the specialized transformations that are provided in the SQL folder in the SAS Data Integration Studio transformation tree.

The SQL folder is shown the following display:

Display 22.1 *SQL Folder*



These specialized transformations enable you to perform basic SQL tasks in SAS Data Integration Studio jobs. You can use the transformations to create tables, insert, merge, and delete rows, update columns, and execute custom SQL code. You can use the transformations in jobs in the same way that Insert Rows is used in the job described in this topic.

Tasks

Create and Populate the Job

Perform the following steps to create and populate a job that includes the Insert Rows transformation:

1. Create an empty job.
2. Select and drag an Insert Rows transformation from the **SQL** folder in the Transformations tree. Then, drop it in the empty job on the **Diagram** tab in the Job Editor window.
3. Select and drag the source table out of the Inventory tree. Then, drop it before the Insert Rows transformation on the **Diagram** tab. For example, you could add the flightdelays table, which contains data about delayed airlines flights, as the source table. The flightdelays table is a SQL Server table.

Note: You can also select the table by clicking the **Select a table** button next to the **Table** field on the **Source** tab.

4. Drag the cursor from the source table to the input port of the Insert Rows transformation. This action connects the sources to the transformation.
5. 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 Insert Rows transformation and click **Replace** in the pop-up menu. Then, use the Table Selector window to select the target table for the job. In this case, you want to insert selected rows into the SQL Server table flightdelays, so select it as the target table.

   The completed flow is shown in the following display:

Display 22.2 *Insert Rows Job Flow*



Note: If you need to use explicit pass-through, the source table and the target table must come from the same database management system. When you use implicit pass-through, the source table and the target table can come from different databases. You must use explicit pass-through if you need to sort the table on the **Filter and Sort** tab.

6. Open the properties window for the Insert Rows transformation.
7. Click **Options** and select **Database pass-through**.
8. Set the **SQL procedure pass-through** option to **Yes**. This setting enables the pass-through processing supported by the database management system for the source and target tables.

Filter and Sort the Data

Perform the following steps to filter and sort the rows that you want to insert:

1. Click **Insert**. Then, click **Edit Query** to access the Query Builder window.
2. Click **Filter and Sort**.
3. Click **New row** above the Filter (WHERE) table to add a row to the table. Then, enter your filter conditions.

   The filter conditions are shown in the following display:

Display 22.3 *WHERE Filter Conditions*

Filter (WHERE):

New row...

#	Boolean	(	Operand	Operator	Operand	)
1			flightdelays.Desti...	=	'LAX'	
2	AND		flightdelays.Delay	>	5	

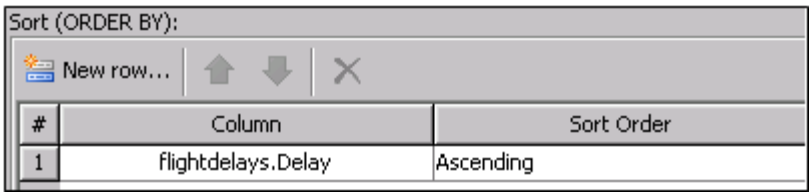
This target table will contain only those rows that have a destination of LAX and a delay of more than five minutes. Note that the operand *'LAX'* is enclosed in single quotation marks. SAS Data Integration Studio cannot successfully generate code for a job that includes a database management system table in which the double

quotation mark is used in the table name or the column names. The table that serves as the source and target for this job is a SQL Server table.

4. Click **New row** above the Sort (ORDER BY) table to add a row to the table. Then, enter your sort conditions.

   The sort conditions are shown in the following display:

   Display 22.4 *Sort Conditions*

   This setting creates an ascending sort based that is on the Delay column.

   Note: The sort function is supported only when explicit pass-through processing is enabled and the source and target tables come from Oracle, DB2, and SQL Server database management systems.

5. Click **OK** to save the query and return to the **Insert** tab. You can review the settings and mappings in the query on the tab.

   The following display shows the portion of the SQL query that contains the filter and sort conditions:

   Display 22.5 *SQL Filter and Sort Code*

   ```
   where
      flightdelays.Destination = 'LAX'
      and flightdelays.Delay > 5
   order by
      flightdelays.Delay
   ```

6. Click **OK** to save the settings in the properties window and return to the job flow.

Run the Job and Review the Results

Perform the following steps to run the job and review the results.

1. Run the job.
2. If the job completes without error, right-click the target table icon and click **Open**.

The View Data window appears, as shown in the following display:

Display 22.6 *Insert Rows Results*

LAX	1-10 Minutes	Domestic	4	6
LAX	1-10 Minutes	Domestic	2	6
LAX	1-10 Minutes	Domestic	2	6
LAX	1-10 Minutes	Domestic	1	6
LAX	1-10 Minutes	Domestic	1	6
LAX	1-10 Minutes	Domestic	4	6
LAX	1-10 Minutes	Domestic	5	6
LAX	1-10 Minutes	Domestic	6	6
LAX	1-10 Minutes	Domestic	7	6
LAX	1-10 Minutes	Domestic	5	7
LAX	1-10 Minutes	Domestic	6	7
LAX	1-10 Minutes	Domestic	1	7
LAX	1-10 Minutes	Domestic	6	7
LAX	1-10 Minutes	Domestic	6	7
LAX	1-10 Minutes	Domestic	5	7
LAX	1-10 Minutes	Domestic	5	7
LAX	1-10 Minutes	Domestic	7	7

Using the SQL Set Operators Transformation

Problem

You want to combine the results of table-based queries.

Solution

You can use the SQL Set Operators transformation in a SAS Data Integration Studio job. This transformation generates a PROC SQL statement that combines the results of two or more queries in various ways by using the following set operators:

- UNION: Produces all unique rows from both queries
- EXCEPT: Produces rows that are part of the first query only
- INTERSECT: Produces rows that are common to both query results
- OUTER UNION: Concatenates the query results

The operator is used between the two queries, as shown in the following example:

```
select columns from table
set-operator
select columns from table;
```

The semicolon is placed after the last SELECT statement only. Set operators combine columns from two queries based on their position in the reference tables without regard to the individual column names. Columns in the same relative position in the two queries must have the same data types. The column names in the first query become the column names of the output table. Therefore, only its columns are propagated to the output table.

Perform the following tasks:

- "Create and Populate the Job" on page 496
- "Configure the Queries" on page 497
- "Run the Job and Review the Results" on page 501

Tasks

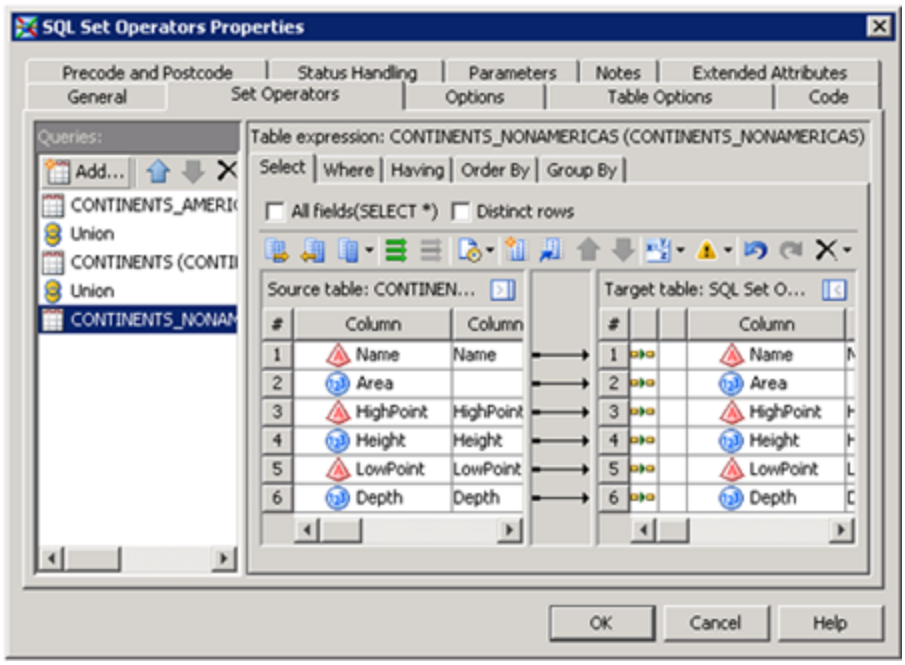
Create and Populate the Job

Perform the following steps to create and populate the job:

1. Create an empty job.
2. Select and drag an SQL Set Operators 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.
3. Open the properties window of the SQL Set Operators transformation.
4. Click **Set Operators**.
5. Click **Add** to access the **Table Query Selector** and select the first table. For example, you could select a table named CONTINENTS_AMERICAS.
6. Click the **Propagate columns** button on the toolbar on the Table for the newly added table. This action propagates the columns from the first table query to the output table.
7. Click **Add** as often as necessary to select the remaining tables. This sample job also contains tables named CONTINENTS and CONTINENTS_NONAMERICAS.

   The following display shows the tables selected as inputs to the SQL Set Operators transformation:

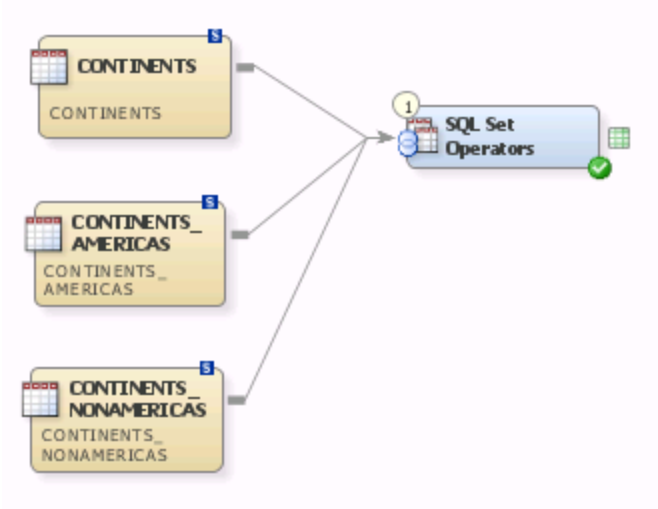
   Display 22.7 *SQL Set Operators Tables*



Note that the table queries are joined with union set operators by default.

The following display shows the resulting SQL set operators process flow in the sample job:

Display 22.8 *SQL Set Operators Process Flow*



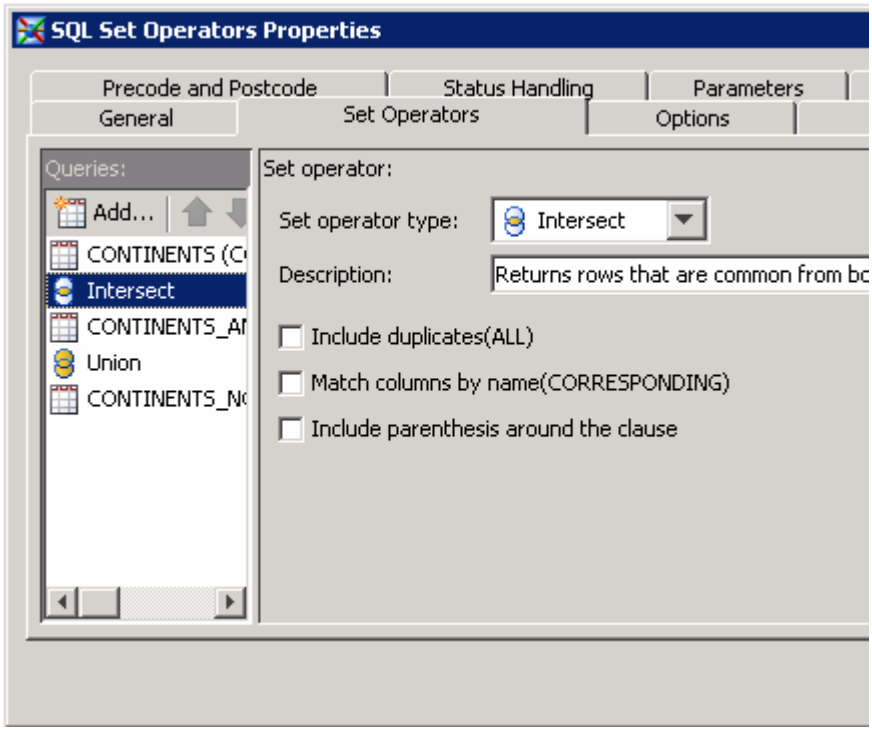
Configure the Queries

Perform the following steps to configure the table queries:

1. Click the set operator that you need to configure, such as the **Union** operator beneath the CONTINENTS_AMERICAS table in the sample job.
2. Select an operator type in the **Set operator type** field (such as **Intersect**).

   The following display shows the set operators section for the table in the sample job:

   Display 22.9 *Set Operators Section*

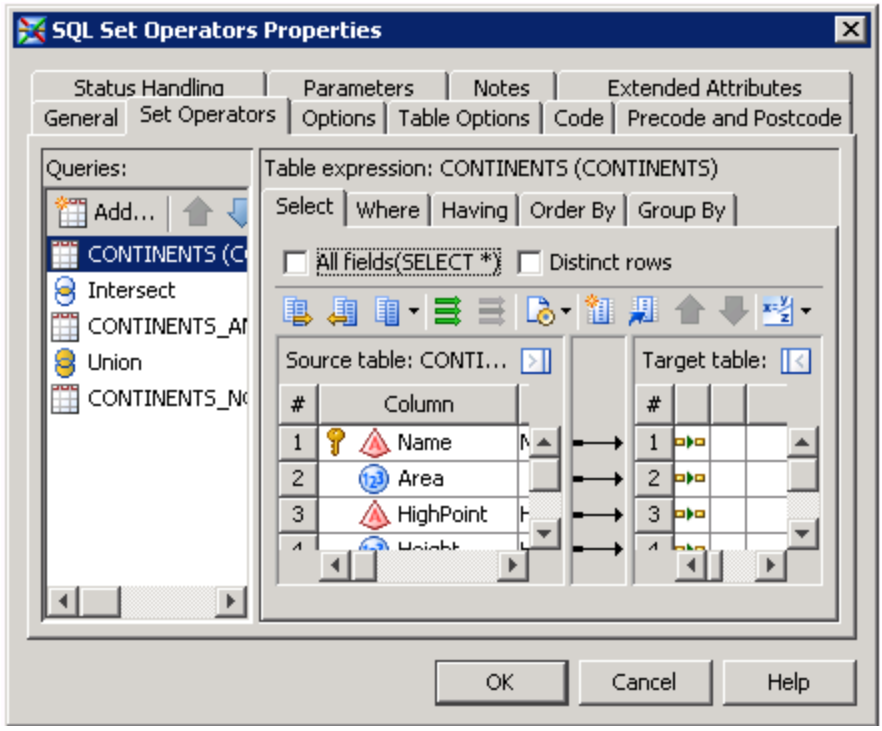


Note that you can appropriate options for each operator type. Repeat this process for all of the operators that you need to configure.

3. Review the SELECT statement for each query.

   The following display shows the SELECT expression for CONTINENTS_AMERICAS table in the sample job:

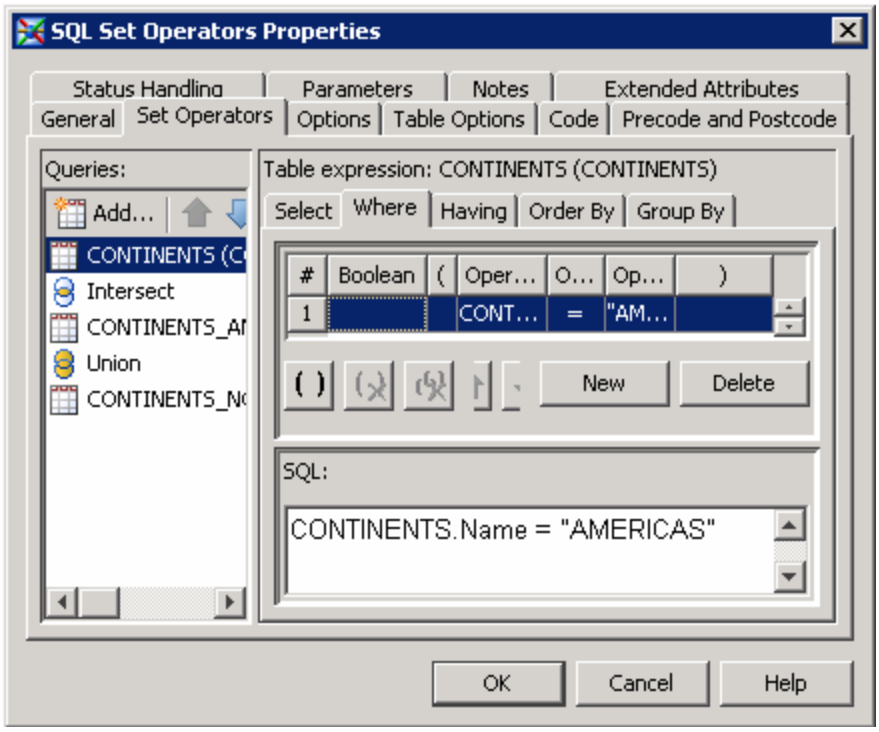
Display 22.10 *SELECT Statement for a Table*



4. Configure the WHERE, HAVING, ORDER BY, and GROUP BY statements for your table queries as needed. Note that the ORDER BY statement is permitted on the last query only. You can have only one ORDER BY statement in each SQL Set Operators transformation.

The following display shows the WHERE statement for the CONTINENTS_AMERICAS table.

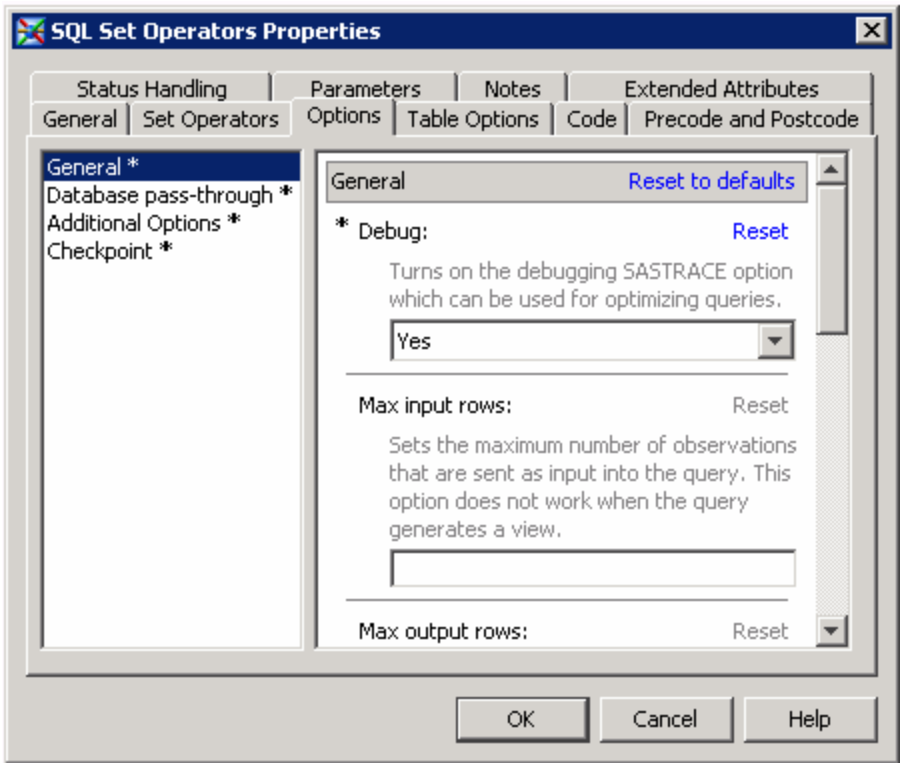
Display 22.11 *WHERE Statement for a Table*



5. Click **Options** to review the options for the SQL Set Operators transformation.

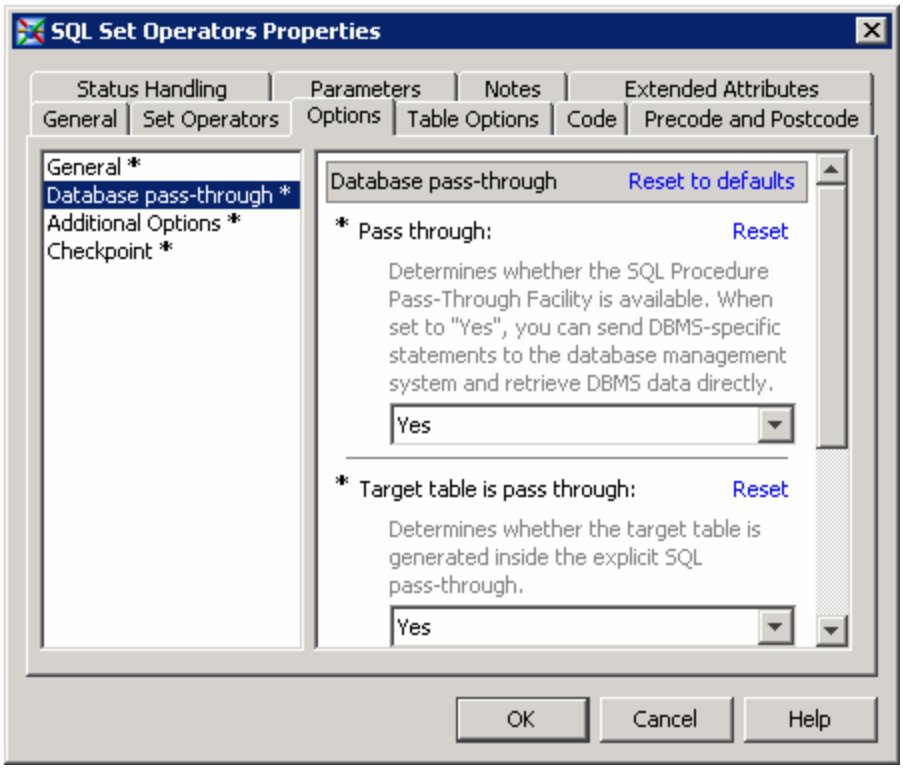
The following display shows debugging options. These options are located in the General section of the **Options** tab in the SQL Set Operators transformation in the sample job:

Display 22.12 *General Options Tab*



The following display shows pass-through options. These options are located in the Database pass-through section of the **Options** tab in the SQL Set Operators transformation in the sample job:

Display 22.13 *Database Pass-through Options Tab*



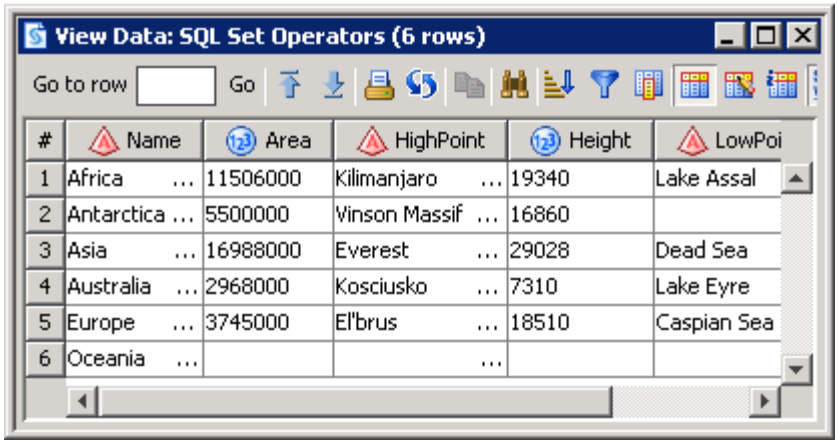
Run the Job and Review the Results

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 are displayed on the **Status** tab, read and respond to the messages as needed.
3. To view the output, right-click the output table and select **Open**. The output of the sample job is found in a temporary output table. You could also store the output in a permanent target table.

The following display shows the output of a set operators job:

Display 22.14 *Output from a Set Operators Job*



#	Name	Area	HighPoint	Height	LowPoi
1	Africa ...	11506000	Kilimanjaro ...	19340	Lake Assal
2	Antarctica ...	5500000	Vinson Massif ...	16860	
3	Asia ...	16988000	Everest ...	29028	Dead Sea
4	Australia ...	2968000	Kosciusko ...	7310	Lake Eyre
5	Europe ...	3745000	El'brus ...	18510	Caspian Sea
6	Oceania ...		...		

Note that the names of the row in the output include do not include the text AMERICAS. This text is present in some of the source tables.

Enabling Explicit Pass-Through Processing for Other SQL Transformations

Problem

You want to enable explicit pass-through processing for a Create Table transformation, Delete transformation, Insert Rows transformation, or an Update transformation.

Solution

You should enable explicit pass-through processing when you connect a database management system table to a Create Table transformation, Delete transformation, Insert Rows transformation, or Update transformation. Keep in mind that the functions that are unique to a database management system are resolved only in the context of explicit pass-through processing. If you rely on implicit pass-through processing, you will receive an error when the job is executed. Perform the following tasks:

- “Determine Whether Explicit Pass-Through Processing Is Possible” on page 502
- “Enable Explicit Pass-Through Processing” on page 503

Tasks

Determine Whether Explicit Pass-Through Processing Is Possible

The Delete, Execute, Insert Rows, Merge, and Update transformations are particularly useful for tables that originate from database management systems including DB2 9.7, Oracle 11g, SQL Server 2005, and Teradata 13. The systems must support the following commands:

- SQL Delete DML

- SQL Merge DML
- SQL Update DML
- SQL Create DML
- SQL Insert DML

In addition, Sybase (ASE/IQ) 12.5 supports non-SQL Merge transformations. Sybase 15.7 adds support for SQL Merge. The SQL Merge transformation does not support SAS tables. The Create Table, Delete, Execute, Insert Rows, and Update transformations do support SAS tables, but they might not support some functions such as sort.

Enable Explicit Pass-Through Processing

To enable explicit pass-through processing by default for new instances of most SQL transformations, select **Tools** ⇨ **Options** ⇨ **Job Editor Tab**, and then select the pass-through check box in the **Automatic Settings** area. This setting affects Join transformations and also any SQL transformation whose properties window includes a **Database pass-through** option on its **Options** tab. This includes SQL transformations such as Create Table, Insert Rows, Set Operators, Delete, and Update.

To enable explicit pass-through processing for individual transformations (Create Table, Insert Rows, Set Operators, Delete, and Update), open the properties window for the transformation and click the **Options** tab. Specify **Yes** for the **Database pass-through** option.

Chapter 23

Working with Iterative Jobs and Parallel Processing

About Iterative Jobs **505**

Creating and Running an Iterative Job **506**
Problem 506
Solution 506
Tasks 506

Creating a Parameterized Job **509**
Problem 509
Solution 509
Tasks 510

Creating a Control Table **512**
Problem 512
Solution 512
Tasks 512

About Parallel Processing **515**

Setting Options for Parallel Processing **516**
Problem 516
Solution 516
Tasks 516

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 23.1 *Iterative Job*

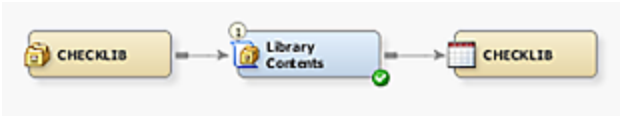


The process flow specifies that the inner Extract Balance job is executed multiple times, as specified by the Loop transformations and the CHECKLIB control table. The inner job is also called a parameterized job because it specifies its inputs and outputs as

parameters. For an example of how the steps in the iterative process are performed, see "Creating and Running an Iterative Job" on page 506.

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 23.2 *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.

See also "Usage Notes for Iterative Jobs" on page 662.

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 506
- "Variation: Add the Library Input and Library Contents Transformation Directly to a Job " on page 507
- "Run the Iterative Job and Examine the Results" on page 508

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 512 and "Creating a Parameterized Job" on page 509 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 23.3 *Completed Iterative Job*



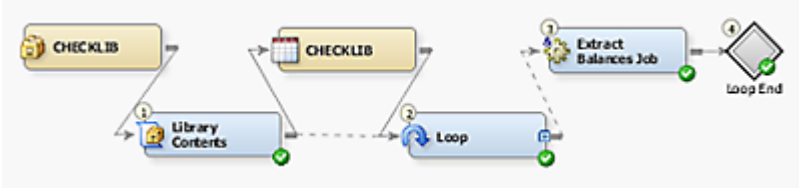
8. Open the **Loop Options** tab in the properties window for the Loop transformation. Select the **Execute iterations in parallel** check box. Also select the **One process for each available CPU node** check box in the **Maximum number of concurrent processes** group box.

   Note: You can set the iterative job to respond to a process error in the **Status Handling** tab of the Loop transformation. Specify the **Abort All** action for the **Error in Process** condition to abort the job. Specify the **Abort After Loop** action for the **Error in Process** condition to abort the job after the loop is completed.
9. Open the **Parameter Mapping** tab. Make sure that the appropriate value from the parameterized job is displayed in the Parameter Name column. Then, click the drop-down selection menu in the column for Mapped Source Column. Finally, select the source column that you want to map to the parameter.
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 23.4 *Control Table Job in an Iterative Job*



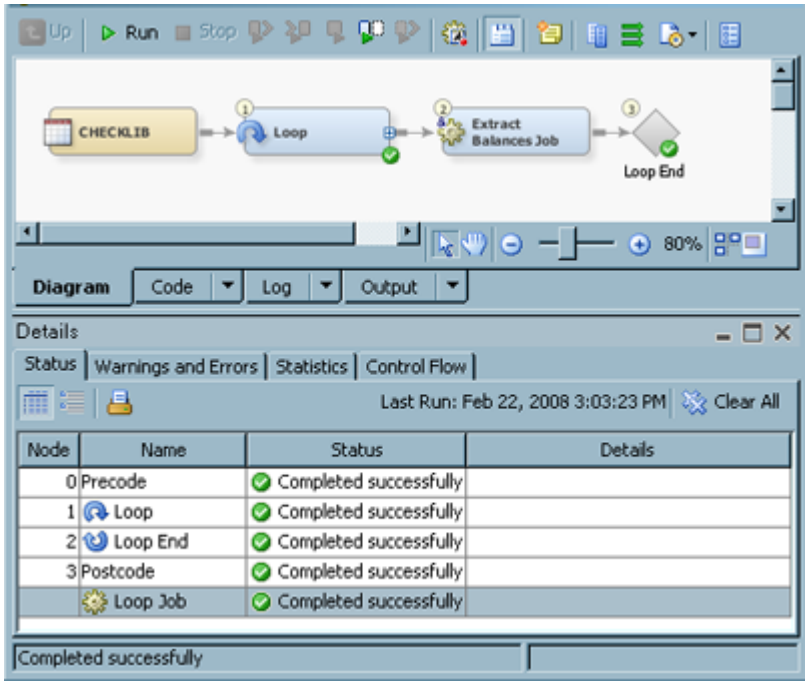
When the input library and the Library Contents transformation are added to the iterative job, the contents of the control table are dynamically generated each time that the iterative job is run. This arrangement ensures that the list of tables in the control table is refreshed each time that the job 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 512.

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:

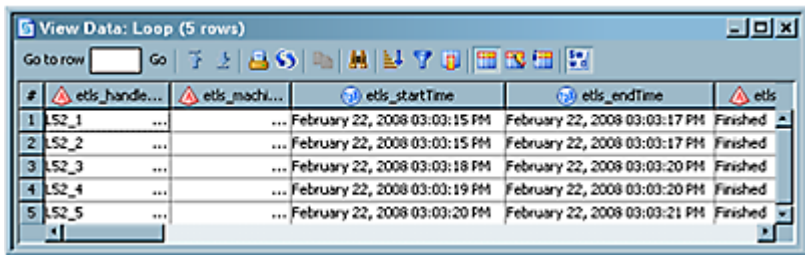
1. Run the iterative job. The following display shows a successfully completed sample job.

Display 23.5 *Sample Successful Iterative Job*



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.

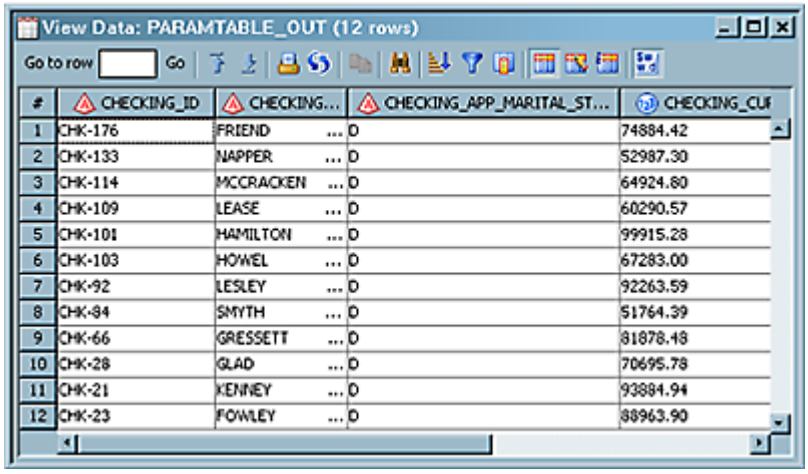
Display 23.6 *Loop Transformation Temporary Table*



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 23.7 *View of Target Table Output*



#	CHECKING_ID	CHECKING...	CHECKING_APP_MARITAL_ST...	CHECKING_CUI
1	CHK-176	FRIEND ...	0	74884.42
2	CHK-133	NAPPER ...	0	52987.30
3	CHK-114	MCCRACKEN ...	0	64924.80
4	CHK-109	LEASE ...	0	60290.57
5	CHK-101	HAMILTON ...	0	99915.28
6	CHK-103	HOWEL ...	0	67283.00
7	CHK-92	LESLEY ...	0	92263.59
8	CHK-84	SMYTH ...	0	51764.39
9	CHK-66	GRESSETT ...	0	81878.48
10	CHK-28	GLAD ...	0	70695.78
11	CHK-21	KENNEY ...	0	93884.94
12	CHK-23	FOWLEY ...	0	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 510
- "Set Input and Output Parameters" on page 510
- "Set Parameters for the Job" on page 511
- "Complete Parameterized Job Configuration" on page 511

Tasks

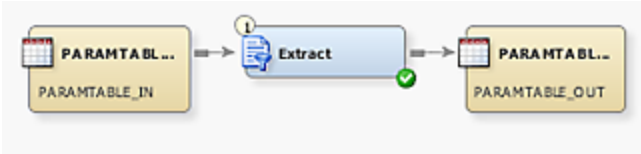
Create and Populate the Job

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 150.) You set the output parameter on this table.

A sample completed parameterized job is shown in the following example.

Display 23.8 *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 512
- "Create and Populate the Job" on page 513
- "Run the Job and Examine the Output" on page 514

Tasks

Create and Register the Control Table

If you have an existing control table, you can use it. If you do not 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**.

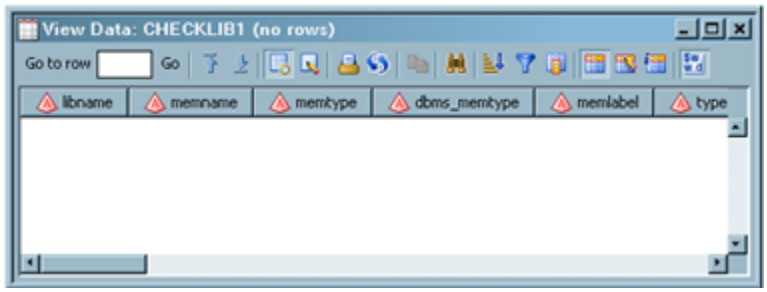
   The Source Editor window appears. Submit code similar to the following code:

```
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.
4. (Optional) You can confirm that the empty control table was created in physical storage. Right-click the metadata object for the table and select **Open**. A sample empty control table is shown in the following example.

Display 23.9 *View of Empty Control Table Output*



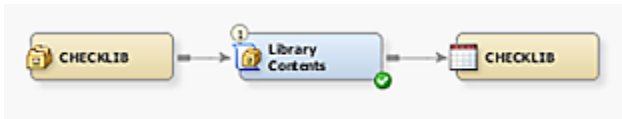
Create and Populate the Job

Perform the following steps to create and populate the job:

1. Create an empty job.
2. 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.
3. 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.
4. Drag the cursor from the library to the input port of the Library Contents transformation. This action connects the library to the 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 150.)
6. Drag the cursor from the output port of the Library Contents transformation to the target table. This action connects the transformation to the target.
7. 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 23.10 *Completed Control Table Job*



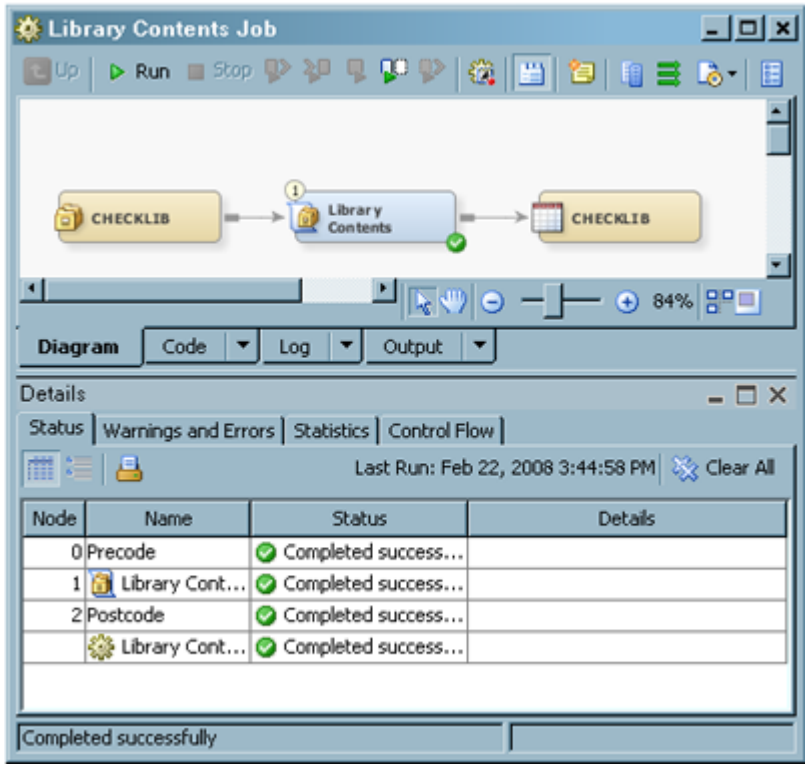
The library for the sample job is named CHECKLIB. The target table is also named CHECKLIB.

Run the Job and Examine the Output

Perform the following steps to run the control table job and examine its output:

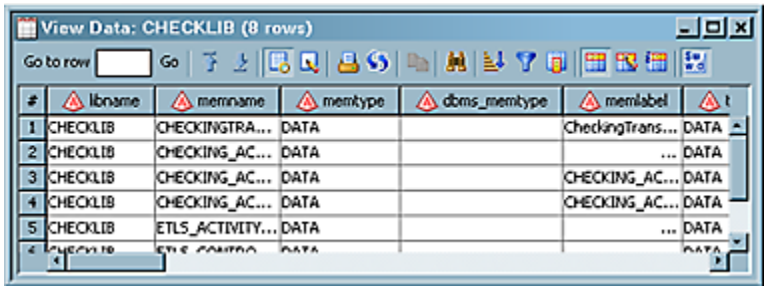
1. Run the job. The following display shows a successfully completed sample job.

Display 23.11 *Successful Sample Control Job*



2. 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 23.12 *View of Control Table Output*



Note that the 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 23.1 *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.

Computers Used for Parallel Processing	Requirements
Grid computing network	Specify an appropriate SAS Metadata Server to get the latest metadata object for a grid server. See the SAS Data Integration Studio chapter in the *SAS Intelligence Platform: Desktop Application Administration Guide*. Specify an appropriate SAS®9 Workspace Server in the metadata for the default. Grid software must be licensed. Define or add a grid server component to the metadata that points to the grid server installation. The controlling server machine must have both a grid server definition and a SAS Workspace Server definition as a minimum to be able to run your machines in a grid. It is recommended that you also have the SAS Metadata Server component accessible to the server definition where your grid machines are located. Install Platform Computing software to handle workload management for the grid.

Note: For additional information 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 23.2 *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 **Tools** ⇨ **Options** from the menu bar. Click the **Code Generation** tab. Specify the desired option.
Grid options set specification	Enables you to specify a collection of grid options, SAS options, and required resources that are associated with a particular SAS client application. A grid options set enables a SAS grid administrator to define a collection of options in SAS metadata that map to one or more SAS client applications. These options are automatically applied to workload submitted to the grid based on the identity of the user accessing the client application.	Select **Tools** ⇨ **Options** from the menu bar. Click the **SAS Server** tab or the **Code Generation** tab. Specify the desired option.
Workload specification	Enables you to select a default workload specification value for the selected server. The grid workload specification consists of a string value based on the grid server definition setup in SAS Management Console.	Select **Tools** ⇨ **Options** from the menu bar. Click the **SAS Server** tab or the **Code Generation** tab. Specify the desired option.
Signon options	Specifies options that users can set when the sign-on is performed to the grid server during the submit to grid method of executing. Some examples of sign-on options are cmacvar, connectremote, connectstatus, inheritlib, and tbufsize.	Select **Tools** ⇨ **Options** from the menu bar. Click the **SAS Server** tab. Specify the desired option.
Number of signon retries	Specifies the number of times to retry the sign-on to a grid server if a failure occurs.	Select **Tools** ⇨ **Options** from the menu bar. Click the **SAS Server** tab. Specify the desired option.

Option	Purpose	Task
Default maximum number of concurrent processes	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. Select from **One process for each available CPU node**, **Use this number**, or **Run all processes concurrently**.	Select **Tools** ⇨ **Options** from the menu bar. Click the **Code Generation** tab. Specify the desired option.

Table 23.3 *Local Options (affects the current job or transformation)*

Option	Purpose	Task
Enable parallel processing macros	When **YES** is selected, this option adds parallel processing macros to the code that is generated for the current job. Parallel processing macros are always included in the code that is generated for a Loop transformation.	Click the **Options** tab in the properties window for the job. Select **YES** or **NO** in the field for this option.
Grid options set specification	Enables you to specify a collection of grid options, SAS options, and required resources that are associated with a particular SAS client application. A grid options set enables a SAS grid administrator to define a collection of options in SAS metadata that map to one or more SAS client applications. These options are automatically applied to workload submitted to the grid based on the identity of the user accessing the client application.	Click the **Loop Options** tab in the properties window for the Loop transformation. Specify the desired option.
Workload specification	Enables you to select a default workload specification value for the selected server. The grid workload specification consists of a string value based on the grid server definition setup in SAS Management Console.	Click the **Loop Options** tab in the properties window for the Loop transformation. Specify the desired option.

Option	Purpose	Task
Wait for all processes to complete before continuing	Specifies that the application server waits for all iterations to complete before continuing with the job workflow.	Click the **Loop Options** tab in the properties window for the Loop transformation. Specify the desired option.
Maximum number of concurrent processes	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. Select from **One process for each available CPU node**, **Use this number**, or **Run all processes concurrently**.	Click the **Loop Options** tab in the properties window for the Loop transformation. Specify the desired option.
Grid supported job options	Enables you to define the name and value option pairs for the Grid Supported Job Options (JOBOPTS=) parameter.	Click the **Loop Options** tab in the properties window for the Loop transformation. Specify the desired option.

Chapter 24

Working with Slowly Changing Dimensions

About Slowly Changing Dimensions **522**
Slowly Changing Dimensions Defined 522
Types of Slowly Changing Dimensions 523
Transformations That Support Slowly Changing Dimensions 523
SCD Project Stages 524

About Dimension Tables **524**
About Change Tracking 524
About Change Detection and Loading for SCD 525
About Generated Keys 525
About Cross-Reference Tables 526
Two Methods for Generating the Change Digest Column 527
About Type 1 Updates 527

About Fact Tables **527**
Overview 527
About the Loading of Fact Tables with the Lookup Transformation 528

Usage Notes for Slowly Changing Dimensions **528**
Hadoop Tables That Are Also Hive Tables or Impala Tables Are Not Suitable for Slowly Changing Dimensions 528

Loading a Dimension Table with Type 1 Updates **528**
Problem 528
Solution 528
Tasks 529

Loading a Dimension Table with Type 1 and 2 Updates **535**
Problem 535
Solution 535
Tasks 536

Comparing Tables **538**
Problem 538
Solution 538
Tasks 539

Loading a Fact Table Using Dimension Table Lookup **545**
Problem 545
Solution 545
Tasks 545

Loading a Table and Adding a Surrogate Primary Key **551**
Problem 551
Solution 551

Tasks 551

Tracking Changes in Source Datetime Values 554
Problem 554
Solution 554
Tasks 554

Closing Out Rows in Datetime Change Tracking 556
Problem 556
Solution 556

About Slowly Changing Dimensions

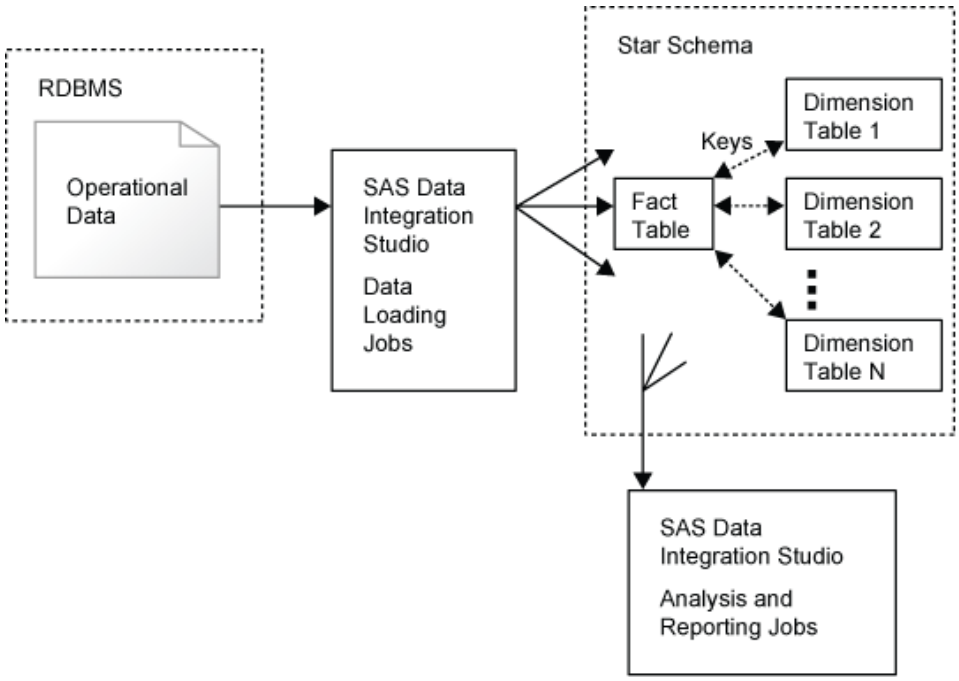
Slowly Changing Dimensions Defined

Slowly changing dimensions (SCD) is the name of a process that loads data into dimension tables. This data changes slowly, rather than changing on a time-based, regular schedule. 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 24.1 *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 1 Loader and SCD Type 2 Loader transformations. Fact tables are loaded with the Lookup transformation.

Types of Slowly Changing Dimensions

The three most common 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 1 Loader transformation performs Type 1 updates. You can use the SCD Type 2 Loader transformation to combine Type 1 and Type 2 updates in a single operation.

Type 2 SCD: full history of data changes
: maintains multiple records for each business key in the dimension table. The latest entry is the current entry for that business key. 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 of data changes
: 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 1 Loader
: inserts new rows, updates existing rows, and generates surrogate key values in a dimension table without maintaining a history of data changes. Each business key is represented by a single row in the dimension table.

SCD Type 2 Loader
: inserts new and Type 2 rows, updates existing rows, and generates surrogate key values in a dimension table. At the same, it maintains a full history of data changes. Each business key is represented by a current row and 0 through n number of closed out rows. The closed out rows enable change analysis over time.

Compare Tables
: detects differences between matching rows in specified columns in two tables. Outputs include changed, new, unchanged, and missing records tables. These outputs can be used as the basis for performing Type 1 or Type 2 updates in a dimension table.

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. This job 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

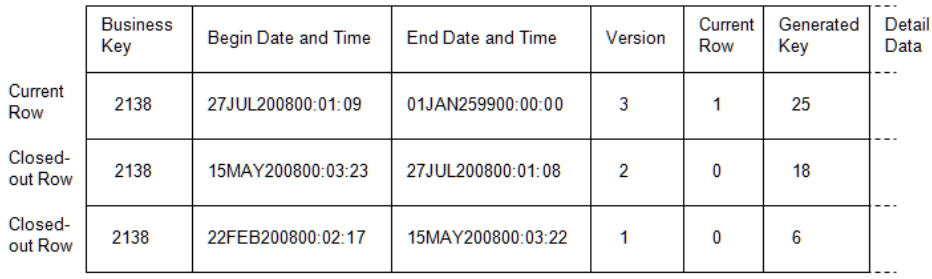
About Change Tracking

Dimension tables that are loaded with the SCD Type 2 Loader consist of a surrogate key column, a business key column, change tracking columns, and any number of detail data columns. The surrogate 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 surrogate key can be defined to consist of more than one column, as determined by the structure of the source data. A surrogate key is typically a system-generated value that contains no semantic meaning. It is almost always a numeric value that you can use to improve join performance between fact and dimension tables.

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 24.2 *Structure of an SCD Dimension Table*



Tracking changes by version number increments a counter when a new row is added. The current row has the highest version number for that business key. The version number for new business keys 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 surrogate key column, the values for which are generated by the SCD Type 2 Loader. The generated surrogate key is necessary in order to uniquely identify individual rows in the dimension table. The generated surrogate 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. If changes are detected in the Type 1 columns in Type 1 upgrades, the source data overwrites the target data in the current row. The data is overwritten even when data differences are not detected in the Type 2 columns.

About Generated Keys

The SCD Type 2 Loader enables you to generate surrogate key values when you load a dimension table. The generated surrogate key values replace the business key as the

primary key. This is because the business key from the source table identifies the member, not the unique row in the dimension table.

You can configure a simple surrogate key in the **Generated Keys** tab of the SCD Type 2 Loader. This surrogate key 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**.

Two Methods for Generating the Change Digest Column

The SCD Type 2 Loader supports two methods for generating the change digest column (DIGEST_VALUE column) in a cross-reference table. To specify one of these methods, open the properties window for that transformation and select **Options** ⇨ **SCD Options**. Select **v.1.1** or **v2.1** in the **Change digest version** field.

The v1.1 method is the default. The v2.1 method uses a different method to concatenate the data columns that were selected for change detection. Try v2.1 if the SCD Type 2 transformation does not detect changes in certain scenarios.

For example, suppose that two consecutive Type 2 columns with data type char are loaded on day X with the following values: col_1="AB" , col_2="C"

The following delta record might contain these values: col_1="A" , col_2="BC"

The v1.1 method uses string handling functions that would concatenate these values into an intermediate string value of "ABC" for both the original record and the delta record. As a result, identical DIGEST_VALUEs would be incorrectly generated for both records, and the change will not be detected. If you encounter this problem, try the v2.1 method, which uses different string handling functions.

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.

Usage Notes for Slowly Changing Dimensions

Hadoop Tables That Are Also Hive Tables or Impala Tables Are Not Suitable for Slowly Changing Dimensions

Due to the limitations of the Hadoop environment, Hadoop tables that are also Hive tables or Impala tables are not suitable for Slowly Changing Dimensions.

Loading a Dimension Table with Type 1 Updates

Problem

You want to load a dimension table using type 1 updates. You need to generate a surrogate key for each target row and optimize performance for large source tables.

Solution

You can create a job that includes the SCD Type 1 Loader transformation. The SCD Type 1 Loader transformation supports either a direct in-memory lookup (hash object) or a sequential disk-based lookup (DATA step merge). The hash lookup method offers better performance than the DATA step merge lookup method, but it requires that the entire cross-reference table fit into system memory. If sufficient system memory is not available, the DATA step merge lookup method can be used instead.

The sample job uses the direct lookup method. For information about the data merge method, see “Use a Sequential Data Merge” on page 535.

Perform the following tasks:

- "Create and Populate the Job" on page 529
- "Configure the Job" on page 529
- "Run the Job and View the Output" on page 532

Tasks

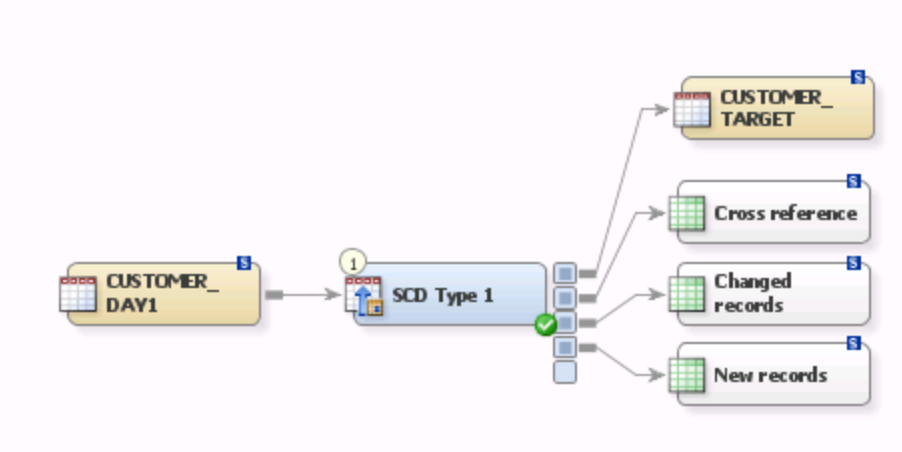
Create and Populate the Job

Perform the following steps to create and populate the job:

1. Create an empty SAS Data Integration Studio job.
2. Drag the SCD Type 1 Loader transformation to the **Diagram** tab of the job.
3. Locate a source table for the job, such as CUSTOMER_DAY1. Drag it to the left side of the **Diagram** tab of the job.
4. Drag the cursor from the source table output of the SCD Type 1 Loader transformation to the source table. This action connects the transformation to the table.
5. Locate a target table for the job, such as CUSTOMER_TARGET.
6. Drag the cursor from the target table output of the SCD Type 1 Loader transformation to the target table. This action connects the transformation to the table.

   The following display shows the completed process flow:

Display 24.1 *SCD Type 1 Loader Process Flow*



Configure the Job

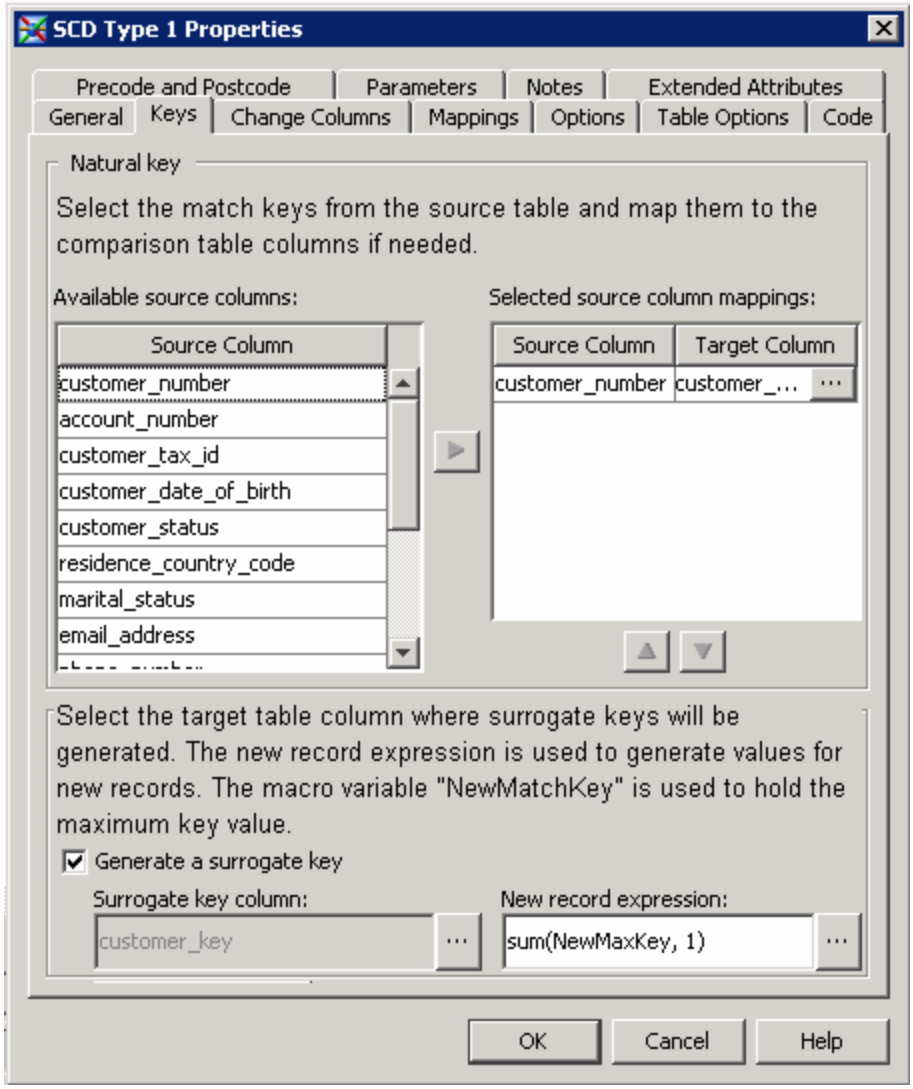
Perform the following steps to configure the job:

1. Open the properties window for the SCD Type 1 Loader transformation and select the **Keys** tab.
2. Select the match keys for your job in the **Available source columns** field and move them to the **Selected source column mappings** field. The sample job uses the column representing the customer number for each record. If necessary, you can map these columns to the appropriate target table columns on the **Mappings** tab.

3. Select the **Generate a surrogate key** check box. Use the browse button to enter the column in the target table that you will use as your surrogate key into the **Surrogate key column** field. The optional surrogate key ensures that each record is uniquely identified. The surrogate key is calculated in the expression found in the **New record expression** field. If you need to define the max key used in this expression, click **Define Max Key**.

   The **Keys** tab is shown in the following display:

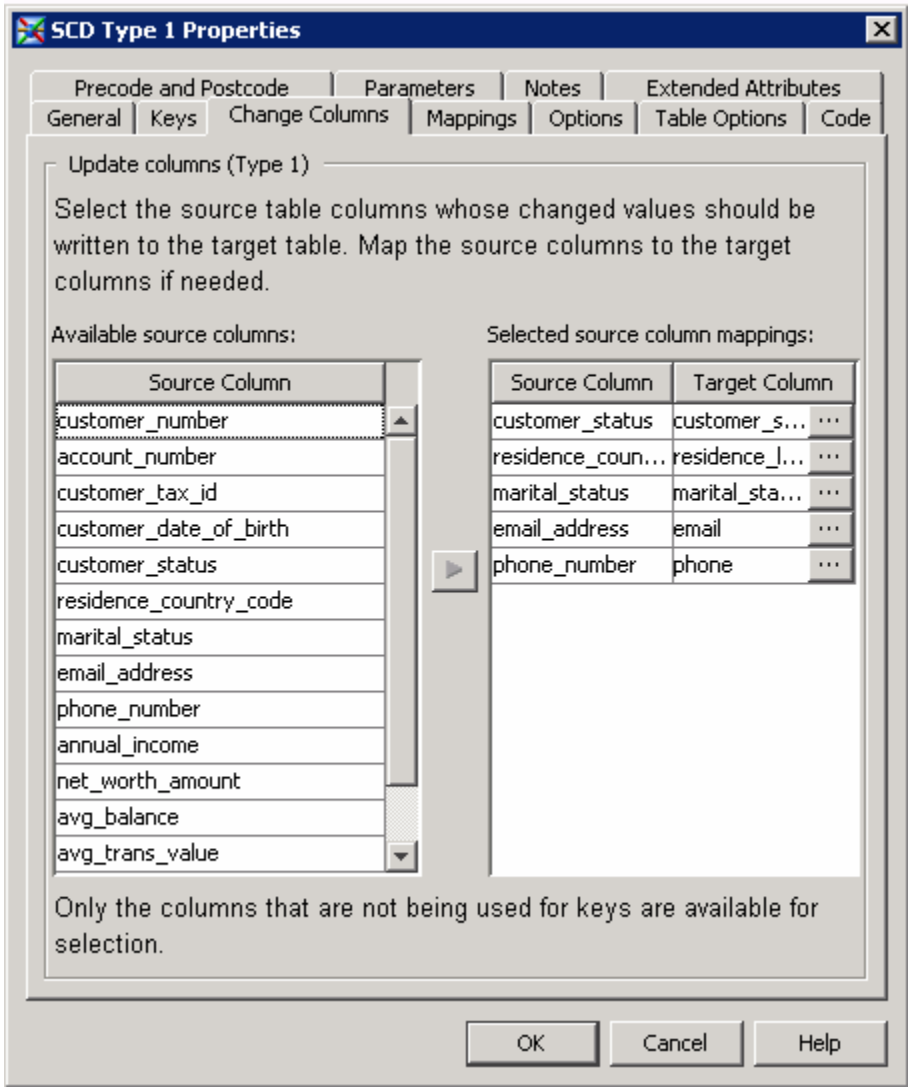
   Display 24.2 *Keys Tab*



4. Click **Change Columns** to select the change columns for your job.

5. Select the change columns for your job in the **Available source columns** field and move them to the **Selected source column mappings** field. The sample job uses the columns shown in the following display:

Display 24.3 *Change Columns Tab*



If necessary, you can map these columns to the appropriate target table columns on the **Mappings** tab.

6. Click **Options**. Set the **Use direct lookup (hash table)** option in the General category to **Yes**. This setting ensures that the SCD Type 1 Loader transformation will use the hash lookup method, which enhances processing performance. Note that you can also adjust the hash object's internal table size. Selecting the appropriate value can result in significant performance increases. The internal table size for the sample job is set to *10*.

   The value of HASHEXP is used as a power-of-two exponent to create the hash table size. For example, a value of 4 for HASHEXP equates to a hash table size of 24, or 16. The maximum value for HASHEXP is 20. You should specify the hash table size relative to the amount of data in the hash object in order to maximize the efficiency of the hash object lookup routines. Try different HASHEXP values until you get the best result.

   Note: If your source tables contain any duplicate business keys, remove the keys before you use the hash lookup method by adding a Sort transformation between each source and the transformation. If the source table does not contain duplicate business keys, a sort is not required. A sort is always required, however, when a

DATA step merge lookup is performed. Finally, the cross-reference table should be redirected to a register library to improve performance. This approach prevents you from having to recreate the change-digest for each record in the target table on each job execution.

7. Click **OK** to save the properties for the transformation.

   Note that you can save the temporary output tables for cross references, changed records, and new records that are attached to the SCD Type 1 Loader transformation. This approach creates permanent tables to collect these outputs. Right-click each table and click **Register Table**. Then, use the Register Table window to perform the registration.

   You should not delete these temporary output tables and replace them with previously registered tables. In addition, you should always wait until after you have performed the following SCD configuration steps:

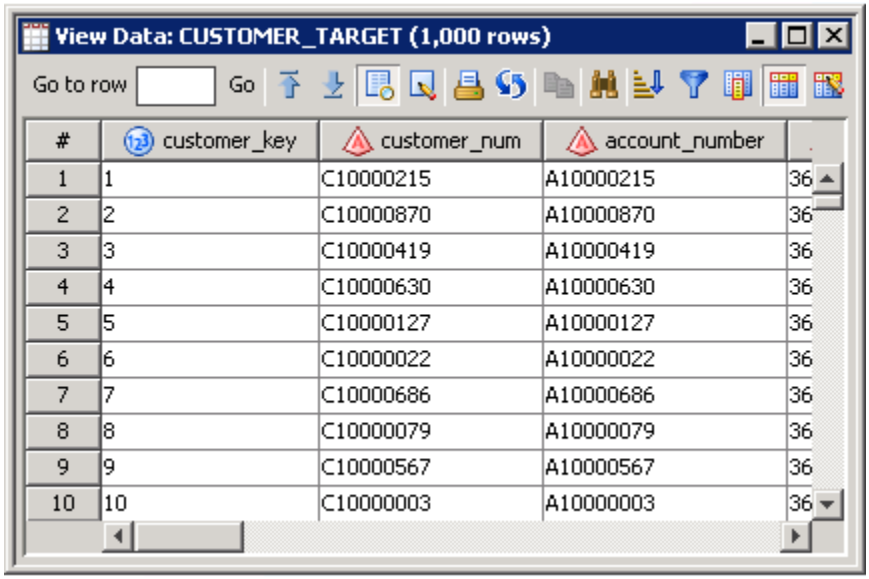
   - adding natural and surrogate keys
   - selecting surrogate keys and a lookup table
   - connecting the target table
   - selecting change columns
   - selecting a last-update column in the **Last update date** field in the **General** section on the **Options** tab
   - selecting a create data in the **Load time column** field in the **Additional Loader Options** section on the **Options** tab

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. If the job completes successfully, you can review the output. Right-click the target table and click **Open** in the pop-up menu.
4. Review the output displayed in the View Data window for the target table, as shown in the following display:

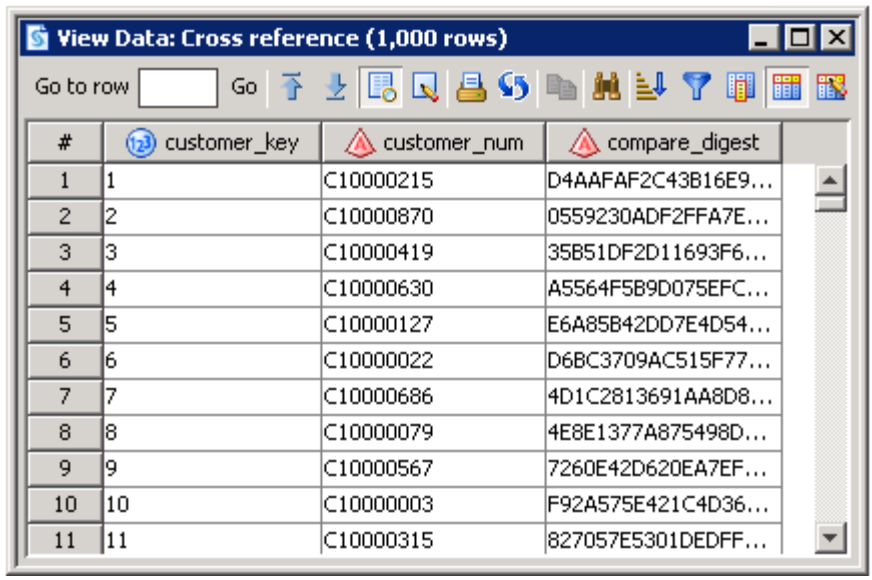
Display 24.4 *Target Table Output*


View Data: CUSTOMER_TARGET (1,000 rows)

Go to row Go

#	customer_key	customer_num	account_number	
1	1	C10000215	A10000215	36
2	2	C10000870	A10000870	36
3	3	C10000419	A10000419	36
4	4	C10000630	A10000630	36
5	5	C10000127	A10000127	36
6	6	C10000022	A10000022	36
7	7	C10000686	A10000686	36
8	8	C10000079	A10000079	36
9	9	C10000567	A10000567	36
10	10	C10000003	A10000003	36

5. Close the View Data window when you are finished.
6. Right-click the cross references table and click **Open** in the pop-up menu.
7. Review the output displayed in the View Data window for the cross reference records table, as shown in the following display:

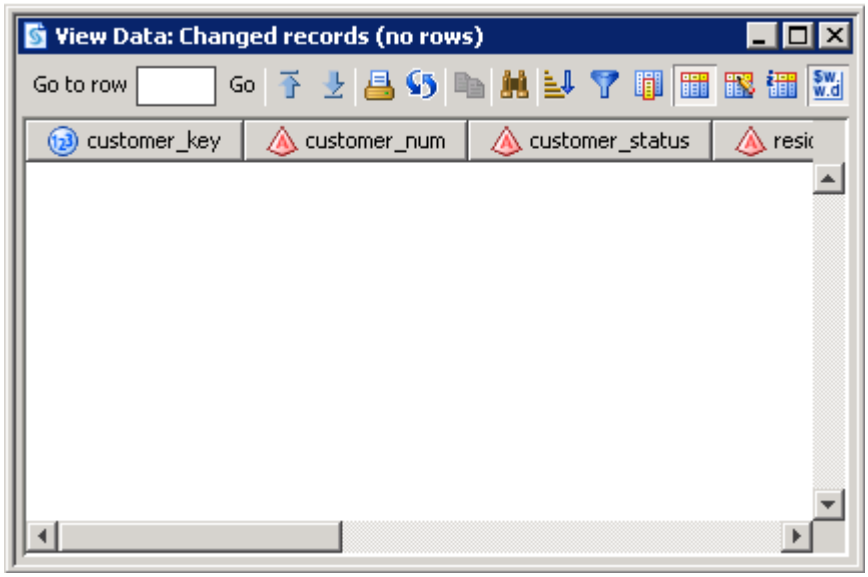

View Data: Cross reference (1,000 rows)

Go to row Go

#	customer_key	customer_num	compare_digest
1	1	C10000215	D4AAFAF2C43B16E9...
2	2	C10000870	0559230ADF2FFA7E...
3	3	C10000419	35B51DF2D11693F6...
4	4	C10000630	A5564F5B9D075EFC...
5	5	C10000127	E6A85B42DD7E4D54...
6	6	C10000022	D6BC3709AC515F77...
7	7	C10000686	4D1C2813691AA8D8...
8	8	C10000079	4E8E1377A875498D...
9	9	C10000567	7260E42D620EA7EF...
10	10	C10000003	F92A575E421C4D36...
11	11	C10000315	827057E5301DEDFF...

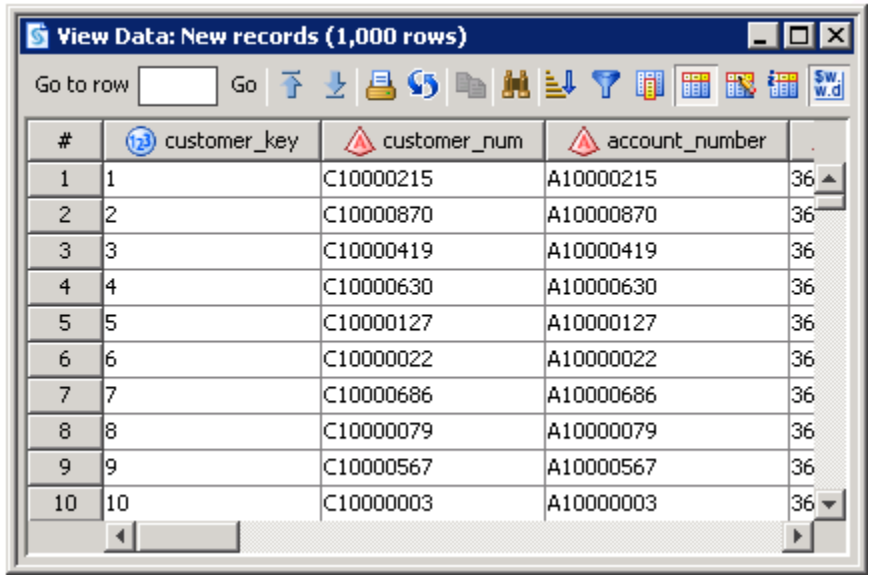
8. Close the View Data window when you are finished.
9. Right-click the changed records table and click **Open** in the pop-up menu.
10. Review the output displayed in the View Data window for the changed records table, as shown in the following display:

Display 24.5 *Changed Records Table Output*



11. Close the View Data window when you are finished.
12. Right-click the new records table and click **Open** in the pop-up menu.
13. Review the output displayed in the View Data window for the new records table, as shown in the following display:

Display 24.6 *New Records Table Output*



#	customer_key	customer_num	account_number	
1	1	C10000215	A10000215	36
2	2	C10000870	A10000870	36
3	3	C10000419	A10000419	36
4	4	C10000630	A10000630	36
5	5	C10000127	A10000127	36
6	6	C10000022	A10000022	36
7	7	C10000686	A10000686	36
8	8	C10000079	A10000079	36
9	9	C10000567	A10000567	36
10	10	C10000003	A10000003	36

14. Close the View Data window when you are finished.

    Note the following results:

    - 1000 target table rows
    - 1000 cross reference rows
    - 0 changed rows
    - 1000 new rows

Use a Sequential Data Merge

Perform the following steps to configure the SCD Type 1 Loader transformation to use a sequential data merge:

1. Open the properties window for the SCD Type 1 Loader transformation.
2. Click **Options**. Set the **Use direct lookup (hash table)** option in the General category to **No**. This setting saves memory at a potential cost to processing performance.
3. Click **OK** to save your settings and close the properties window.

   Note: The DATA step merge lookup method requires the source table to be presorted by business key with duplicate business keys removed. If the source table is not presorted by business key or the source table contains duplicate business keys, you should add a Sort transformation before you sort and remove duplicate business keys with the SCD Type 1 Loader transformation. In the Sort transform, navigate to the **Options** tab and change the first SAS Sort option entitled **Remove duplicate records** to **Remove rows with duplicate keys (NODUPKEY)**. You should remove duplicate business keys from source tables before you use the SCD Type 1 Loader transformation to avoid unexpected results.
4. Insert a Sort transformation between the source table and the SCD Type 1 Loader transformation.
5. Open the Sort transformation and click **Sort Columns**.
6. Move the columns that you want to use for the match keys in the source table to the **Sort by columns** field.
7. Click **OK** to save your settings and close the properties window.

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 536
- "Configure the SCD Type 2 Loader" on page 537
- "Run the Job and View the Output" on page 537

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.

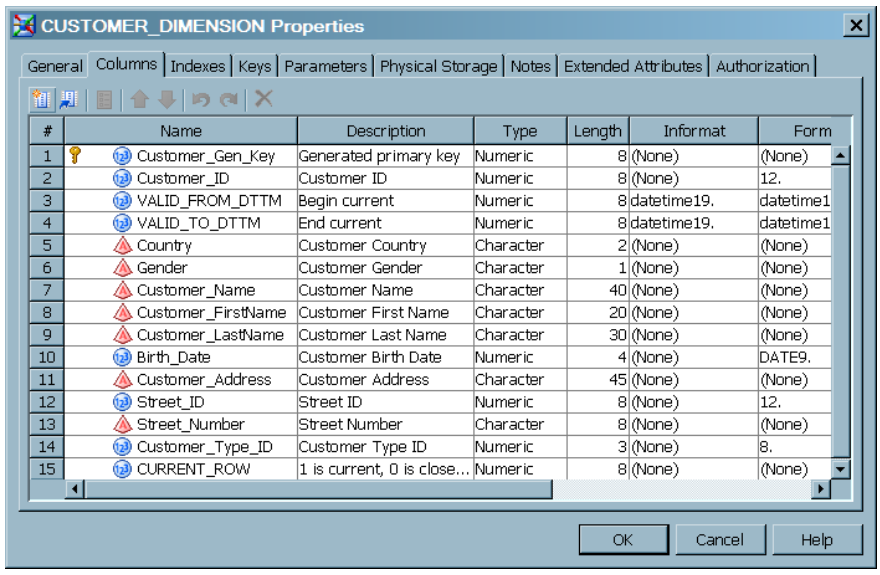
   VALID_TO_DTTM
   receives end datetime values.

   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 24.7 *Target Column Properties*



#	Name	Description	Type	Length	Informat	Form
1	Customer_Gen_Key	Generated primary key	Numeric	8	(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)
6	Gender	Customer Gender	Character	1	(None)	(None)
7	Customer_Name	Customer Name	Character	40	(None)	(None)
8	Customer_FirstName	Customer First Name	Character	20	(None)	(None)
9	Customer_LastName	Customer Last Name	Character	30	(None)	(None)
10	Birth_Date	Customer Birth Date	Numeric	4	(None)	DATE9.
11	Customer_Address	Customer Address	Character	45	(None)	(None)
12	Street_ID	Street ID	Numeric	8	(None)	12.
13	Street_Number	Street Number	Character	8	(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)

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 24.8 *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.

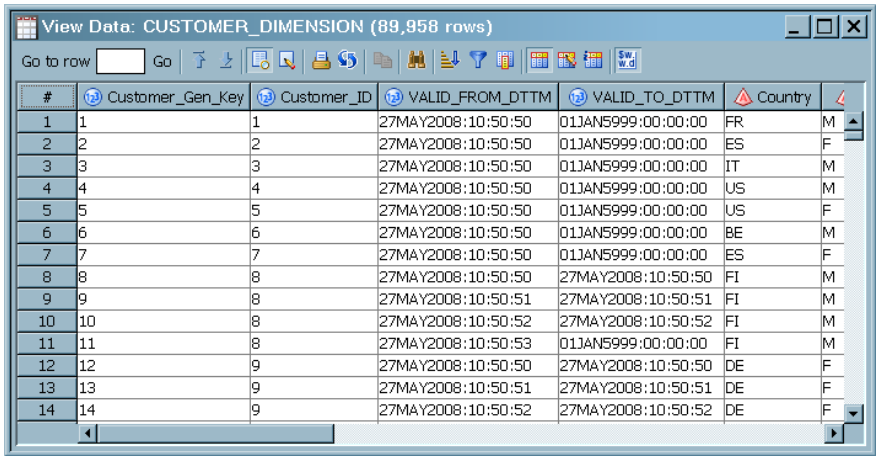
Run the Job and View the Output

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 24.9 *Key Columns and Change Tracking Columns in the Sample Target Table*



View Data: CUSTOMER_DIMENSION (89,958 rows)

Go to row Go

#	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

Comparing Tables

Problem

You want to detect changes between two tables such as an update table and a master table. For example, a PRICE_COST update table could contain the unit cost and unit price of all products currently being promoted through a channel. At the same time, a PRICE_COST_HIST table could serve as the master table.

You need to accommodate the following inputs and outputs for the comparison:

- Input Port 1: The source table
- Input Port 2: The comparison table
- Output Port 1: Matched: Changed Records
- Output Port 2: Unmatched: New Records (source table only)
- Output Port 3: Matched: Unchanged Records
- Output Port 4: Unmatched: Missing Records (comparison table only)

Solution

You can create a job that contains the Compare Tables transformation. The transformation generates a variety of output for matched and unmatched records. The Compare Tables transformation supports either a direct in-memory lookup (hash object) or a sequential disk-based lookup (DATA step merge). The hash lookup method offers better performance than the DATA step merge lookup method, but it requires that the entire cross-reference table fit into system memory. If sufficient system memory is not available, the DATA step merge lookup method can be used instead.

The sample job uses the direct lookup method. See "Use a Sequential Data Merge" on page 544 for information about the data merge method.

Perform the following tasks:

- "Create and Populate the Job" on page 539
- "Configure the Job" on page 540
- "Run the Job and View the Output" on page 542

You could use Compare Tables to update the tables as follows:

1. New PRICE_COST records are added to PRICE_COST_HIST.
2. Any updates to UNIT_PRICE or UNIT_COST for a PRICE_COST record cause the current historical record to be logically deleted and the updated record is added.
3. Records that now longer appear on the current PRICE_COST table are logically deleted from the PRICE_COST_HIST table.
4. All logically deleted records are available for query or reactivation, because they are retained in the table but are marked as deleted.

The transformation can handle new records, changed records, missing records, and unchanged records as output. You can choose to retain or delete any of the possible outputs as needed to increase efficiency. The transformation also generates its results in a single pass of the data.

Tasks

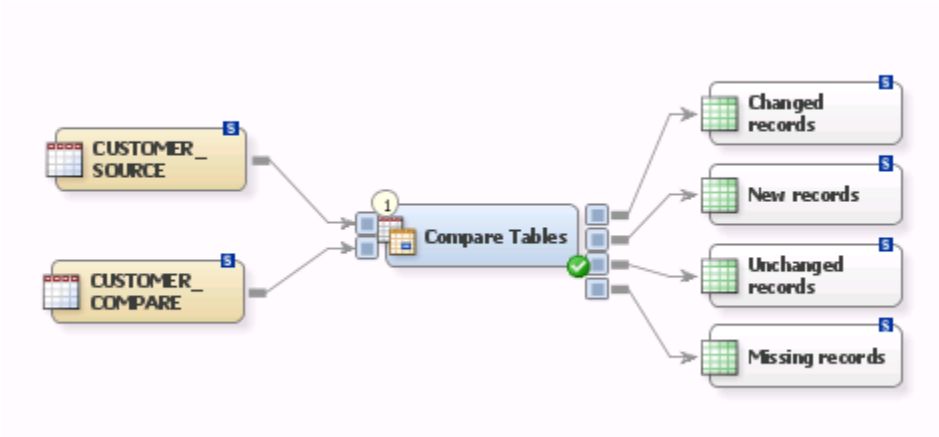
Create and Populate the Job

Perform the following steps to create and populate the job:

1. Create an empty SAS Data Integration Studio job.
2. Drag the Compare Tables transformation to the **Diagram** tab of the job.
3. Locate the tables that you want to compare, such as CUSTOMER_SOURCE and CUSTOMER_COMPARE. Drag them to the right side of the **Diagram** tab of the job.
4. Drag the cursor from each comparison table to Compare Tables transformation. This action connects the comparison tables to the Compare Tables transformations.

The following display shows the completed process flow:

Display 24.10 *Compare Tables Process Flow*



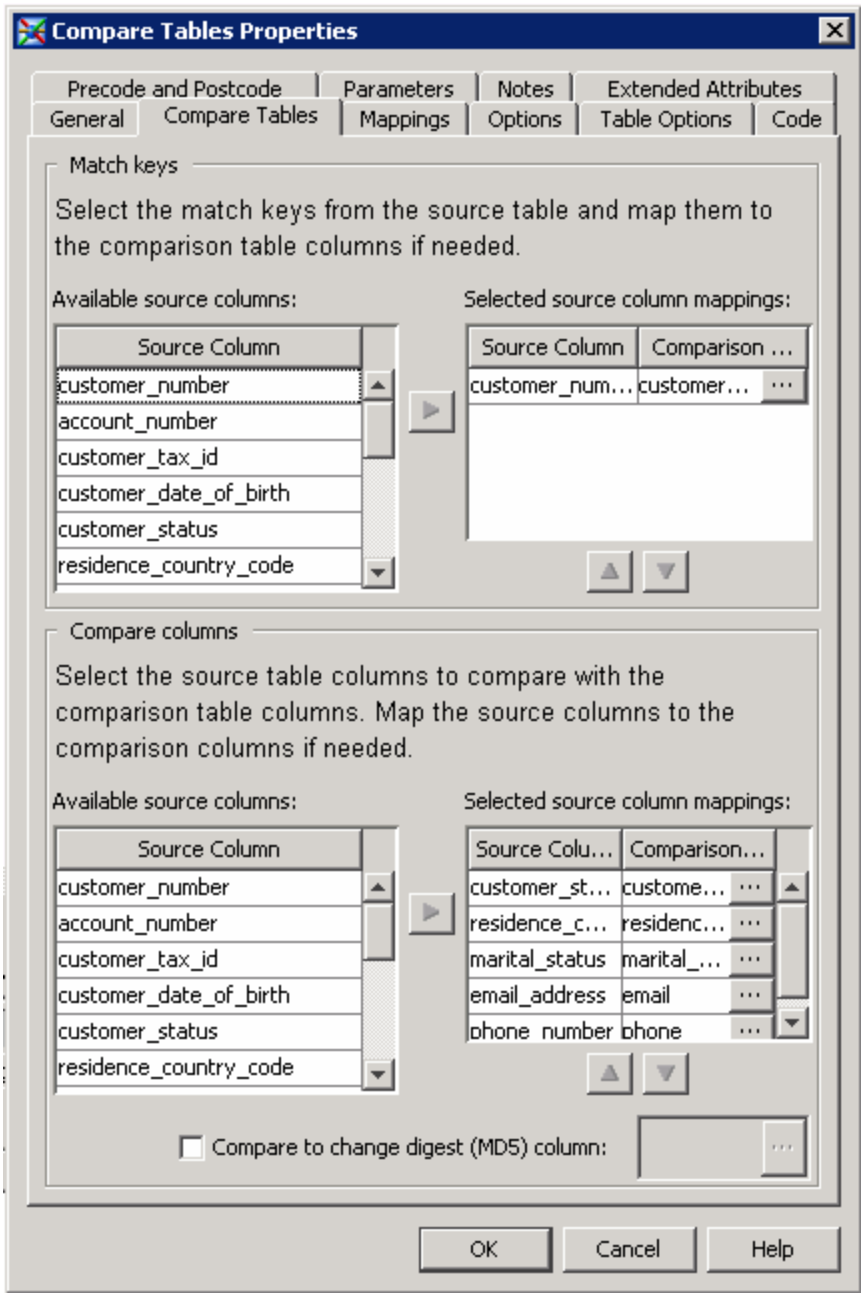
Configure the Job

Perform the following steps to configure the job:

1. Open the properties window for the Compare Tables transformation and select the **Compare Tables** tab.
2. Select the match keys for the comparison in your job in the **Available source columns** field and move them to the **Selected source column mappings** field. The sample job uses the column representing the customer number for each record. If necessary, you can map the column to the appropriate target table column on the **Mappings** tab.
3. Select one or more source table columns to compare with the comparison table columns. You can also map these columns on the **Mappings** tab, if needed.

Several columns that contain customer contact information are selected in the sample job, as shown in the following display:

Display 24.11 *Compare Tables Properties Window*



4. Click **Options**. Set the **Use direct lookup (hash table)** option in the General category to **Yes**. This setting ensures that the Compare Tables transformation will use the hash lookup method, which enhances processing performance. Note that you can also adjust the hash object's internal table size. Selecting the appropriate value can result in significant performance increases. The internal table size for the sample job is set to *10*.

   The value of HASHEXP is used as a power-of-two exponent to create the hash table size. For example, a value of 4 for HASHEXP equates to a hash table size of 24, or 16. The maximum value for HASHEXP is 20. You should specify the hash table size relative to the amount of data in the hash object in order to maximize the efficiency

of the hash object lookup routines. Try different HASHEXP values until you get the best result.

Note: If your source tables contain any duplicate business keys, remove the keys before you use the hash lookup method by adding a Sort transformation between each source and the transformation. If the source table does not contain duplicate business keys, a sort is not required. A sort is always required, however, when a DATA step merge lookup is performed. Finally, the cross-reference table should be redirected to a register library to improve performance. This approach prevents you from having to recreate the change-digest for each record in the target table on each job execution.

5. Click **OK** to save your settings and close the properties window.

   Note that you can register the temporary output tables for changed records, new records. unchanged records, and missing records that are attached to the Compare Tables transformation. This capability enables you to create permanent tables to collect these outputs. Right-click each table and click **Register Table**. Then, use the Register Table window to perform the registration.

   Note: You should not delete these temporary output tables and replace them with previously registered tables. In addition, you should always wait until after you have performed the following compare tables configuration:

   - adding match keys
   - selecting compare columns

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. If the job completes successfully, you can review the output. Right-click the changed records output table and click **Open** in the pop-up menu.
4. Review the output displayed in the View Data window for the changed records table, as shown in the following display:

Display 24.12 *Changed Records Output*

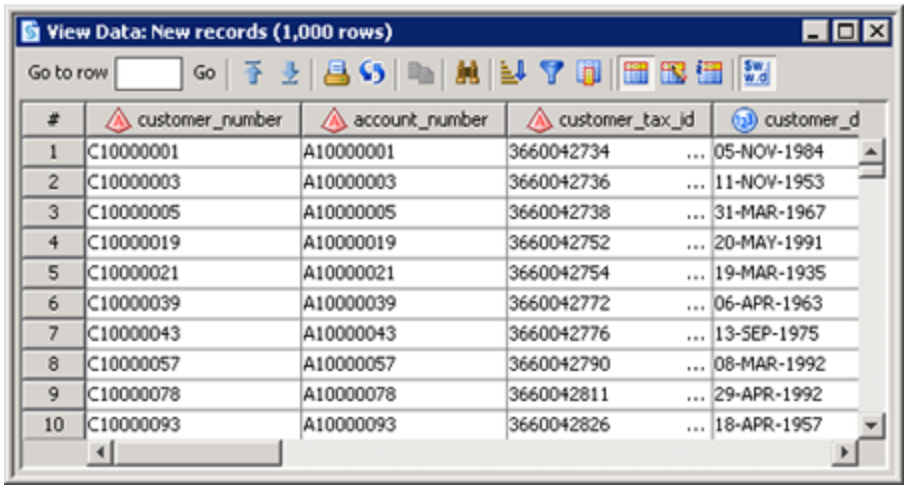
View Data: Changed records (1,000 rows)

Go to row Go

#	customer_number	account_number	customer_tax_id		customer_date
1	C10000012	A10000012	3660042745	...	13-JUN-1933
2	C10000023	A10000023	3660042756	...	17-DEC-1951
3	C10000024	A10000024	3660042757	...	03-OCT-1986
4	C10000027	A10000027	3660042760	...	03-SEP-1992
5	C10000029	A10000029	3660042762	...	31-MAY-1963
6	C10000045	A10000045	3660042778	...	27-JAN-1939
7	C10000055	A10000055	3660042788	...	02-APR-1938
8	C10000062	A10000062	3660042795	...	29-MAY-1949
9	C10000070	A10000070	3660042803	...	12-JAN-1964
10	C10000077	A10000077	3660042810	...	06-APR-1951

5. Review the output displayed in the View Data window for the new records table, as shown in the following display:

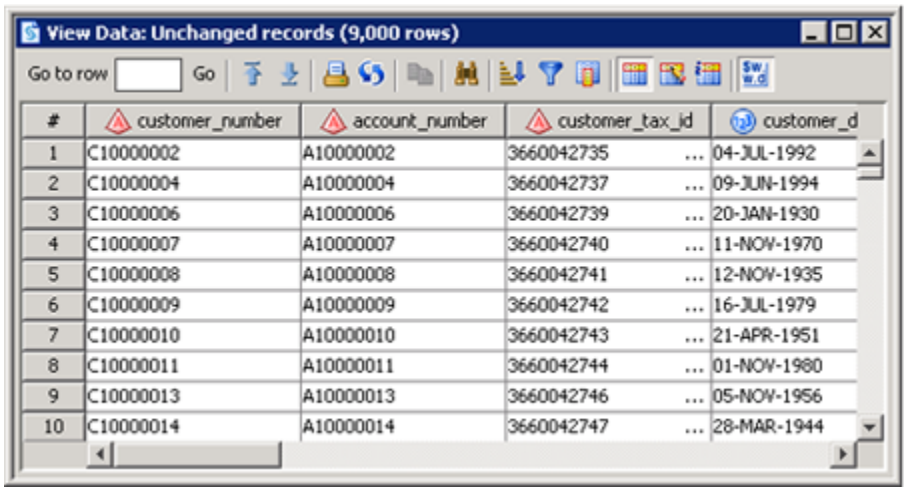
Display 24.13 *New Records Output*



#	customer_number	account_number	customer_tax_id		customer_d
1	C10000001	A10000001	3660042734	...	05-NOV-1984
2	C10000003	A10000003	3660042736	...	11-NOV-1953
3	C10000005	A10000005	3660042738	...	31-MAR-1967
4	C10000019	A10000019	3660042752	...	20-MAY-1991
5	C10000021	A10000021	3660042754	...	19-MAR-1935
6	C10000039	A10000039	3660042772	...	06-APR-1963
7	C10000043	A10000043	3660042776	...	13-SEP-1975
8	C10000057	A10000057	3660042790	...	08-MAR-1992
9	C10000078	A10000078	3660042811	...	29-APR-1992
10	C10000093	A10000093	3660042826	...	18-APR-1957

6. Review the output displayed in the View Data window for the unchanged records table, as shown in the following display:

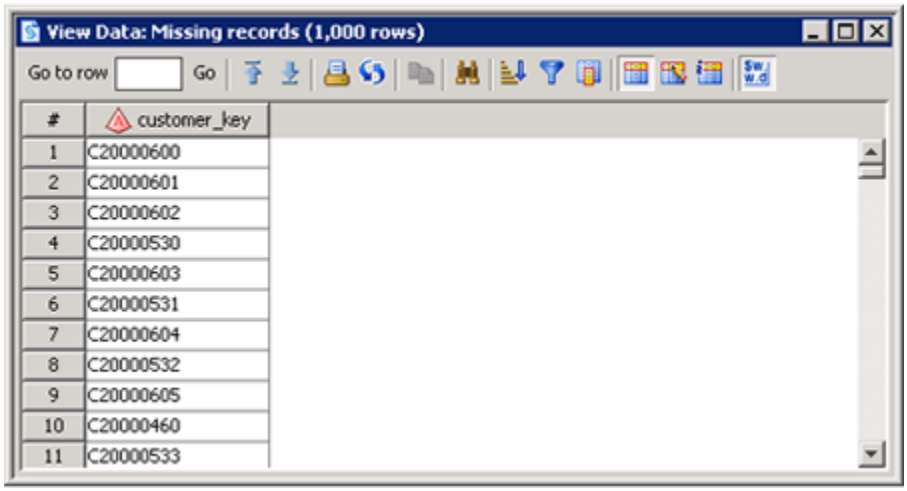
Display 24.14 *Missing Records Output*



#	customer_number	account_number	customer_tax_id		customer_d
1	C10000002	A10000002	3660042735	...	04-JUL-1992
2	C10000004	A10000004	3660042737	...	09-JUN-1994
3	C10000006	A10000006	3660042739	...	20-JAN-1930
4	C10000007	A10000007	3660042740	...	11-NOV-1970
5	C10000008	A10000008	3660042741	...	12-NOV-1935
6	C10000009	A10000009	3660042742	...	16-JUL-1979
7	C10000010	A10000010	3660042743	...	21-APR-1951
8	C10000011	A10000011	3660042744	...	01-NOV-1980
9	C10000013	A10000013	3660042746	...	05-NOV-1956
10	C10000014	A10000014	3660042747	...	28-MAR-1944

7. Review the output displayed in the View Data window for the missing records table, as shown in the following display:

Display 24.15 *Missing Records Output*



#	customer_key
1	C20000600
2	C20000601
3	C20000602
4	C20000530
5	C20000603
6	C20000531
7	C20000604
8	C20000532
9	C20000605
10	C20000460
11	C20000533

Note the following results:

- 1000 changed rows
- 1000 new rows
- 9000 changed rows
- 1000 missing rows

Use a Sequential Data Merge

Perform the following steps to configure the Compare Tables transformation to use a sequential data merge:

1. Open the properties window for the Compare Tables transformation.
2. Click **Options**. Set the **Use direct lookup (hash table)** option in the General category to **No**. This setting saves memory at a potential cost to processing performance.
3. Click **OK** to save your settings and close the properties window.

   Note: The DATA step merge lookup method requires the source table to be presorted by business key with duplicate business keys removed. If the source table is not presorted by business key or the source table contains duplicate business keys, you should add a Sort transformation before you sort and remove duplicate business keys with the Compare Tables transformation. In the Sort transform, navigate to the **Options** tab and change the first SAS Sort option entitled **Remove duplicate records** to **Remove rows with duplicate keys (NODUPKEY)**. You should remove duplicate business keys from source tables before you use the Compare Tables transformation to avoid unexpected results.
4. Insert a Sort transformation between each of the source tables and the Compare Tables transformation.
5. Open each Sort transformation and click **Sort Columns**.
6. Move the columns that you want to use for the match keys in the comparison tables to the **Sort by columns** field.
7. Click **OK** to save your settings and close the properties window.

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 545
- Map Source Columns Into the Target on page 546
- Map Key Columns Between the Source and Lookup Tables on page 547
- Map Lookup Columns Into the Target on page 548
- Create Error and Exception Tables on page 548
- Configure Exception Handling on page 549
- Run the Job and View the Output on page 549

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.

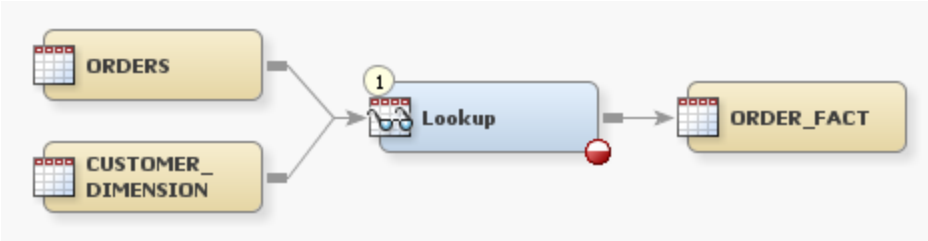
   Note: To add more lookup tables, right-click the Lookup transformation and click **Add Input**.

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 150.)

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 24.16 *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.

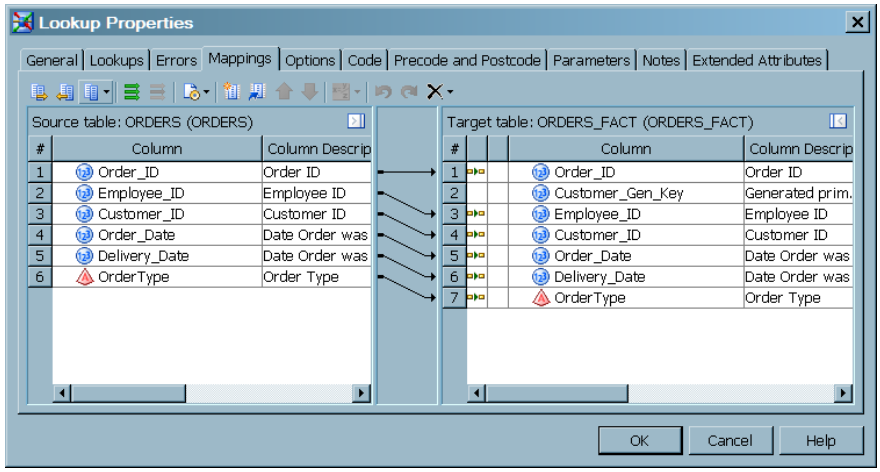
Map Source Columns Into the Target

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.

2. 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 24.17 *Mappings Between Source and Target*



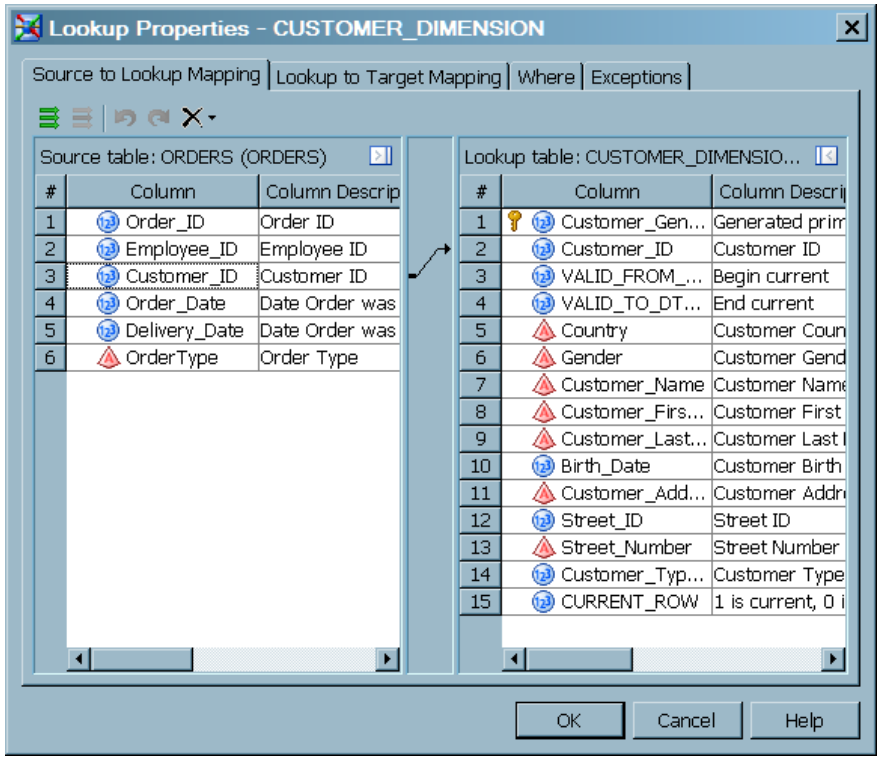
Map Key Columns between the Source and Lookup Tables

Perform the following steps to define the conditions under which values from a lookup table are loaded into the target.

1. Select the **Lookups** tab.
2. Select the lookup table and click **Lookup Properties**.
3. Use the **Source to Lookup Mapping** tab to specify the source and lookup columns that are compared at run time. 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 24.18 *Source to Lookup Column Mapping*



4. 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 525.

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).

1. Open the **Lookup to Target Mapping** tab, and select the **Customer_Gen_Key** column.
2. 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:

1. Open the properties window of the Lookup transformation and select the **Errors** tab.
2. Click **Create error table** and then click **Choose columns**.
3. 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.
4. 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.

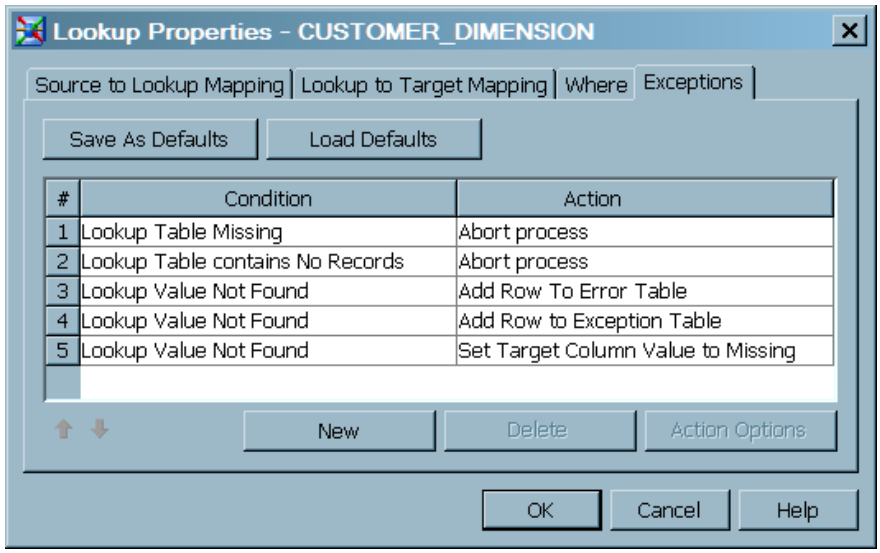
Configure Exception Handling

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 24.19 *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.

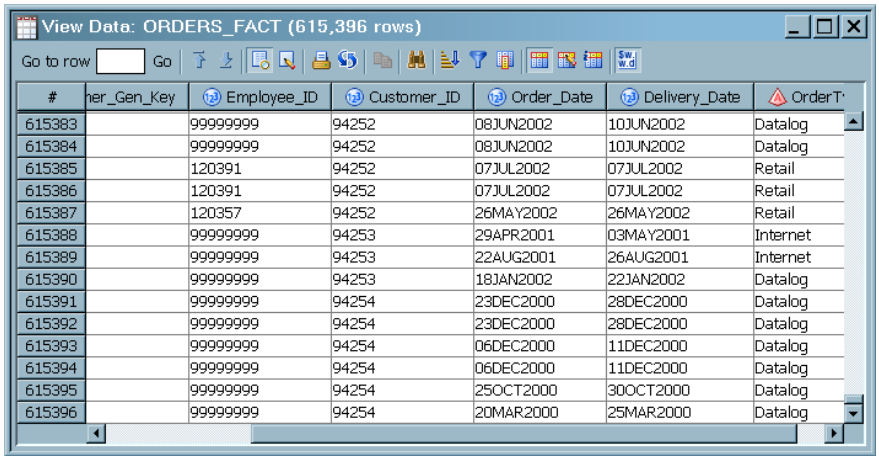
4. Click **OK** to store your entries and close the properties window of the Lookup transformation.

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.

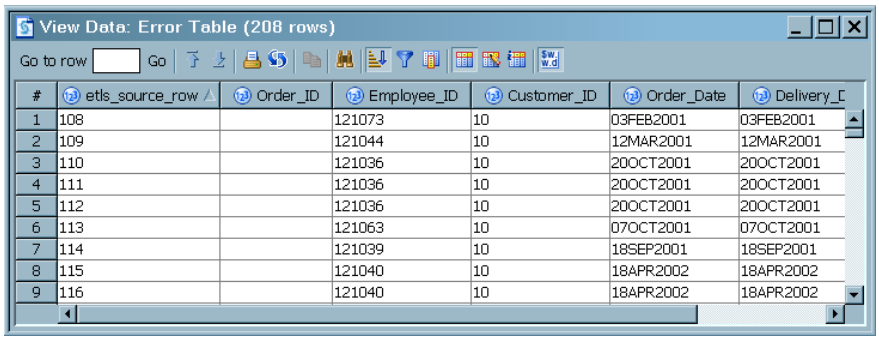
Display 24.20 *Target Table Data*



#	her_Gen_Key	Employee_ID	Customer_ID	Order_Date	Delivery_Date	OrderT
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

4. 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.

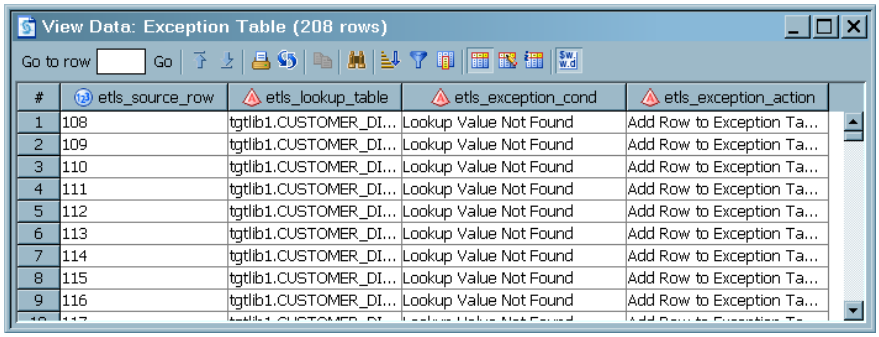
Display 24.21 *Error Table Data*



#	etls_source_row	Order_ID	Employee_ID	Customer_ID	Order_Date	Delivery_D
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

5. 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 24.22 *Exception Table Data*



#	etls_source_row	etls_lookup_table	etls_exception_cond	etls_exception_action
1	108	tgtlib1.CUSTOMER_DI...	Lookup Value Not Found	Add Row to Exception Ta...
2	109	tgtlib1.CUSTOMER_DI...	Lookup Value Not Found	Add Row to Exception Ta...
3	110	tgtlib1.CUSTOMER_DI...	Lookup Value Not Found	Add Row to Exception Ta...
4	111	tgtlib1.CUSTOMER_DI...	Lookup Value Not Found	Add Row to Exception Ta...
5	112	tgtlib1.CUSTOMER_DI...	Lookup Value Not Found	Add Row to Exception Ta...
6	113	tgtlib1.CUSTOMER_DI...	Lookup Value Not Found	Add Row to Exception Ta...
7	114	tgtlib1.CUSTOMER_DI...	Lookup Value Not Found	Add Row to Exception Ta...
8	115	tgtlib1.CUSTOMER_DI...	Lookup Value Not Found	Add Row to Exception Ta...
9	116	tgtlib1.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 551
- "Add the Surrogate Key Column to the Target" on page 551
- "Identify Tables and Columns in the Transformation" on page 552
- "Run the Job and View the Output" on page 553

Tasks

Create and Populate the Job

Perform the following steps to create and populate the job:

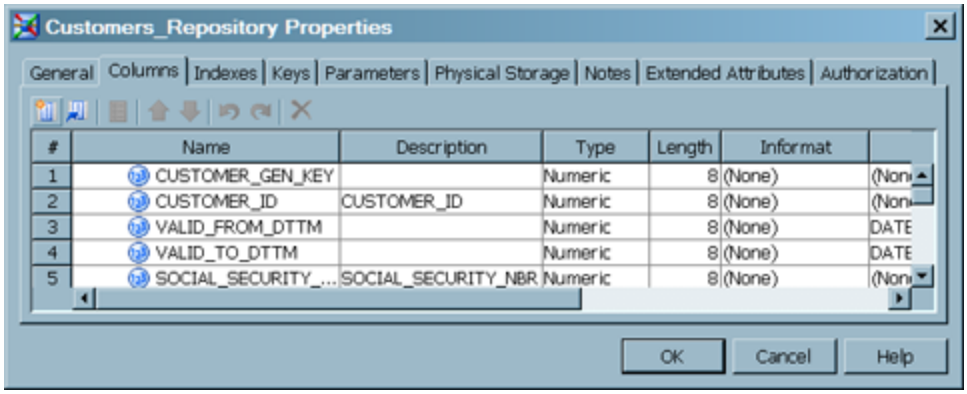
1. Create an empty SAS Data Integration Studio job.
2. In the Transformations tree, in the **Data** folder, drag the Surrogate Key Generator transformation into the empty job on the **Diagram** tab.
3. Select and drag the source table from its folder and drop it before the Surrogate Key Generator transformation 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 **Register Table**. Enter an appropriate name in the **Name** field on the **General** tab. Then go to the **Physical Storage** tab and enter a table name that does not exist in the **Physical Name** field. Finally, select a permanent library for the target table creation and select **OK**.

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. Enter 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 24.23 *Completed Columns Tab for Sample Job*



6. Click **OK** to save your changes and close the properties window.

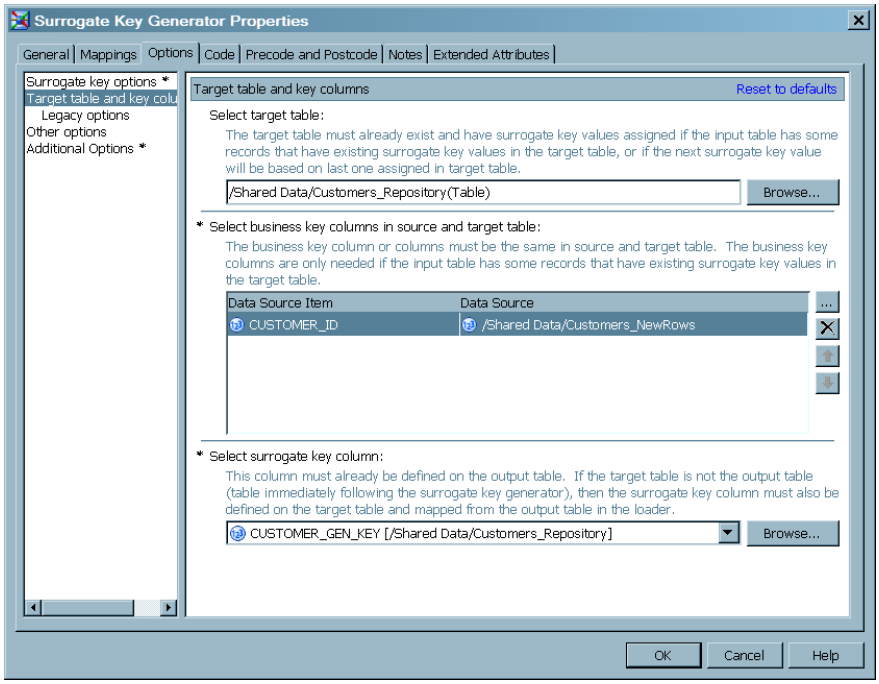
Identify Tables and Columns in the Transformation

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.
4. 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 24.24 *Completed Options Tab for Sample Job*



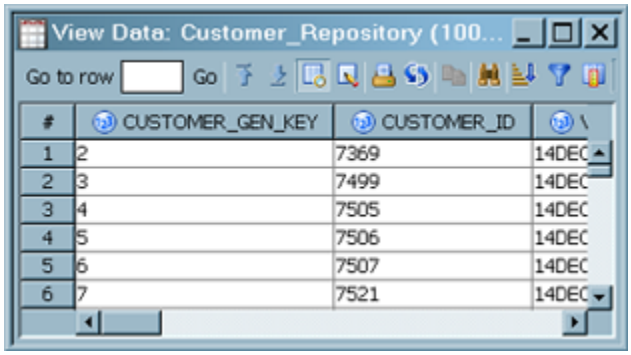
5. Click **OK** to save the option specifications and close the properties window.

Run the Job and View the Output

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

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 generated surrogate key values. The following display depicts the target table data for the sample job.

Display 24.25 *Generated Key Values in the Sample Target Table*



#	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 554
- "Identify Source Columns" on page 554
- "Run the Job and View the Output" on page 555

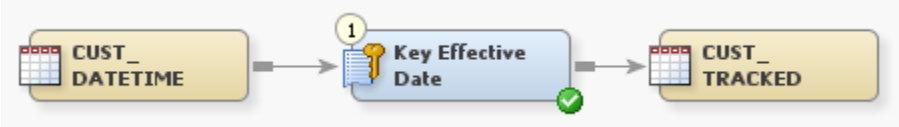
Tasks

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 24.26 *Sample Key Effective Date Process Flow Diagram*



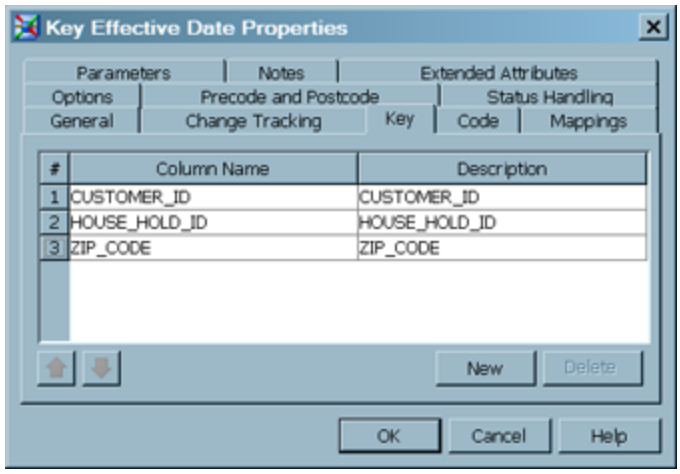
Identify Source Columns

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 24.27 *Order of Primary Key Columns on the Key Tab*



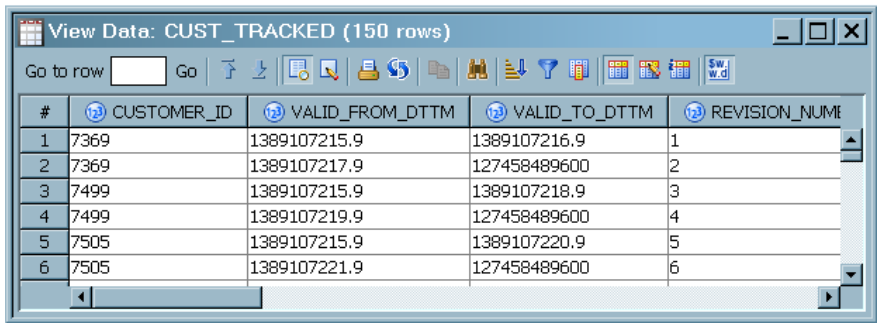
5. Click **OK** to close the properties window.

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 24.28 *Tracked Datetime Values in the Sample Target Table*



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 25

Working with Change Data Capture

About the Change Data Capture Transformations **557**
Change Data Capture Defined 557
Prerequisites for Change Data Capture 558
About CDC Changed Data Tables **559**
About CDC Control Tables **559**
Capture Changed Data from Oracle **560**
Problem 560
Solution 560
Tasks 561

About the Change Data Capture Transformations

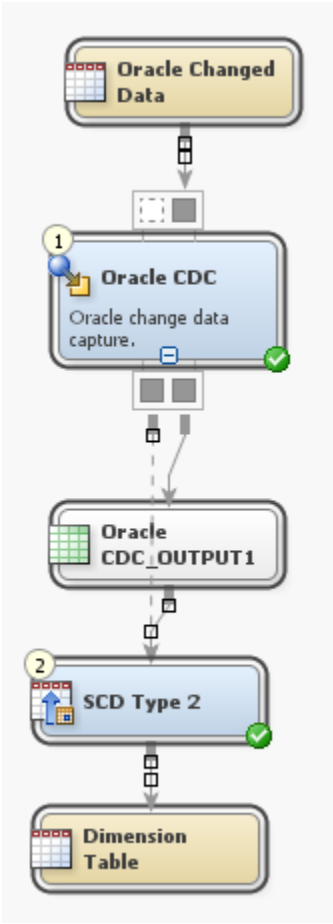
Change Data Capture Defined

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 522.

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 25.1 *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 559.

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. Note that Oracle CDC supports Asynchronous Autolog Publishing.

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 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 columns can vary. To identify the columns to the CDC transformations, you specify option values in properties window.

The CDC transformations generate target data that is suitable for loading into star schemas using the SCD Type 2 Loader transformation.

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 535.

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 561

- "Create and Populate the Job" on page 561
- "Configure Row Processing" on page 562
- "Configure the Use of the Control Table" on page 563
- "Run the Job, Update the Metadata, and View the Output" on page 564

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 558.
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.

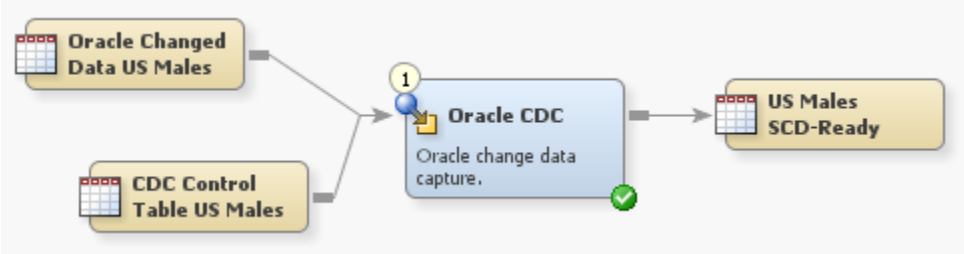
Create and Populate the Job

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 150.)

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 25.1 *Sample Oracle CDC Process Flow Diagram*

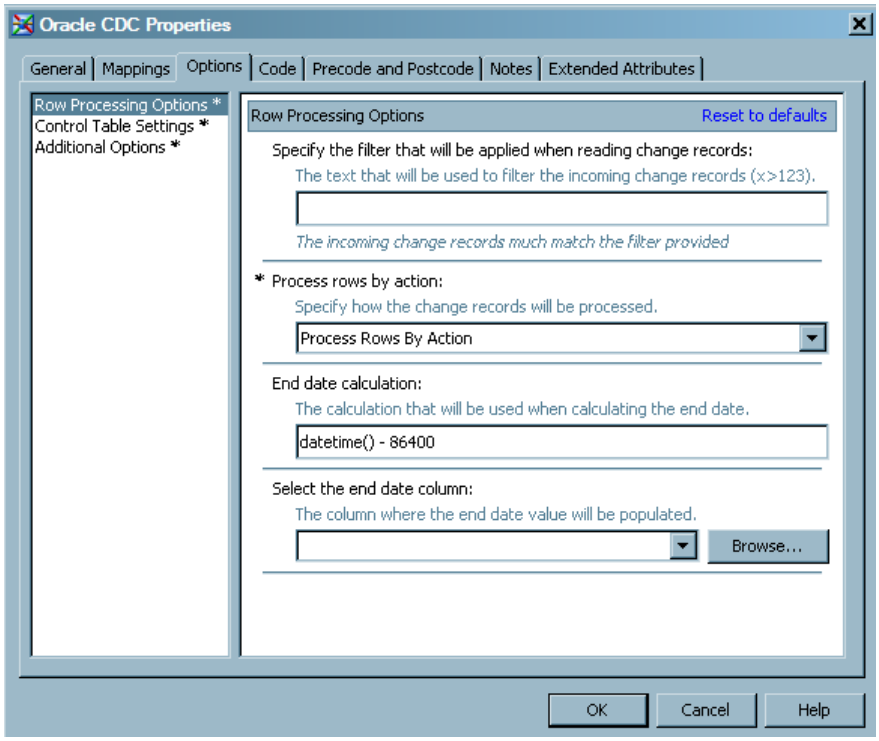


Configure Row Processing

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 25.2 *Row Processing Options*

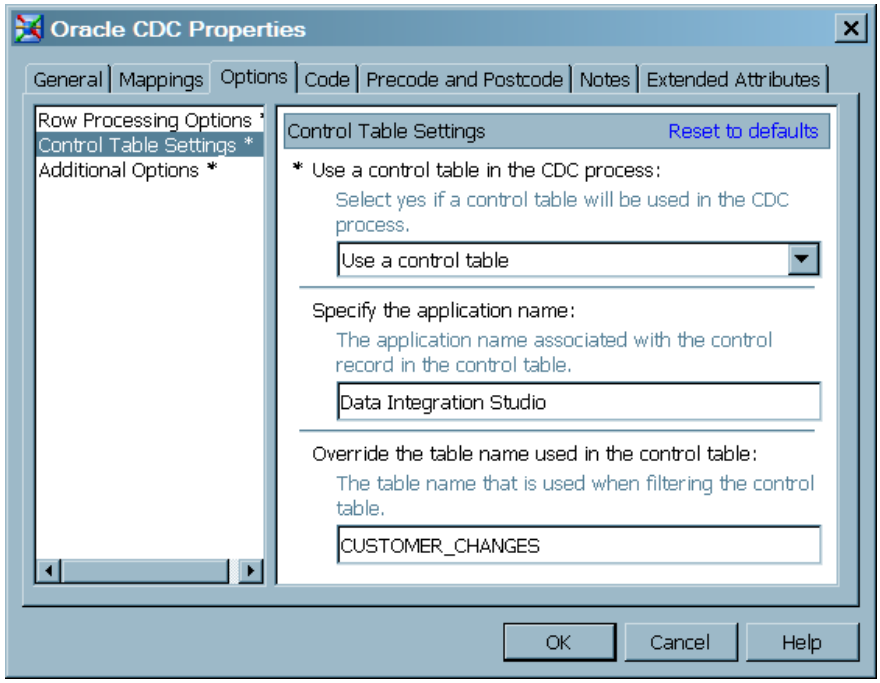


Configure the Use of the Control Table

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 25.3 *Completed Control Table Options*



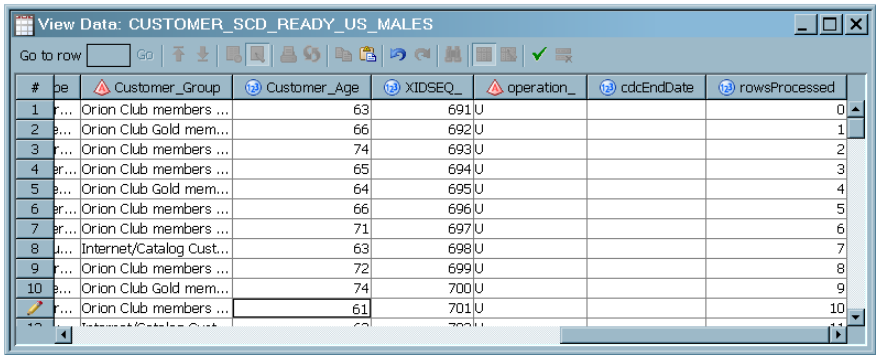
5. Click **OK** to save the option specifications and close the properties window.

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 25.4 *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
	r...	Orion Club members ...	61	701	U		10

Chapter 26

Working with Message Queues

About Message Queues **567**

Prerequisites for Message Queues **568**

Selecting Message Queue Transformations **569**
Problem 569
Solution 569
Tasks 569

Processing a WebSphere Queue **570**
Problem 570
Solution 570
Tasks 570

Polling a Websphere Message Queue **572**
Problem 572
Solution 572
Tasks 573

Processing a Microsoft Queue **574**
Problem 574
Solution 574
Tasks 574

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 26.1 *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 572.

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 26.2 *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 570
- "Configure and Run the Websphere Queue Writer Job" on page 571
- "Verify the Websphere Queue Writer Job" on page 571
- "Create the Websphere Queue Reader Job" on page 571
- "Configure and Run the Websphere Queue Reader Job" on page 571
- "Verify the Websphere Queue Reader Job" on page 572

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 26.1 *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:

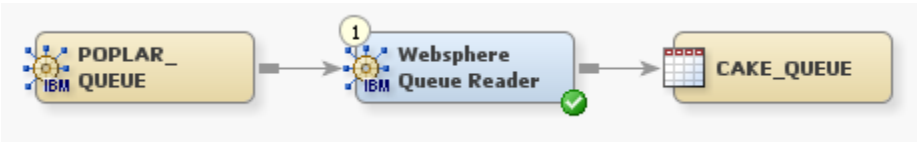
1. Open the IBM WebSphere Queue Explorer application.
2. Select the queue that you created and ran. Then, verify that the expected messages are sitting on the queue.

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 150.)
6. After these steps have been completed, the process flow diagram for this example resembles the following display.

Display 26.2 *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.

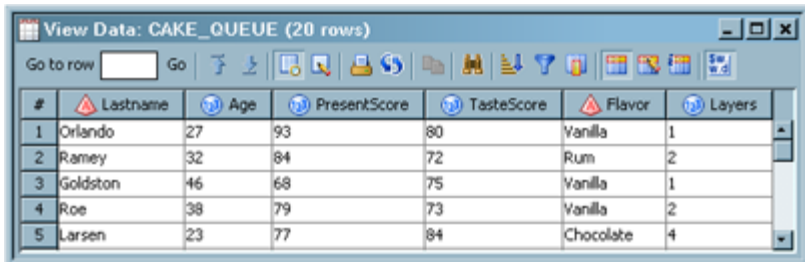
3. Run the job. If you are prompted to do so, enter a user ID and password for the default SAS Application Server that generates and runs SAS code for the job. The server executes the SAS code for the job.
4. If the job completes without error, go to the next section. If error messages appear, read and 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 26.3 *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 570.
3. Deploy the queue reader job for scheduling. See "Deploy the SAS Job for Scheduling" on page 573.
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 574
- "Configure and Run the Microsoft Queue Job" on page 575
- "Verify the Microsoft Queue Job" on page 575

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 150.)

   The job resembles the sample shown in the following display.

Display 26.4 *Sample Microsoft Message Queue Process Flow*



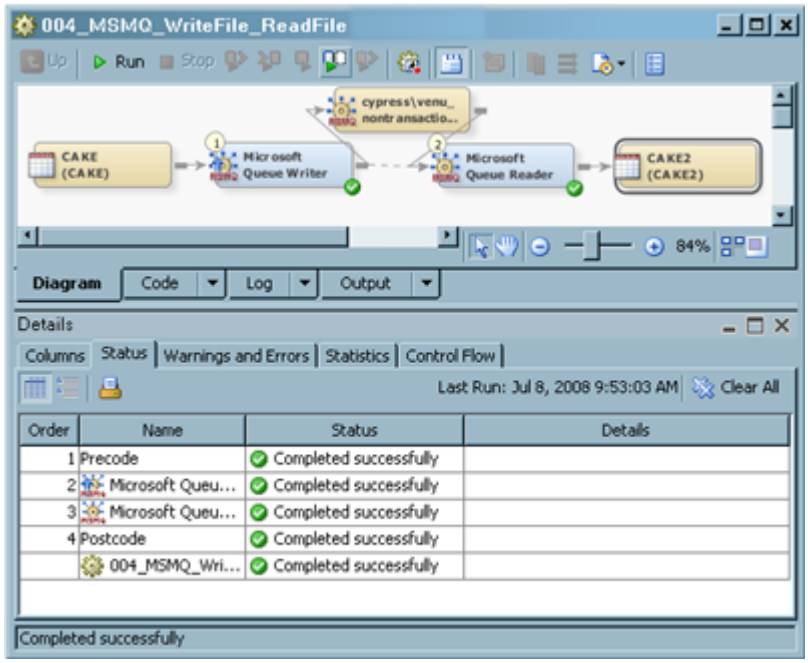
The source table for the sample job is named CAKE. The target table is named CAKE2, and the queue is named cypress\venu\nontransactional.

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 26.5 *Sample Completed Microsoft Message Queue Job*



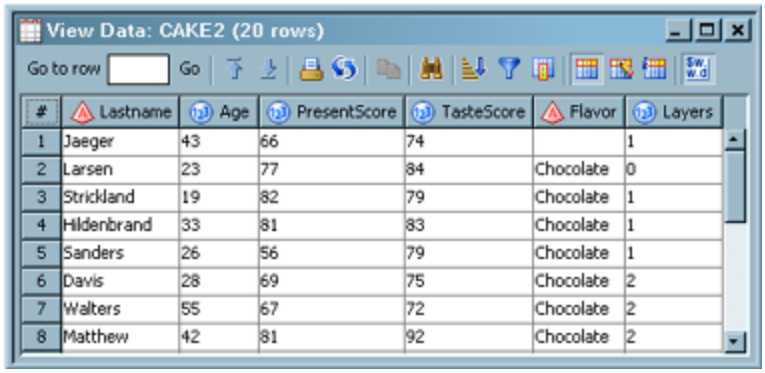
Verify the Microsoft Queue Job

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 26.6 *Target Table Data for the Sample Job*



#	Lastname	Age	PresentScore	TasteScore	Flavor	Layers
1	Jaeger	43	66	74		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 27

Working with SPD Server Cluster Tables

About SPD Server Cluster Tables **577**

Creating an SPD Server Cluster Table **578**
Problem 578
Solution 578
Tasks 578

Maintaining an SPD Server Cluster **579**
Problem 579
Solution 580

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 82.

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 578
- "Specify Options for the Create or Add to a Cluster Transformation" on page 579

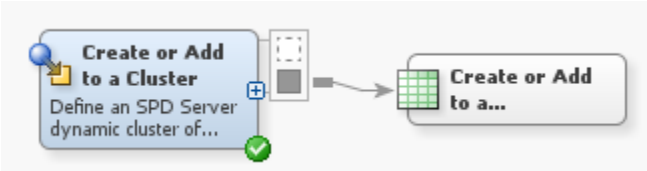
Tasks

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.

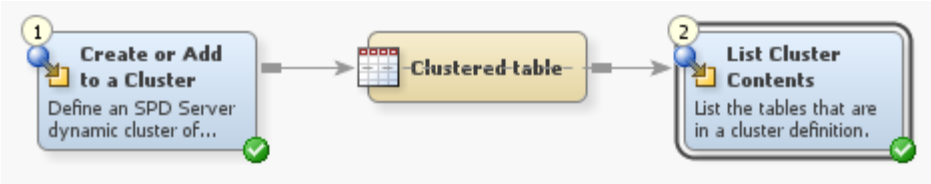
   Display 27.1 *Sample SPD Server Cluster Table Job with Temporary Output Table*

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 150.
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 27.2 *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 27.3 *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 27.1 *SPD Server Transformations*

Server	Tasks That Require This Server
Add a table to a cluster	Perform the following steps to use the Create or Add to a Cluster transformation: 1. Create an empty job. 2. Drop the Create or Add to a Cluster transformation into the Job Editor window. 3. Replace the temporary output table with the clustered table. 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. 5. Run the job.
Generate a list of tables in a cluster	Perform the following steps to use the List Cluster Contents transformation: 1. Create an empty job. 2. Drop the List Cluster Contents transformation into the Job Editor window. 3. Drop the cluster table onto the Job Editor window. 4. Drag the cursor from the output port of the cluster table to the input port of the List Cluster Contents transformation. 5. Run the job. 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.

Server	Tasks That Require This Server
Remove a cluster definition	Perform the following steps to use the Remove Cluster transformation: 1. Create an empty job. 2. Drop the Remove Cluster transformation into the Job Editor window. 3. Drop the cluster table into the Job Editor window. 4. Drag the cursor from the output port of the cluster table to the input port of the Remove Cluster transformation. 5. Run the job. The cluster table is now removed and the tables that were in the cluster are now available as individual tables.

Chapter 28

Working with Hadoop and SAS LASR Analytic Server

Overview of the Hadoop Transformations **584**

Prerequisites for the Hadoop Transformations **585**
Prerequisites for the Hive Transformation 585
Prerequisites for Other Hadoop Transformations 585

Creating a Pig Job **586**
Problem 586
Solution 586
Tasks 586

Creating a Hive Job **591**
Problem 591
Solution 591
Tasks 592

Creating a Hadoop Container Job **594**
Problem 594
Solution 594
Tasks 595

Monitoring Hadoop Clusters **603**

Overview of the High-Performance Analytics Transformations **603**
High-Performance Analytics Transformations 603
Software Used by These Transformations 604

Prerequisites for the High-Performance Analytics Transformations **605**
For SAS Data Integration Studio Users 605
For Administrators 606

Loading a Table on the SAS LASR Analytic Server **606**
Problem 606
Solution 607
Tasks 607

Usage Notes for HPA Software and Hadoop **608**
High-Performance Analytics Software 608
Hadoop 609

Overview of the Hadoop Transformations

The Transformations tree in SAS Data Integration Studio includes a **Hadoop** folder. The transformations in this folder enable you to perform the following operations within a job:

- read files from a Hadoop cluster and write files to the cluster
- transfer files to and from a Hadoop cluster
- submit your own Pig Latin, HiveQL, or Map Reduce code
- use one transformation to perform a series of steps in one connection to the Hadoop cluster, such as transfers to and from Hadoop, Map Reduce processing, and Pig Latin processing

Hadoop is an open-source technology for large data volume storage and processing. Hadoop provides scalability through the union of the Hadoop Distributed File System (HDFS), its high bandwidth and clustered storage system, and Map Reduce, its fault-tolerant, distributed processing algorithm.

Apache Pig is a high-level platform for creating Map Reduce programs that are used with Hadoop. The language for this platform is called Pig Latin. Apache Hive is a data warehouse infrastructure built on top of Hadoop for data queries, analysis, and summarization. It provides an SQL-like language called HiveQL.

The following table describes the **Hadoop** transformations.

Name	Description
Hadoop Container	Enables you to use one transformation to perform a series of steps in one connection to the Hadoop cluster. The steps can include transfers to and from Hadoop, Map Reduce processing, and Pig Latin processing. For more information, see “Creating a Hadoop Container Job” on page 594.
Hadoop File Reader	Reads a specified file from a Hadoop cluster.
Hadoop File Writer	Writes a specified file to a Hadoop cluster.
Hive	Enables you to submit your own HiveQL code in the context of a job. For more information, see “Creating a Hive Job” on page 591.
Map Reduce	Enables you to submit your own Map Reduce code in the context of a job. You must create your own Map Reduce program in Java and save it to a JAR file. You then specify this JAR file in the Map Reduce transformation, along with some relevant arguments. Your Hadoop installation usually includes an example Map Reduce program. For an example of Map Reduce processing in a Hadoop container job, see “Creating a Hadoop Container Job” on page 594.
Pig	Enables you to submit your own Pig Latin code in the context of a job. For more information, see “Creating a Pig Job” on page 586.

Name	Description
Transfer From Hadoop	Transfer a specified file from a Hadoop cluster. For an example of how this transformation can be used, see "Creating a Hadoop Container Job" on page 594.
Transfer To Hadoop	Transfer a specified file to a Hadoop cluster.

Prerequisites for the Hadoop Transformations

Prerequisites for the Hive Transformation

The Hive transformation uses the SAS/ACCESS Interface for Hadoop. Accordingly, the Hive transformation has the following prerequisites:

- Your site must meet the system requirements for SAS/ACCESS Interface for Hadoop, for your SAS release and operating system. You can review system requirements for SAS products at: **http://support.sas.com/documentation/installcenter/index.html**.
- You must establish connectivity to Hadoop. This includes registering the Hadoop Server and a "Hadoop via Hive" library on the SAS Metadata Server. See "Establishing Connectivity to Hadoop" in the *SAS Intelligence Platform: Data Administration Guide*. This book is available at the SAS Intelligence Platform documentation page: **http://support.sas.com/documentation/onlinedoc/intellplatform/**.
- You must copy certain JAR files from the Hadoop installation folder to a folder on the SAS Workspace Server that executes SAS Data Integration Studio jobs. For a list of the JAR files required to support the SAS/ACCESS Interface for Hadoop, see *SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS*. This book is available at the Third-Party Software page: **http://support.sas.com/resources/thirdpartysupport/**.

Prerequisites for Other Hadoop Transformations

Unlike the Hive transformation, the following Hadoop transformations use the HADOOP procedure:

- Hadoop Container
- Hadoop File Reader
- Hadoop File Writer
- Map Reduce
- Pig
- Transfer From Hadoop
- Transfer to Hadoop

Accordingly, these transformations have the following prerequisites:

- You must establish connectivity to Hadoop. This includes registering the Hadoop Server on the SAS Metadata Server. See "Establishing Connectivity to Hadoop" in the *SAS Intelligence Platform: Data Administration Guide*. This book is available at the SAS Intelligence Platform documentation page: **http://support.sas.com/documentation/onlinedoc/intellplatform/**.
- You must copy certain JAR files from the Hadoop installation folder to a folder on the SAS Workspace Server that executes SAS Data Integration Studio jobs. For a list of the JAR files required to support Base SAS (including the HADOOP procedure), see *SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS*. This book is available at the Third-Party Software page: **http://support.sas.com/resources/thirdpartysupport/**.

See also "Usage Notes for HPA Software and Hadoop" on page 608.

Creating a Pig Job

Problem

You want to submit user-written Pig Latin code in the flow for a SAS Data Integration Studio job.

Solution

You can create a SAS Data Integration Studio job that contains the Pig transformation. This transformation enables you to submit your own Pig Latin code in the context of a job. Pig Latin is a high-level language used for expressing and evaluating data analysis programs. It is assumed that you know Pig Latin well enough to use it in a production environment.

Perform the following tasks.

- "Create a Pig Job" on page 586
- "Configure the Job" on page 587
- "Run the Job and Review the Output" on page 590

Tasks

Create a Pig Job

The Pig job for this example contains a text table used as a source table, a Transfer to Hadoop transformation, a target table and a Pig transformation. This job structure enables you to transform the text table into a Hadoop file that can be processed in the Pig transformation.

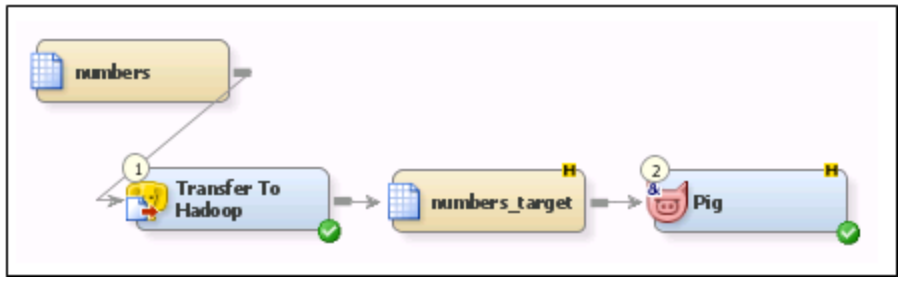
Perform the following steps to create and populate the job:

1. Create an empty SAS Data Integration Studio job.
2. Locate the source table in the Inventory tree. Then, drop it in the empty job on the **Diagram** tab in the Job Editor window. The source table for this sample job is a text table named numbers.txt.

3. Select a Transfer to Hadoop transformation from the Hadoop folder in the Transformations tree. Then, drop it onto the **Diagram** tab.
4. Connect the source table to the Transfer to Hadoop transformation.
5. Locate the target table and drop it onto the **Diagram** tab. The target table for this sample job is a Hadoop table named numbers_target.

   Note that the **All files in this location** check box must be selected on the **File Location** tab in the properties window for the Pig target table. This step enables you to see the data in the table after the job has completed successfully.
6. Connect the Transfer to Hadoop transformation to the target table.
7. Select a Pig transformation from the Hadoop folder in the Transformations tree. Then, drop it onto the **Diagram** tab.
8. Connect the target table to the Pig transformation. The **Diagram** tab for the job is shown in the following display:

Display 28.1 *Pig Job Flow*



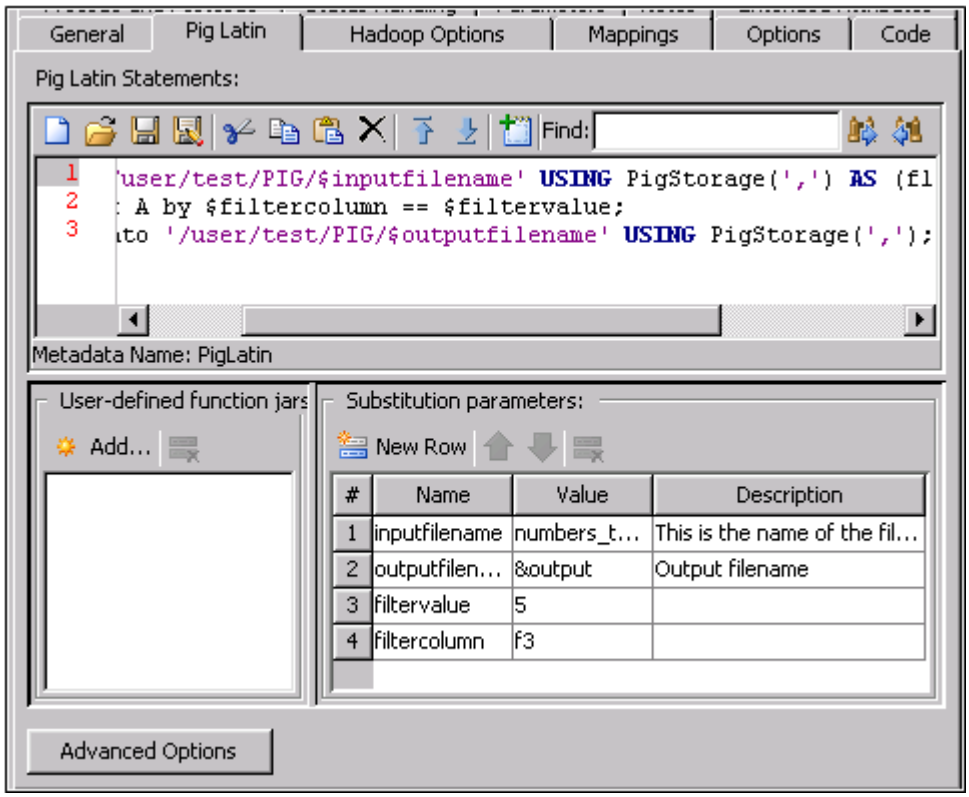
Configure the Job

The configuration for the Pig transformation is simple. Open the **Hadoop Options** tab and select the **Delete outputs before executing hadoop statements** check box.

The configuration needed for the Pig transformation varies from job to job. This sample job requires that you add three Pig Latin statements and four substitution parameters on the **Pig Latin** tab.

The tab is shown in the following display:

Display 28.2 *Pig Latin Tab*



Note that the Pig Latin statements are entered in the **Pig Latin** field. The sample job contains the following statements:

```
A = load '/user/test/PIG/$inputfilename' USING PigStorage(',')
AS (f1:int,f2:int,f3:int);
B1 = filter A by $filtercolumn == $filtervalue;
store B1 into '/user/test/PIG/$outputfilename' USING PigStorage(',');
```

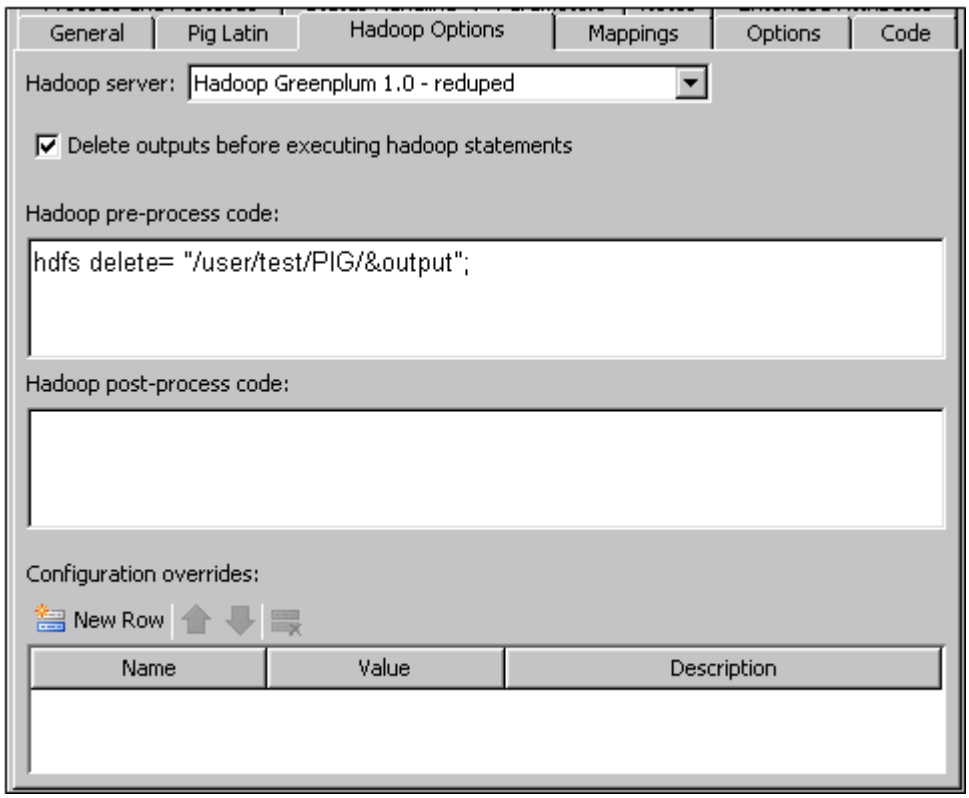
Similarly, the substitution parameters are entered in the **Substitution parameters** field, as follows:

```
Name = inputfilename, Value = numbers_target.txt,
Description = This is the name of the file loaded into hadoop
Name = outputfilename, Value = &output, Description = Output filename
Name = filtervalue, Value = 5
Name = filtercolumn, Value = f3
```

You also need to open the **Hadoop Options** tab to select the **Delete outputs before executing hadoop statements** check box.

Then, enter appropriate code into the **Hadoop pre-process code** field, as shown in the following display:

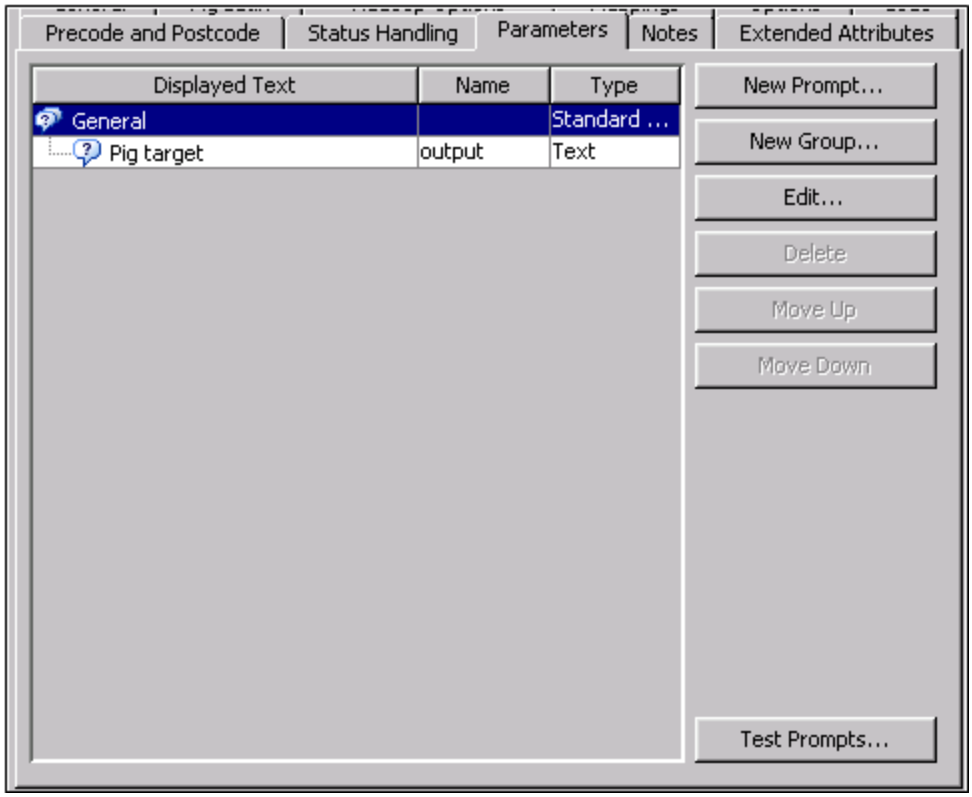
Display 28.3 *Hadoop Options in the Pig Transformation*



Finally, you need to create a new prompt for the substitution parameter on the **Parameters** tab. The general values for this job are **`Name=output`** and **`Displayed text=Pig target`**. The prompt type and values are **`Prompt type=Text`** and **`Default value=PIG_SubstitutionParamtarget.txt`**.

The following display shows the completed **Parameters** tab:

Display 28.4 *Parameters Tab*

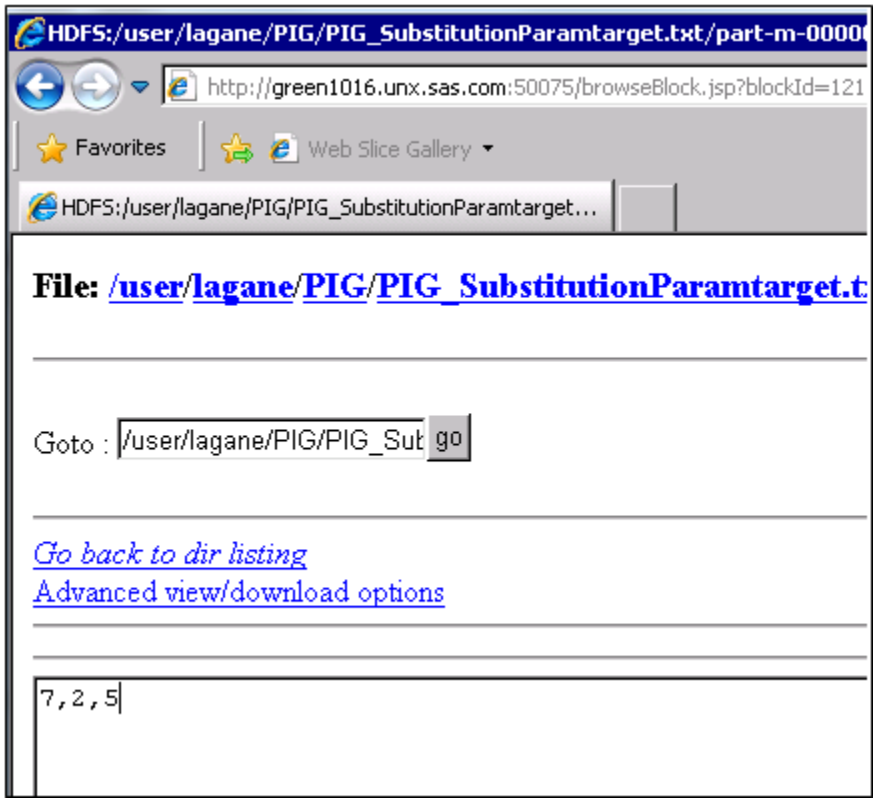


Run the Job and Review the Output

Run the job and verify that the job completes without error. Then, you can use the Hadoop monitoring tools to review the output. Right-click the Pig transformation on the **Diagram** tab and select **Hadoop Monitoring** ⇨ **Distributed file system viewer**. Then, click **Browse the filesystem** and drill down through the **User** folder and the **test** folder to the **PIG** folder. Then click the name of the substitution variable for the job, in this case **PIG_SubstitutionParamtarget.txt**. Finally, click the data file, **part-m-00000**. Verify that the expected output, which is `7,2,5`, is displayed in the browser window.

The output for the sample job is shown in the following display:

Display 28.5 *Pig Job Output*



Creating a Hive Job

Problem

You want to submit user-written HiveQL code in the flow for a SAS Data Integration Studio job.

Solution

You can create a SAS Data Integration Studio job that contains the Hive transformation. This transformation enables you to submit your own HiveQL code in the context of a job. Apache Hive is a data warehouse infrastructure built on top of Hadoop for data queries, analysis, and summarization. It provides an SQL-like language called HiveQL. It is assumed that you know HiveQL well enough to use it in a production environment.

Perform the following tasks.

- "Create a Hive Job" on page 592
- "Configure the Job" on page 592
- "Run the Job and Review the Output" on page 593

Tasks

Create a Hive Job

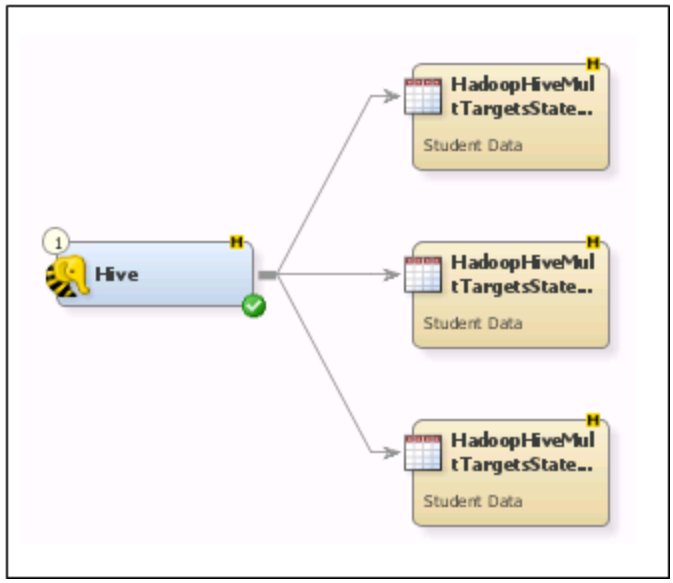
The Hive job for this example contains the Hive transformation and three target tables. This job structure enables the submission of multiple Hadoop Hive statements to multiple Hive target tables.

Perform the following steps to create and populate the job:

1. Create an empty SAS Data Integration Studio job.
2. Select a Hive transformation from the Hadoop folder in the Transformations tree. Then, drop it in the empty job on the **Diagram** tab in the Job Editor window.
3. Select one or more target tables and add them to the **Diagram** tab. The target tables for the sample job are named HadoopHiveMultTargetsStatements1, HadoopHiveMultTargetsStatements2, and HadoopHiveMultTargetsStatements3.
4. Connect each target table to an output from the Hive transformation.

   The **Diagram** tab for the job is shown in the following display:

   Display 28.6 *Hadoop Hive Job Flow*



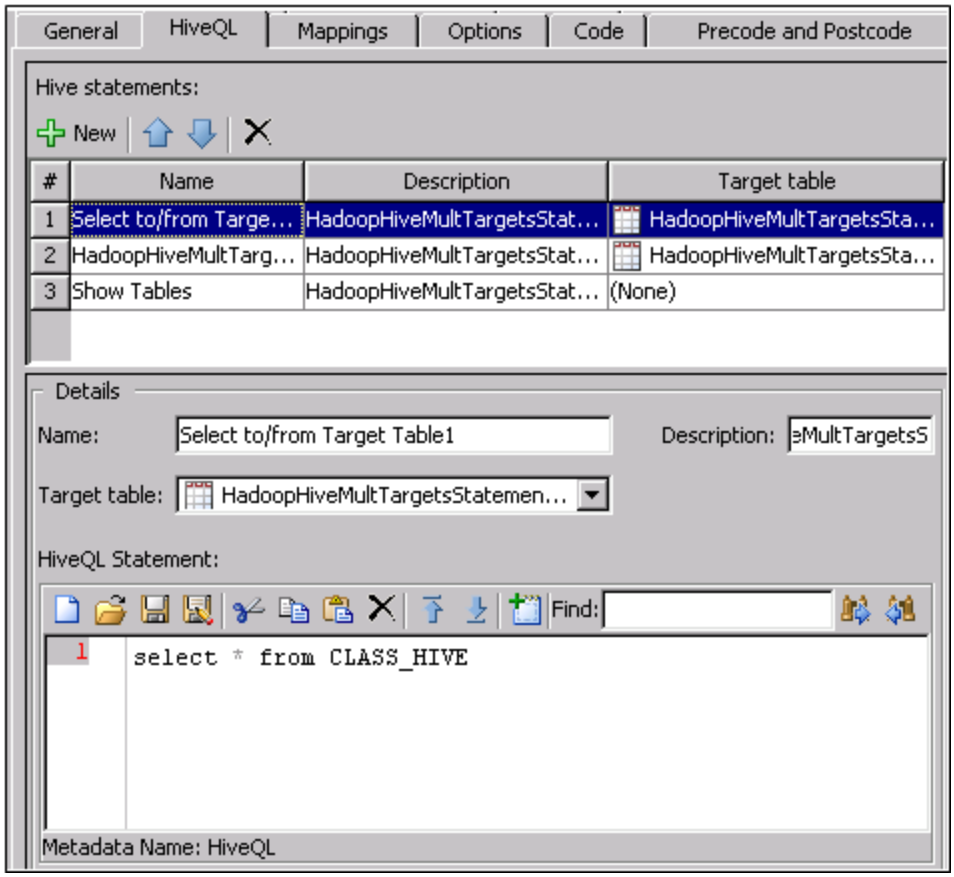
   Note that the source table for this particular job is specified in the HiveQL statements described in the next section. You can also create a job that includes a source table that you add to the **Diagram** tab and connect to the input port of the Hive transformation. The source table for the **Hive** transformation must be a Hadoop table.

Configure the Job

The configuration needed for the Hive transformation varies from job to job. This sample job requires that you add two SELECT statements and one SHOW TABLES statement to the table on the **HiveQL** tab.

The HiveHQ tab is shown in the following display:

Display 28.7 *HiveHQ Tab*



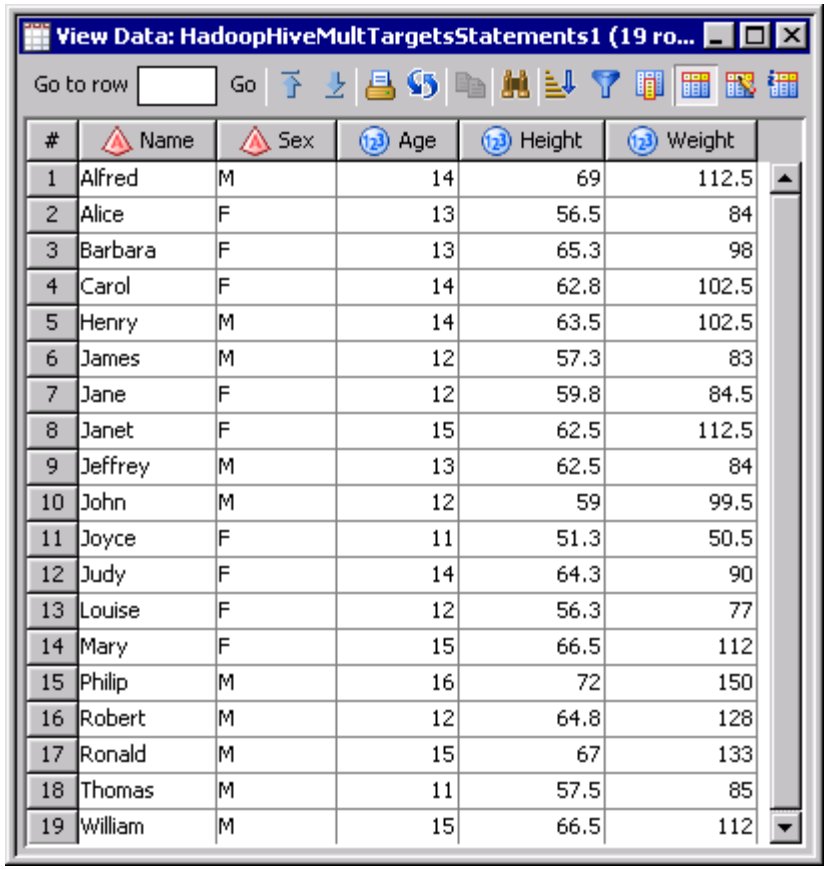
Note that the Hive statements are listed in the **Hive statement** field at the top of the tab. The code for the selected statement is displayed in the **HiveQL Statement** field at the bottom of the tab. For this job, the first two statements select the contents of the CLASS_HIVE table and specify a target table. The third row adds a SHOW TABLES statement with no target table.

Run the Job and Review the Output

Run the job and verify that the job completes without error. Then, review the output. Based on the HiveQL statements specified in the **Hive** transformation, the first two tables should contain the same output and the third table should contain no observations.

The following display shows the output from the first table:

Display 28.8 *Hive Output*



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

Note: This table can take an unexpectedly long time to close. The delay is longer when row counts are enabled. Therefore, consider clearing the **Enable row count on basic properties for tables** check box on the **General** tab of the Options window. (You can access the Options window from the **Tools** menu.)

Creating a Hadoop Container Job

Problem

You want to run multiple Hadoop processes without creating an overly complicated SAS Data Integration Studio job.

Solution

You can create a SAS Data Integration Studio job that contains the Hadoop Container transformation. This transformation enables you to use one transformation to perform a series of steps in one connection to the Hadoop cluster. The steps can include transfers to and from Hadoop, Map Reduce processing, and Pig Latin processing.

For example, you can create a sample job that performs the following tasks that are run through the Hadoop Container transformation:

- Transfer data from a text source file to a Hadoop output file (Transfer To Hadoop step).
- Transfer data from a Hadoop source file to a text output file (Transfer From Hadoop step).
- Process data using a Map Reduce step.
- Process data using a Pig step.

Perform the following tasks:

- "Create a Hadoop Container Job" on page 595
- "Add and Review Hadoop Steps" on page 596
- "Configure Transfer Steps" on page 598
- "Configure a Map Reduce Step" on page 599
- "Configure a Pig Step" on page 601
- "Configure the Hadoop Options Tab" on page 602
- "Run the Job and Review the Output" on page 602

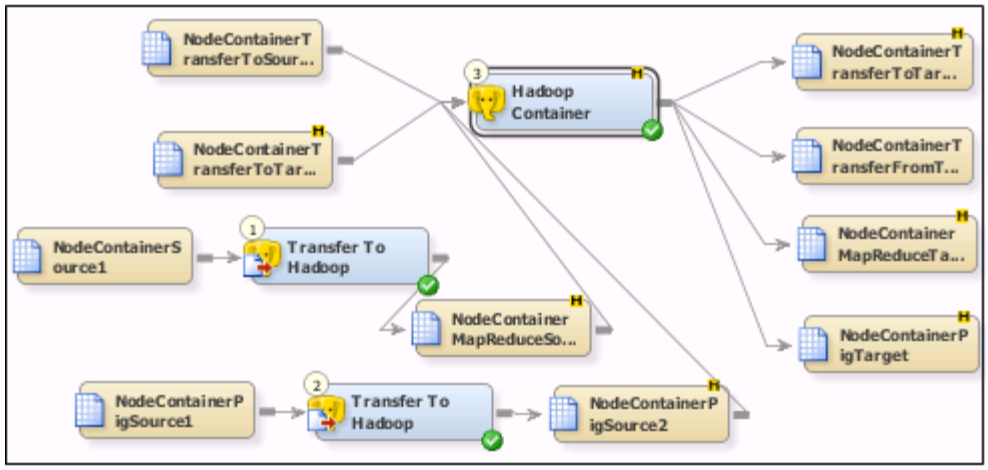
Tasks

Create a Hadoop Container Job

You can create a Hadoop Container job similar to the sample job, which contains four Hadoop steps that correspond to four rows of tables and transformations.

The **Diagram** tab for the job is shown in the following display:

Display 28.9 *Hadoop Container Flow*



The two rows at the top of the tab are used for the Transfer To and Transfer From steps. Note that the first row contains a text source table and Hadoop target table, where the second row contains a Hadoop source table and a text target table. The two rows at the bottom of the tab are used for the Map Reduce and Pig steps. Note that each row begins with a text source and a Transfer To Hadoop transformation that creates the Hadoop

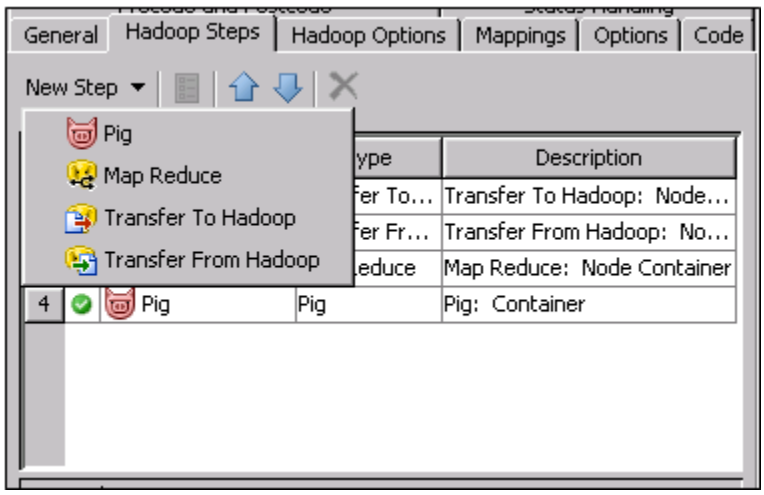
source table for the Hadoop Container transformation. Both rows feed steps that send output to Hadoop target tables.

Add and Review Hadoop Steps

The steps processed in the Hadoop Container transformation are listed in a table on the **Hadoop Steps** tab. You can add, edit, reorder, and delete steps by clicking the buttons in the toolbar at the top of the tab.

The available step types are shown in the following display:

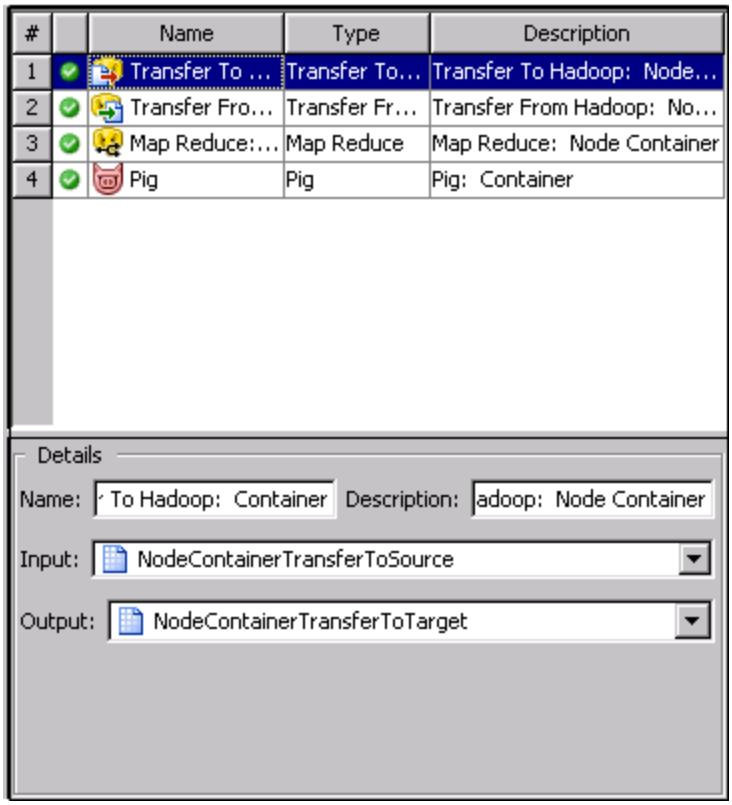
Display 28.10 *Hadoop Step Types*



You can click a row in the table to review its name, description, input, and output in the Details panel at the bottom of the tab. If a step has multiple inputs or outputs, you can use the drop-down arrow to select the object that you need.

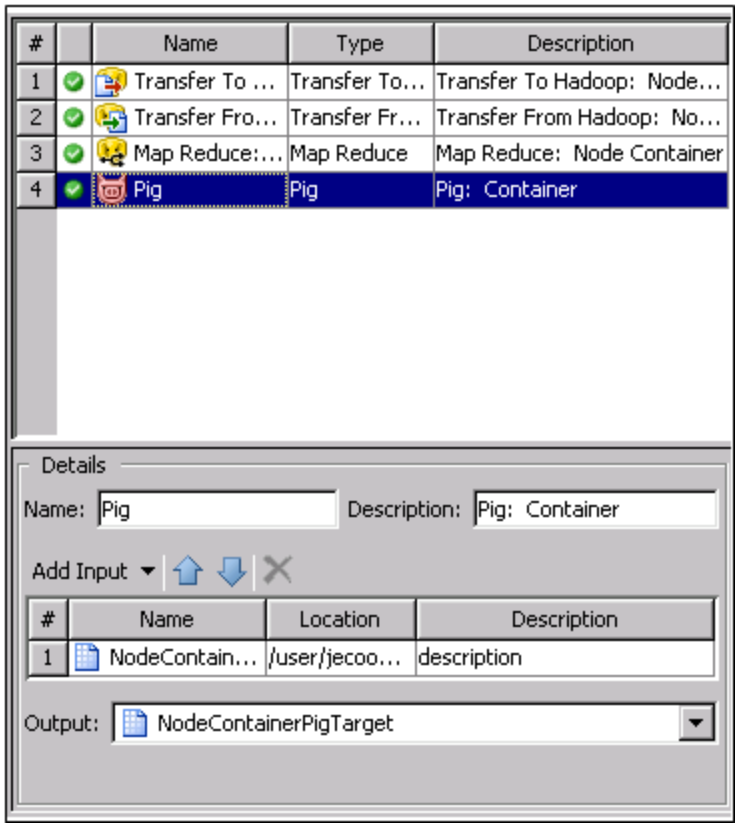
The following display shows the Details panel for the Transfer To step in the sample job:

Display 28.11 *Transfer to Details*



The Details panel for the Pig step is shown in the following display:

Display 28.12 *Pig Details Panel*



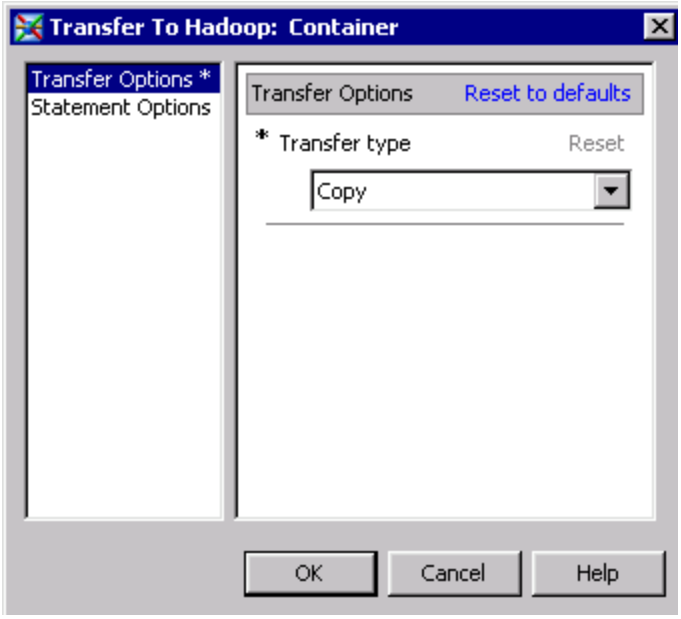
You can select a step and click the **Properties** button to configure, review, and edit its properties.

Configure Transfer Steps

The Transfer To and Transfer From steps are configured in a window that contains panes that cover **Transfer Options** and **Statement Options**.

The following display shows the transfer options set for the Transfer To step in the sample job:

Display 28.13 *Transfer Options Window*

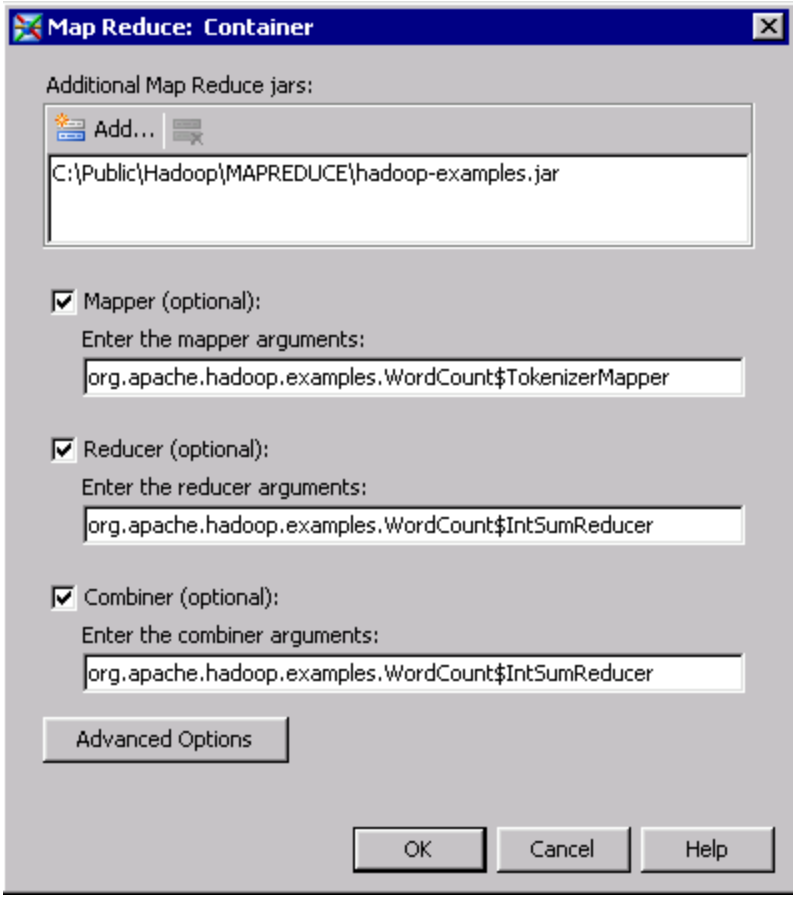


Configure a Map Reduce Step

The Map Reduce step enables you to submit your own Map Reduce code in the context of a job. It is assumed that you know Map Reduce well enough to use it in a production environment. Your Hadoop installation usually includes an example Map Reduce program. You must create your own Map Reduce program in Java and save it to a JAR file. You then specify this JAR file in the properties window for the Map Reduce step, along with some relevant arguments.

The properties for the Map Reduce step in the sample job are shown in the following display:

Display 28.14 *Map Reduce Properties*



Use the **Additional Map Reduce jars** field to specify a JAR file that contains your user-written Map Reduce code. Use the other fields to specify relevant arguments from that code.

Finally, the Map Reduce step in the sample job includes the following advanced options (accessed by clicking **Advanced Options**):

- **Output format class name:** org.apache.hadoop.mapreduce.lib.output.TextOutputFormat
- **Output key class name:** org.apache.hadoop.io.Text
- **Output value class name:** org.apache.hadoop.io.IntWritable

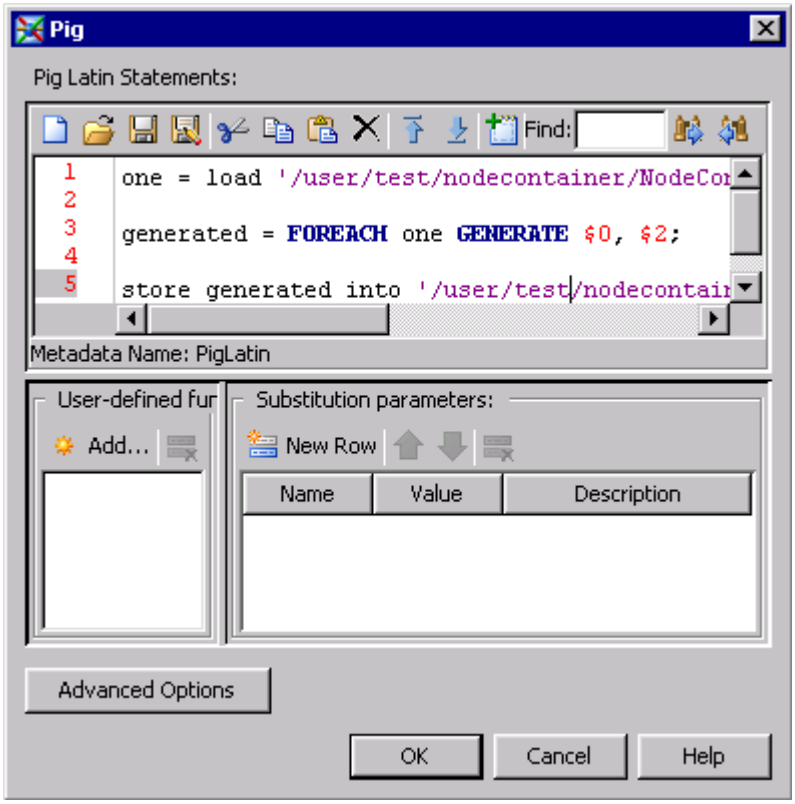
The **Map Reduce Options** section of the Advanced Options window also contains several other options that are not set in the sample job.

Note that the **All files in this location** check box must be selected on the **File Location** tab in the properties window for the Map Reduce target table. This step enables you to see the data in the table after the job has completed successfully.

Configure a Pig Step

The Pig step enables you to submit your own Pig Latin code in the context of a job. It is assumed that you know Pig Latin well enough to use it in a production environment. The properties for the Pig step in the sample job are shown in the following display:

Display 28.15 *Pig Properties*



The Pig step contains the following statements:

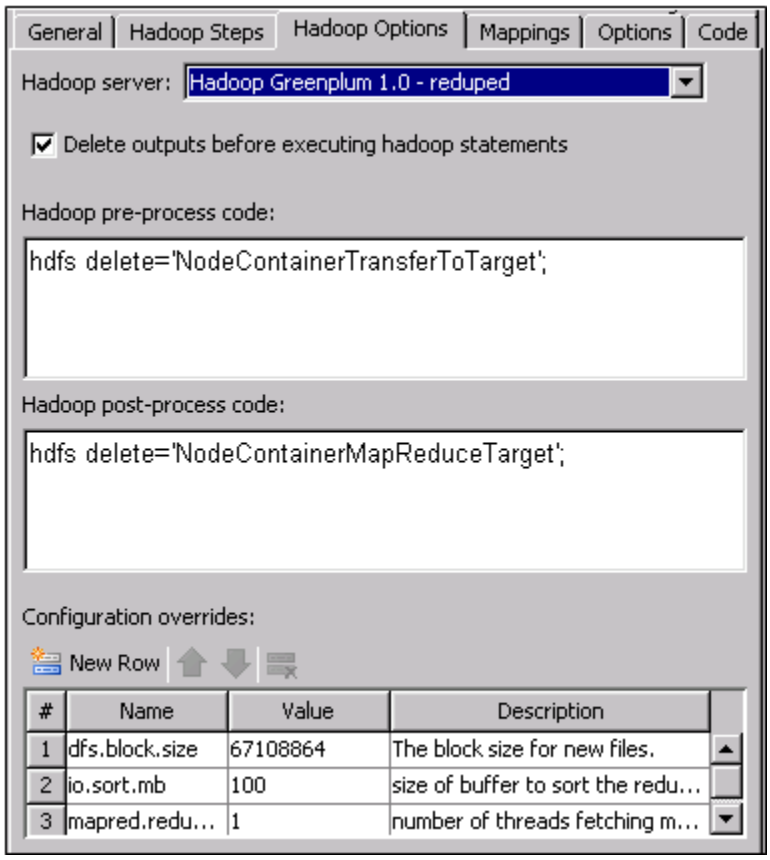
```
one = load '/user/test/nodecontainer/NodeContainerPigSource2.txt' using PigStorage();
generated = FOREACH one GENERATE $0, $2;
store generated into '/user/test/nodecontainer/NodeContainerPigTarget.txt'
USING PigStorage(',')
```

Note that the **All files in this location** check box must be selected in the **File Location** tab in the properties window for the Pig target table. This step enables you to see the data in the table after the job has completed successfully.

Configure the Hadoop Options Tab

The following display shows the **Hadoop Options** tab:

Display 28.16 *Hadoop Options Tab*



The **Hadoop Options** tab enables you to set options such as server selection, output deletion, pre- and post-process code, and configuration overrides for all of the steps. Note that the pre- and post-process code on this tab is run against the Hadoop server only. This code is not the standard pre- and post-process code that is run on the SAS workspace server. Therefore, SAS code is not appropriate input for these fields.

Run the Job and Review the Output

Run the job and verify that the job completes without error. Then, review the output. You should see the following results:

- The Hadoop external file target NodeContainerTransferToTarget has the same 6 observations as its NodeContainerTransferToSource source external file.
- The external file target NodeContainerTransferFromTarget, which is not a Hadoop target, has the same 6 observations as its NodeContainerTransferToTarget source external file.
- The Hadoop external file target NodeContainerMapReduce has 7,816 different words, each in separate observations.
- The Hadoop external File target NodeContainerPigTarget has the same 6 observations as its NodeContainerPigSource2 source external file.

Monitoring Hadoop Clusters

You can access reports that you use to monitor Hadoop clusters by clicking **Hadoop Monitoring** in the **Tools** menu.

After you select a server in the submenu, you can click one of the following reports to select it:

- Job Tracker
- Distributed file system viewer
- Secondary distributed file system viewer

The Job Tracker report provides information about running and completing jobs. The two distributed file system viewer reports provide information about the nodes included in the Hadoop cluster. You can click the links provided in these reports to drill down for more details.

You can also right-click in a completed Pig or Map Reduce job and click **Hadoop Job Statistics** in the pop-up menu to access detailed statistics about the selected job.

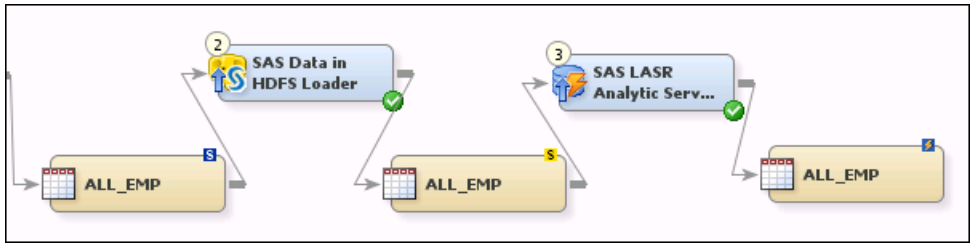
Overview of the High-Performance Analytics Transformations

High-Performance Analytics Transformations

The Transformations tree in SAS Data Integration Studio includes a **High-Performance Analytics** folder. You can use these transformations to load and unload tables on a Hadoop cluster or a SAS LASR Analytic Server. These transformations are typically used to support a SAS Analytics solution that includes both SAS Data Integration Studio and SAS LASR Analytic Server.

For example, SAS Data Integration Studio can be used to support a SAS LASR Analytic Server with the Hadoop Distributed File System (HDFS) as a co-located data provider. You can create a process flow that loads the output table from a transformation to HDFS. Then the HDFS table can be loaded onto the SAS LASR Analytic Server, where it can be analyzed with SAS Visual Analytics. The following display illustrates such a job.

Display 28.17 *Example Flow for High-Performance Analytics Transformations*



The process flow in the previous display includes the following components:

- The first ALL_EMP table in the flow is the output from a previous transformation in the flow. It is a table in a Base SAS library.
- The SAS Data in HDFS Loader reads the first ALL_EMP in the flow. It loads the contents of ALL_EMP into a table with the same physical storage name in a SAS Data in HDFS library. The library is used to make the connection to HDFS.
- The SAS LASR Analytic Server Loader reads the second ALL_EMP in the flow and loads its contents into a table with the same physical storage name in a SAS LASR Analytic Server library. The library is used to make the connection to the SAS LASR Analytic Server cluster and load the table into memory.

The following table describes the High-Performance Analytics transformations.

Name	Description
SAS Data in HDFS Loader	Loads a table to the file system (HDFS) on a Hadoop cluster. The source can be a SAS data set or a table in any DBMS supported by SAS. The target is a table in a SAS Data in HDFS Library.
SAS Data in HDFS Unloader	Unloads a table from HDFS. The input is a table in a SAS Data in HDFS Library.
SAS LASR Analytic Server Loader	Loads a table to memory on a SAS LASR Analytic Server. The source can be a SAS data set, a table in any DBMS supported by SAS, or a table in a SAS Data in HDFS Library. The target is an in-memory table in a SAS LASR Analytic Server Library.
SAS LASR Analytic Server Unloader	Unloads a table from memory on a SAS LASR Analytic Server. The input is an in-memory table in a SAS LASR Analytic Server Library.

See also "Prerequisites for the High-Performance Analytics Transformations" on page 605 and "Usage Notes for HPA Software and Hadoop" on page 608 .

Software Used by These Transformations

SAS LASR Analytic Server

The High-Performance Analytics transformations in SAS Data Integration Studio use the OLIPHANT procedure and the LASR procedure, which are part of SAS LASR Analytic Server.

The SAS LASR Analytic Server is an analytic platform that provides a secure, multi-user environment for concurrent access to data that is loaded into memory. The server can take advantage of a distributed computing environment by distributing data and the workload among multiple machines and performing massively parallel processing. The server can also be deployed on a single machine where the workload and data volumes do not demand a distributed computing environment.

For distributed deployments, having local storage available on machines is critical in order to store large data sets in a distributed form. The SAS LASR Analytic Server supports HDFS as a co-located data provider. HDFS is used because the server can read from and write to HDFS in parallel. In addition, HDFS provides replication for data redundancy. HDFS stores data as blocks in distributed form on the blades and the replication provides failover capabilities.

For more information about this server and related software, see the *SAS LASR Analytic Server: Reference Guide*. This book is available at the following location: **http://support.sas.com/documentation/onlinedoc/securedoc/index_lasrserver.html**

SAS Data in HDFS Library

The second ALL_EMP table in the previous display is stored in a SAS Data in HDFS library. This is a library for tables that are stored in the Hadoop Distributed File System (HDFS). The library works only with SASHDAT files that are created with the OLIPHANT procedure or with the SAS Data in HDFS Engine. SASHDAT is the data format used for SAS tables that are added to HDFS.

OLIPHANT Procedure

The SAS Data in HDFS loader and unloader use the OLIPHANT procedure to add, delete, and manage SASHDAT files that are stored in the Hadoop Distributed File System (HDFS). You can manually specify some OLIPHANT options in the transformation. For more information about this procedure, see the *SAS LASR Analytic Server: Reference Guide*.

LASR Procedure

The SAS LASR Analytic Server transformations use the LASR procedure to load and unload tables from that server. You can manually specify some LASR options in the transformation. For more information about this procedure, see the *SAS LASR Analytic Server: Reference Guide*.

SAS LASR Analytic Server Library

The third ALL_EMP table in the previous display is stored in a SAS LASR Analytic Server library. This is a library used to register tables that are loaded in memory on a SAS LASR Analytic Server.

Prerequisites for the High-Performance Analytics Transformations

For SAS Data Integration Studio Users

High-Performance Analytics transformations have the following unique prerequisites:

- SAS Data in HDFS Loaders require a SAS Data in HDFS library for their target tables. Administrators should let you know which libraries are available to you.
- SAS LASR Analytic Server Loaders require a SAS LASR Analytic Server library for their target tables. Administrators should let you know which libraries are available to you.
- In order to submit a job with any High-Performance Analytics transformation, you must have login credentials that are configured for Passwordless Secure Shell (SSH) on the machines in the High-Performance Analytics cluster. Contact your administrator to obtain these login credentials.

Administrators usually set up the servers, libraries, and users that are required by the High-Performance Analytics transformations. SAS Data Integration Studio users are

simply told what user credentials to use and which libraries should be specified in a given job. See also "Usage Notes for HPA Software and Hadoop" on page 608.

For Administrators

Libraries, Users, and Servers Are in the Same Metadata Environment

The servers, libraries, and user IDs that are required by the High-Performance Analytics transformations are assumed to be registered in the same metadata repositories on the SAS Metadata Server. This is the usual configuration when SAS Data Integration Studio and SAS LASR Analytic Server are installed as part of a SAS Analytics solution.

Verify HDFS and LASR Libraries

An initial set of libraries might be configured when SAS Data Integration Studio and SAS LASR Analytic Server are installed as part of a SAS Analytics solution. This includes SAS Data in HDFS libraries and SAS LASR Analytic Server libraries. These libraries are used to connect to HDFS and the SAS LASR Analytic Server. Let SAS Data Integration Studio users know which libraries are available to them.

For more information about registering SAS Data in HDFS libraries and SAS LASR Analytic Server libraries, see the" Connecting to Common Data Sources" chapter in *SAS Intelligence Platform: System Administration Guide*. This book is available at: **`http://support.sas.com/documentation/onlinedoc/intellplatform/`**

Set Up Passwordless Secure Shell (SSH) Access for Selected Users

Note: This is a required, post-installation task for administrators.

Each user who wants to submit a job that includes a High-Performance Analytics transformation must log on to SAS Data Integration Studio with a special operating system user ID. This ID must be configured for Passwordless Secure Shell (SSH) on the machines in the High-Performance Analytics cluster. One way to do this is to configure the operating system user IDs of appropriate SAS Data Integration Studio users. The operating system user ID is specified in the metadata connection profile for the SAS Data Integration Studio user.

For more information about this task, see in the sections about SSH in *SAS LASR Analytic Server: Reference Guide*. This book is available at: **`http://support.sas.com/documentation/onlinedoc/securedoc/index_lasrserver.html`**.

Loading a Table on the SAS LASR Analytic Server

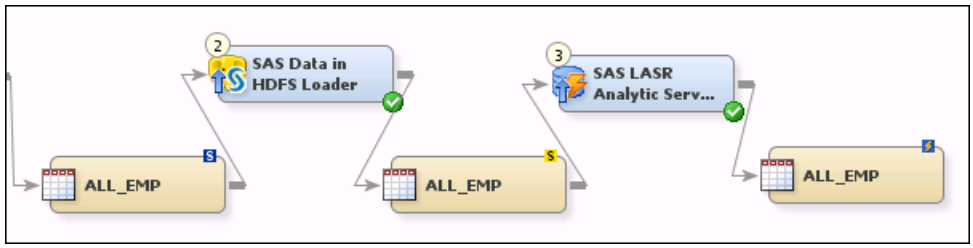
Problem

You want to load the output table from a transformation to the Hadoop Distributed File System (HDFS). The HDFS table should then be loaded onto the SAS LASR Analytic Server, where it can be analyzed with SAS Visual Analytics.

Solution

You can create a process flow similar to the one shown in the next display.

Display 28.18 *Example Flow for High-Performance Analytics Transformations*



The process flow in the previous display includes the following components:

- The first ALL_EMP table in the flow is the output from a previous transformation in the flow. It is a table in a Base SAS library.
- The SAS Data in HDFS Loader reads the first ALL_EMP in the flow. It loads the contents of ALL_EMP into a table with the same physical storage name in a SAS Data in HDFS library. The library is used to make the connection to HDFS.
- The SAS LASR Analytic Server Loader reads the second ALL_EMP in the flow and loads its contents into a table with the same physical storage name in a SAS LASR Analytic Server library. The library is used to make the connection to the SAS LASR Analytic Server cluster and load the table into memory.

It is assumed that you are familiar with how to create a SAS Data Integration Studio job, as described in "Creating a Process Flow for a Job" on page 148 . Perform the following tasks.

- "Verify Prerequisites for the HPA Loaders" on page 607
- "Register Target Tables for the HPA Loaders" on page 607
- "Create and Run a Job with HPA Loaders" on page 608

Tasks

Verify Prerequisites for the HPA Loaders

See the prerequisites topic "For SAS Data Integration Studio Users " on page 605.

Register Target Tables for the HPA Loaders

The second and third tables in Display 28.18 on page 607 do not yet exist. They are created the first time the job is run. In order to create a flow similar to the one shown in the display, use the New Table wizard to register the new tables. 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.

In the current example, you register one target table in a SAS Data in HDFS library. Then you register another target table in a SAS LASR Analytic Server library.

Note: See "Target Table for HPA Transformations Must Have the Same SAS Name as the Source Table" on page 608.

Create and Run a Job with HPA Loaders

Create a job with a flow similar to that shown in Display 28.18 on page 607. Run the job and verify the output.

Usage Notes for HPA Software and Hadoop

High-Performance Analytics Software

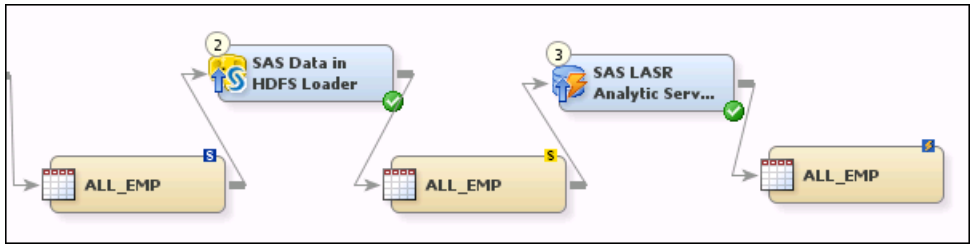
Target Table for HPA Transformations Must Have the Same SAS Name as the Source Table

The properties window for a table enables you to specify both a descriptive name and a physical storage name (SAS name) for the table. The descriptive name is specified on the **General** tab. The physical storage name is specified on the **Physical Storage** tab.

In a SAS Data Integration Studio job, the physical storage name for the target of a SAS Data in HDFS Loader must match the physical storage name of the source table. The same is true for the target of a SAS LASR Analytic Server Loader.

For example, suppose that you had a job with a flow similar to that of the next display.

Display 28.19 *Example Flow for High-Performance Analytics Transformations*



In the previous flow, the SAS Data in HDFS Loader reads a source table with the physical storage name of ALL_EMP. It writes a target table with the same physical name. Likewise, the SAS LASR Analytic Server Loader reads a source table with the physical storage name of ALL_EMP, and it writes a target table with the same physical name. Otherwise, the job would fail. This requirement is due to the underlying SAS procedures that are used by these transformations.

Using HDFS Tables and LASR Tables in Jobs

Tables in a SAS Data in HDFS library can be used as the target of the following transformations: Extract, Sort, Append, Splitter, Frequency, Compare Tables, SQL Create Table, SQL Join, SQL Set Operators. The same is true for tables in a SAS LASR Analytic Server library.

Turn Off Collect Table Statistics for Jobs with the SAS Data in HDFS Unloader

The properties window for a SAS Data Integration Studio job enables the **Options tab** ⇨ **Diagnostics/Statistics panel** ⇨ **Collect Statistics option** by default. If this option is

enabled for a job with a SAS Data in HDFS Unloader, the job will fail with the following error:

"ERROR: The SASHDAT engine is a uni-directional engine. Data flows from the SAS client to the Hadoop Distributed File System. The engine cannot be used to fetch data from HDFS."

If you deselect the **Collect Statistics** option for the job, the job will run correctly.

View Data Window Cannot Display Tables in an HDFS Library or a LASR Library

The View Data window cannot display tables in a SAS Data in HDFS Library or a SAS LASR Analytic Server Library. For more information about the View Data window, see "Browsing Table Data" on page 111.

Hadoop

General Usage Notes for Cloudera Impala

The following usage notes apply to Cloudera Impala, a query engine that runs on Apache Hadoop.

Impala does not support SQL DELETE or UPDATE statements. Accordingly, any SAS Data Integration Studio transformation that generates SQL DELETE or UPDATE statements will fail if these operations are attempted on an Impala table. Affected transformations include the SQL Delete transformation, the SQL Update transformation, and the Table Loader transformation. For more information about the loader transformation, see "Cloudera Impala: Usage Notes for Loaders" on page 669.

Impala Requires SQL ORDER BY statements to specify a limit. The SQL Join transformation does not specify a limit on an ORDER BY statement by default. Accordingly, if an SQL Join transformation attempts to execute an ORDER BY statement, the transformation will fail. If explicit pass-through is set for the SQL Join, one remedy is to add the limit clause in the generated code.

One way to add the limit clause is as follows:

1. Open the properties window for the SQL Join.
2. Click the **Code** tab.
3. Change the Code generation mode to All user written.
4. Add the limit clause after the ORDER BY clause.

   Here is an example where the limit is 100 observations:

   ```
   order by <column> limit 100
   ```

5. Click **OK** to close the properties window and save your changes.

If implicit pass-through is set for the SQL join, one remedy is to specify a data set option, such as *OBS=100*, in the properties window for the Impala table.

One way to specify a data set option is as follows:

1. Open the properties window for the table.
2. Select **Options tab** ⇨ **General** ⇨ **Additional Table Options**.
3. Specify an option such as *OBS=100*.

Usage Notes for Table Loader and Cloudera Impala Tables

Cloudera Impala is optimized for queries rather than inserts into HDFS. Accordingly, avoid using a table in a SAS Impala library as the target of any table loader transformation.

Impala does not support SQL DELETE or UPDATE statements. Accordingly, any SAS Data Integration Studio transformation that generates SQL DELETE or UPDATE statements will fail if these operations are attempted on an Impala table. Affected transformations include the Table Loader transformation. Specifically, the following combination of load options on the **Load Techniques** tab do not work for Impala tables: **Load style**: Update/Insert and **Matching Rows**: SQL Set.

Part 4

Appendixes

Appendix 1
Main Windows and Wizards *613*

Appendix 2
Usage Notes *643*

Appendix 3
Miscellaneous Transformations *673*

Appendix 4
Java Code and Methods for Report Plug-ins *763*

Appendix 1

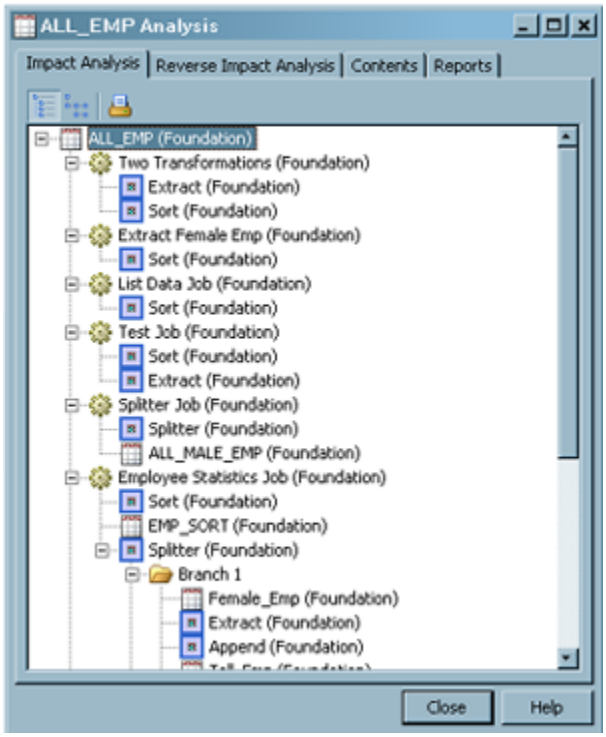
Main Windows and Wizards

Analysis Window **614**
Checkouts Tree **615**
Code Editor **615**
Comparison Results Window **616**
Connection Profile Window **617**
Desktop **617**
Details Pane **619**
Expression Builder **620**
Overview 620
Database Functions 621
User-Defined Functions 622
Folders Tree **625**
Inventory Tree **625**
Job Editor **629**
Properties Windows **631**
Basic Properties 631
Job Properties 631
Transformation Properties 632
Table Properties 632
Reports Window **634**
Tools-Options Window **634**
Tree View **635**
View Data Windows **637**
View Data Window 637
View File Window 638
Wizards **639**
New Object Wizards 639
Register Tables Wizards 640
Cube Wizards 640
Data Surveyor Wizards 641
Metadata Import and Export Wizards 641

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:

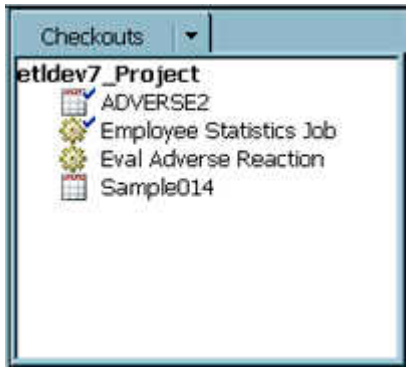
- **Impact Analysis**: identifies the data stores, columns, jobs, and transformations that are *affected by* a change in a selected object. For more information, see "Performing an Impact Analysis" on page 313.
- **Reverse Impact Analysis**: identifies the data stores, columns, jobs, and transformations that *contribute to* the content of a selected object. For more information, see "Performing Reverse Impact Analysis " on page 317.
- **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 763.

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*

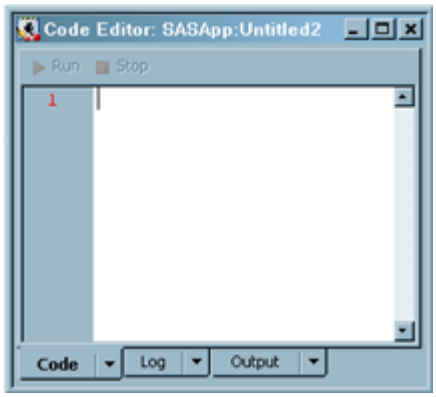


Metadata that has been checked out for update has a check mark beside it, such as the first two objects in the preceding display. New metadata objects that have never been checked in do not have a check mark beside them, such as the last two objects in the preceding display. For more information about change management, see "Working with Change Management" on page 47.

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.

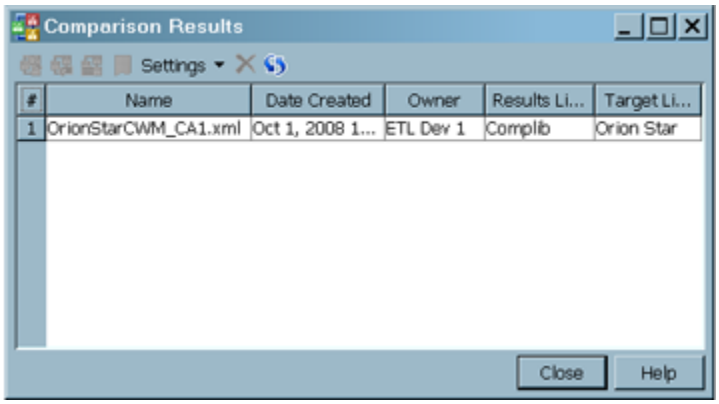
To display the Code Editor window, select **Tools** ⇨ **Code Editor** from the desktop. To submit code for execution, click **Run** on the **Code Editor** toolbar. Alternatively, you can select **Actions** ⇨ **Run** from the desktop. To display Help for the **Code Editor**, press the F1 key. To customize 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*

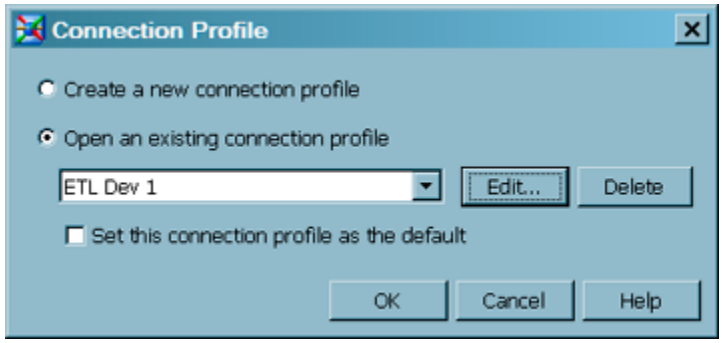


When you select a comparison result, the icons on the toolbar are activated. You can click these icons to view the differences between the imported metadata and existing metadata, or to perform other tasks. For more information, see “Importing Updated Metadata with a SAS Metadata Bridge” on page 70.

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*

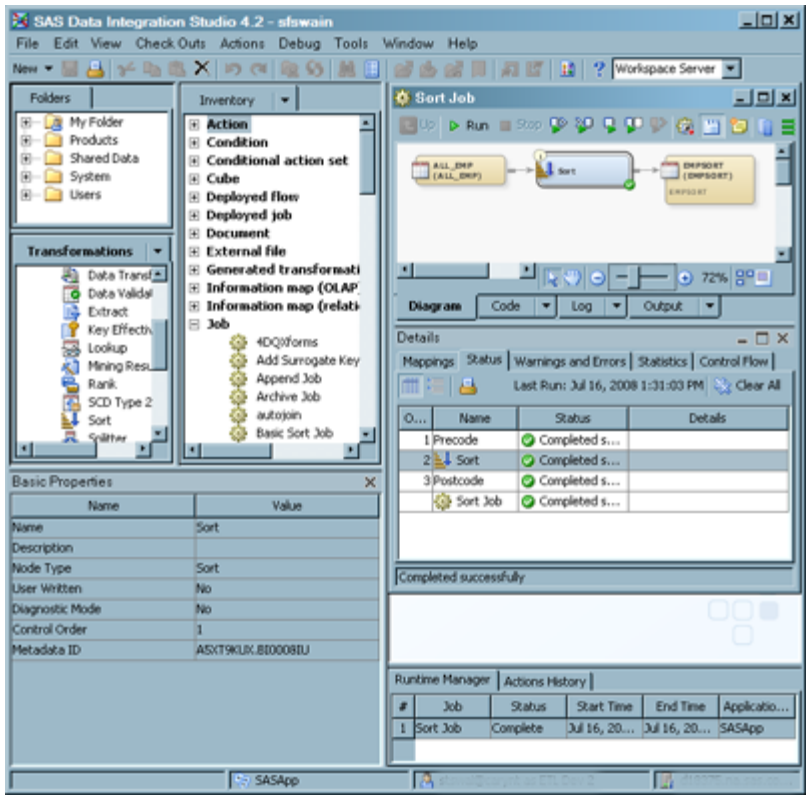


A connection profile enables you to connect to a SAS Metadata Server. You cannot do any work until you open an existing profile or create a new one. For more information, see "Connecting to a SAS Metadata Server" on page 22.

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*

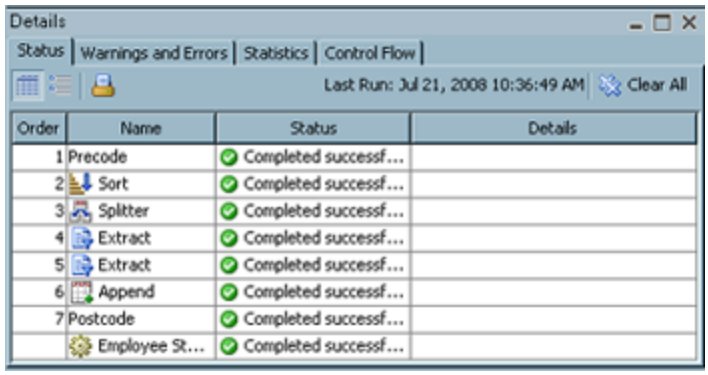
Component	Location	Description
Title bar	Top of the desktop	Shows the current version of SAS Data Integration Studio and the name of the current connection profile.
Menu bar	Under the title bar	Provides access to the drop-down menus. The list of active options varies according to the current work area 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 635.

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 **View** ⇨ **Basic Properties** from the desktop. For more information, see "Properties Windows" on page 631.
Status bar	Bottom of the desktop	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. To select a different SAS Application Server, double-click the name of that server to display a dialog box. 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.
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 **Open**. For more information, see "Job Editor" on page 629.
Details pane	Under the Job Editor	Used to monitor and debug a job in the Job Editor. To display this pane, select **View** ⇨ **Details** from the desktop. For more information, see "Details Pane" on page 619.
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 **View** ⇨ **Runtime Manager** from the desktop.
Actions History	Under the Details pane	Displays low-priority errors and warnings. To display this pane, select **View** ⇨ **Actions History** 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
Status	Used to display the status of each step in a submitted job.
Warnings and Errors	Used to display any warnings and errors that are generated when a job is submitted.
Statistics	Used to display run-time and table statistics that are generated by a submitted job. Includes tabular and graphical displays.
Control Flow	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.
Columns	Used to review and update columns in a table or external file in a job.
Mappings	Used to review and update mappings for transformations in a job.

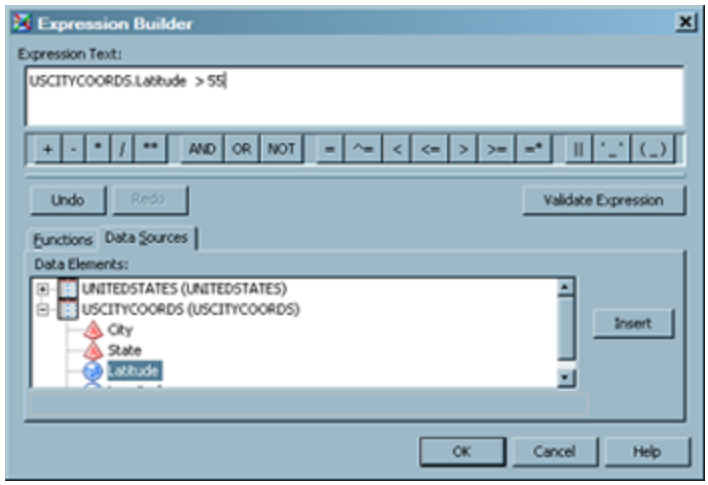
The **Status**, **Warnings and Errors**, **Statistics**, and **Control** tabs are displayed whenever the Details pane is enabled for an opened job. The **Columns** tab is displayed when a table or external file in a job is selected. The **Mappings** tab is displayed when a transformation is selected.

Expression Builder

Overview

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.

The **Functions** tab of the Expression Builder enables you to select SAS functions, formats, and other components and add them to an expression. Documentation for a selected function is displayed to the right of the function. For details about SAS expressions, see *SAS Language Reference: Concepts*.

The Expression Builder supports the following specialized function types:

- “Database Functions” on page 621
- “User-Defined Functions” on page 622

Database Functions

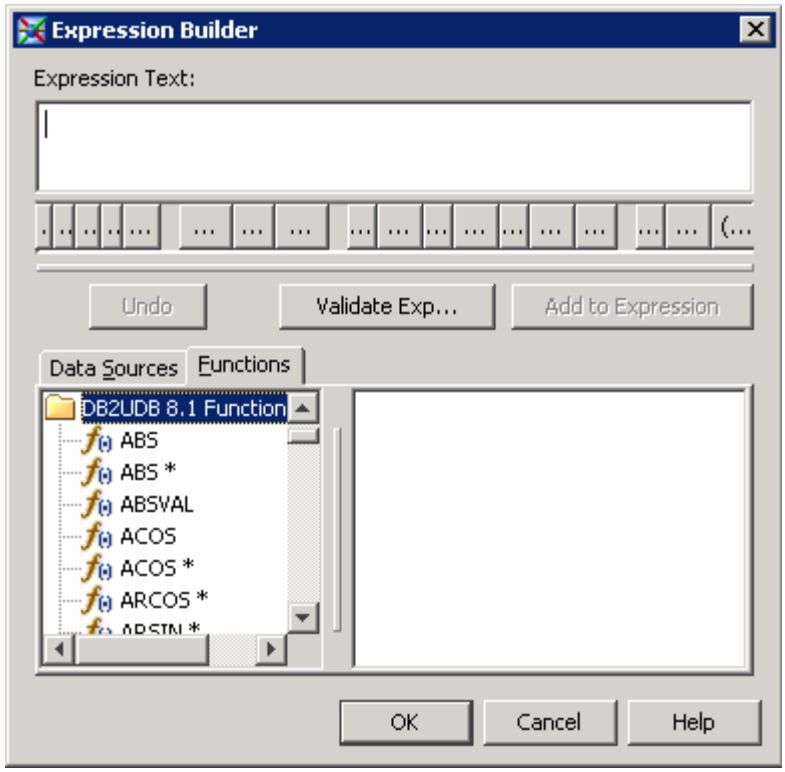
The **Functions** tab in the Expression Builder window contains a list of common functions that you can use in expressions. It also contains specialized functions for the following relational databases:

- DB2 10.1 Functions
- DB2 8.1 Functions

- FedServer TKSQL Functions
- Greenplum Functions
- MySQL
- ODBC
- Oracle
- Netezza Functions
- Postgres Functions
- SQL Server
- Teradata
- Vertica Functions

The following display shows a portion of the functions available for DB2UDB 8.1 in the Expression Builder window:

Display A1.9 *DB2UDB 8.1 Functions*



The functions that are marked with * (such as ABS*) can be pushed down for native processing in the database. For information about these native functions, see the documentation for the databases.

User-Defined Functions

Overview

You can import user-defined functions for models registered through Model Manager for DB2, Teradata, and Netezza databases. You can also import native user-defined

functions from Oracle, DB2, and Teradata databases. After you import these user-defined functions, you can find them in the **Functions** tab of the Expression Builder window.

This feature supports standard DBMS user-defined functions and Enterprise Miner publishing Mining Analytics user-defined functions, standard user-defined functions and Enterprise Miner publishing Mining Analytics user-defined functions, and single-click mapping for column inserts. With single-click mapping, you can select a user-defined function in the **Function** tab and double-click a value in the **Data Sources** tab to insert it in the expression. For example, the expression *ABS(CONTINENTS_NONAMERICAS.Area)* draws *ABS* from a selected value on the **Function** tab and *CONTINENTS_NONAMERICAS.Area)* from the double-clicked value *Area* on the **Data Sources** tab.

Importing User-Defined Functions

Perform the following steps to import user-defined functions:

1. Open the **Import User Defined Functions** window from the **Tools** menu in SAS Data Integration Studio.
2. Enter the name of a container in the **Container** field or select a container from the drop-down list.
3. Select the library for the container. The available function sets are displayed in the **Available** field.
4. Move one or more functions sets to the **Selected** field.
5. Click **Preview** to review the functions contained in the selected functions sets.
6. Click **Import** to create an XML file named UserDefinedFunctions.xml. This file makes the imported user-defined functions available for use in the **Expression Builder** window.
7. Click **Close** to close the window when you have finished importing user-defined functions.

   Note that if one or more function sets share names with previously imported function sets, you are warned that the previous function sets exist. If you choose to proceed with the import, the new function sets replace the identically named function sets.

Note: You can use the user-defined functions without modification if you enable explicit pass-through. To enable pass-through, select **`Yes`** in the **Use the optimized pass-through facility for SQL statements** field. The field is located in the **Options** tab in the SQL Join transformation. If you need to use user-defined functions in a job that uses implicit pass through, perform the steps in "Enable Implicit Pass-Through Processing for User-Defined Functions" on page 623.

Note: If you import user-defined functions from an Oracle library, the SAS metadata for that library should specify a SCHEMA of SYS, or should specify no values for SCHEMA. Otherwise, the import will fail. To update the SCHEMA value in the SAS metadata for an Oracle library, open the properties window for that library in SAS Integration Studio and click the **Data Server** tab.

Enable Implicit Pass-Through Processing for User-Defined Functions

If you want to process user-defined functions using implicit pass-through, you must link your DBMS user-defined functions to SAS functions. These SAS functions must have the same names and same returned values as the user-defined functions. You can define the SAS functions with PROC FCMP.

Perform the following steps:

1. Create the DBMS user-defined function on the database server. For example, you could create the following Oracle function:

```
Create FUNCTION ora_udf(in_var IN NUMBER)
      RETURN NUMBER
   IS
   RESULT_VAR NUMBER;
   BEGIN
      RESULT_VAR := 5;
      RETURN (RESULT_VAR);
    END;
```

2. Launch SAS Data Integration Studio and create a DBMS library that connects to the database server. Then, import the DBMS user-defined function into the library through the process described in "Importing User-Defined Functions" on page 623. For example, you could create an Oracle library that contains ora_udf.

3. Open a job with appropriate data and registrations. In this case, the job contains a table with a numeric column and the SQL Join transformation. Right-click the temporary output table and make sure that the **Create as view check box** is deselected. Also, make sure that the temporary output table is redirected to the Oracle library that contains the DBMS user-defined function. For information about redirection, see "Redirecting Temporary Output Tables" on page 196.

4. Create a data set to store the new DBMS function. To do this, open the **Precode and Postcode** tab in the properties for the job. Then insert code that creates a SAS function and connects it to the user-defined function that you created for the DBMS. The sample job contains the following code:

```
data work.newfunc;
       SASFUNCNAME = "ORA_UDF";
       SASFUNCNAMELEN = 7;
       DBMSFUNCNAME = "ORA_UDF";
       DBMSFUNCNAMELEN = 7;
       FUNCTION_CATEGORY = "CONSTANT";
       FUNC_USAGE_CONTEXT = "WHERE_ORDERBY";
       FUNCTION_RETURNTYP = "NUMERIC";
       FUNCTION_NUM_ARGS =1;
       CONVERT_ARGS = 0;
       ENGINEINDEX = 0;
       output;
    run;

    OPTIONS CMPLIB=work.newfuncs;
    PROC FCMP OUTLIB=work.newfuncs.ORA_UDF;
       FUNCTION ORA_UDF(a);
       RETURN(5);
       ENDSUB;
     RUN;
```

   Note that this SAS function has the same name and returned values as the DBMS user-defined function.

5. Set the options necessary to add the SAS function to the existing in-memory SAS function list. To open the **Other Options** tab for the DBMS library, select **Properties ⇨ Options ⇨ Advanced Options**. These options are set in the **Options to be appended** field . The following options are set for the sample job:

```
sql_functions="EXTERNAL_APPEND=work.newfunc" sql_functions_copy=saslog
```

After these steps are completed, you can process user-defined functions in jobs that use implicit pass-through.

Changing the User-Defined Functions Storage Directory

You can configure a system property with the name UserDefinedFunctionsPath. Use this property to specify the file system path of the location of the user-defined function.xml file. You can specify this property in the distudio.ini file in the SAS Data Integration Studio installation directory. Use the following syntax:

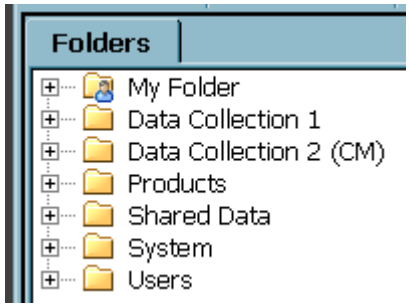
```
JavaArgs_<#>=-DUserDefinedFunctionsPath="<path>"
```

Be sure to give the argument a unique number and add the filename to the end of the path.

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 A1.10 *Example Folders in the Folders Tree*



For more information, see "Working with the Folders Tree" on page 24.

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 **Skip the Record** and **Send Email**, that can be performed when certain conditions are met during the execution of a job. You cannot add or update actions.
Business Rule Flow		Metadata for an XML file containing ordered rules, conditional events, and analytical models that can be deployed for execution to produce a decision.
Condition		Metadata for a Status Handling conditions. SAS provides a number of conditions, such as **Successful** and **Error in Process**, 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.
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).
Decision Logic		Metadata for an Enterprise Decision Management flow, which used to structure the input data in an Enterprise Decision Management transformation into output data.
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.
Dimension (shared)		Metadata for a dimension that is used by multiple cubes.

Folder Name	Icon	Description
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.
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.

Folder Name	Icon	Description
Prompt group		Metadata for prompt groups. In general, do not add or update prompt groups in SAS Data Integration Studio.
Search folder		Metadata for saved searches that you can initiate from the **Tools** menu and save for future use.
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.

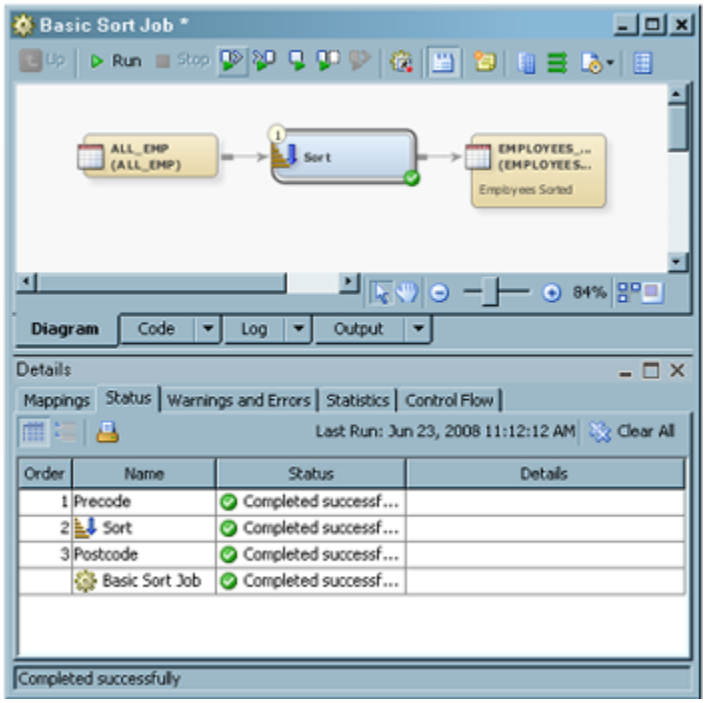
Table A1.4 *Icon Overlays for Metadata Objects in the Inventory Tree and Folders Tree*

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.
	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 A1.11 *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
Diagram	Always displayed.	Used to build and update the process flow for a job.
Code	Select **Tools** ⇨ **Options** from the desktop. On the **General** tab, select **Show Code Tab**.	Used to review or update code for a job.
Log	Select **Tools** ⇨ **Options** from the desktop. On the **General** tab, select **Show Log Tab**.	Used to review the log for a submitted job.
Output	Select **Tools** ⇨ **Options** from the desktop. On the **General** tab, select **Show Output Tab**.	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 **View** ⇨ **Details** from the desktop.	Used to monitor and debug a job in the Job Editor. For more information, see "Details Pane" on page 619.
Runtime Manager	Select **View** ⇨ **Runtime Manager** from the desktop.	Displays the run-time status of the current job, the last time 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 **View** ⇨ **Actions History** from the desktop.	Displays low-priority errors and warnings.

A modifier icon called an icon overlay indicates that an object is in a certain state or has special attributes. Put the cursor on an overlay to view a description of what the overlay means. The following table describes some of the overlay icons for metadata objects in the Job Editor window.

Icon Overlay	Description
	A vertical arrow pointing down in an icon for a transformation indicates that an implicit data transfer is taking place. An implicit data transfer can be inefficient when you have large amounts of data. You can add the Data Transfer transformation in these situations.
	A horizontal green triangle in the upper right corner of the icon for a transformation indicates that a checkpoint has been added to the transformation. Checkpoints are used to restart jobs.
	A rectangle in the upper right corner of the icon specifies the data format of the table, such as SAS or Teradata.

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
General	Enables you to enter general information that identifies, describes, and locates the job.
Code	Enables you to review and modify the code that is generated for the job.
Precode and Postcode	Enables you review and modify user-written code that is inserted at the beginning or end of the job.
Status Handing	Enables you to review and modify status handling conditions and actions for the job. (Some transformations have this tab as well.)
Parameters	Enables you to review and modify parameters for the job.
Options	Enables you to review and modify options for the job.
Notes	Enables you to review and modify notes for the job.
Extended Attributes	Enables you to review and modify extended attributes for the job.

Tab	Description
Authorization	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
General	Enables you to enter general information that identifies and describes the transformation.
Mappings	Enables you to review and modify the mappings for the transformation.
Options	Enables you to review and modify options for the transformation.
Table Options	Enables you to review and modify table options for the transformation.
Code	Enables you to review and modify the code that is generated for the transformation.
Precode and Postcode	Enables you review and modify user-written code that is inserted at the beginning or end of the transformation.
Parameters	Enables you to review and modify parameters for the transformation.
Notes	Enables you to review and modify notes for the transformation.
Extended Attributes	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
General	Enables you to enter general information that identifies and describes the table.
Columns	Enables you to maintain column metadata.
Indexes	Enables you to review, add, and modify indexes on table columns.
Keys	Enables you to review, add, and modify key columns.
Parameters	Enables you to review and modify parameters for the table.
Physical Storage	Enables you to specify the format and location of a table.
Options	Enables you to review and modify options for the table.
Notes	Enables you to review and modify notes for the table.
Extended Attributes	Enables you to review and modify extended attributes for the table.
Authorization	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*

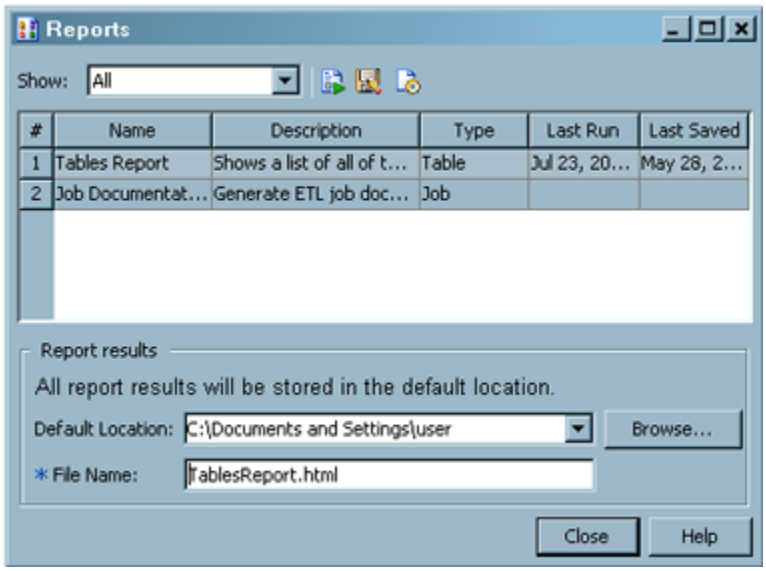
Category	Icon	Description
Column		Metadata for a character column.
		Metadata for a numeric column.
Index		Metadata for an index.

Category	Icon	Description
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 A1.12 *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

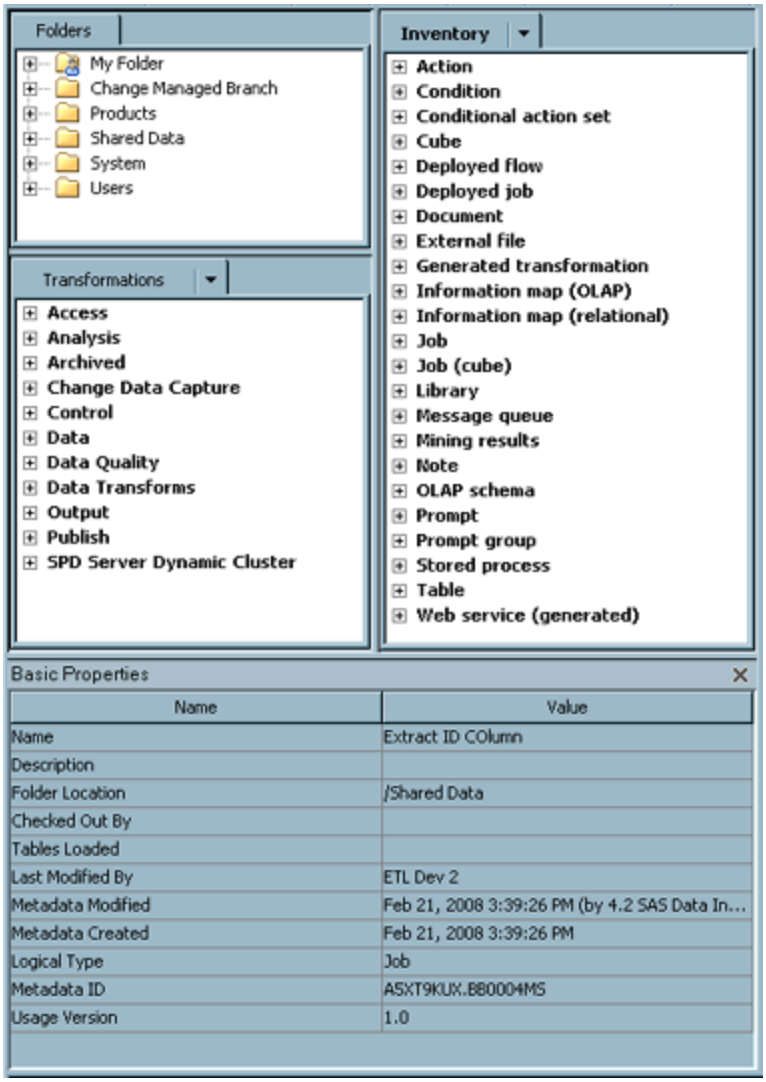
The Options window is used to specify global options for SAS Data Integration Studio. To display this window, select **Tools** ⇨ **Options** from the desktop. The following table describes the purpose of each tab in the Options window. For more information about each tab, see the Help for that tab.

Table A1.11 Option Window Tabs

Tab	Description
General	Specifies general user interface options, such as whether SAS Data Integration Studio should prompt before discarding changes to metadata.
Job Editor	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.
Code Editor	Specifies interface options for the **Code** tab in the Job Editor window, such as whether to display line numbers.
SAS Server	Specifies the default SAS Application Server for SAS Data Integration Studio and enables you to set options for submitting jobs to a grid.
View Data	Specifies interface options for the View Data window, such as whether View Data should prompt before proceeding with a lengthy navigation operation.
Code Generation	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.
Data Quality	Specifies options that are used by the transformations in the **Data Quality** folder of the Transformations tree. For example, you can specify the location of the DQ setup file.

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 A1.12 *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. **My Folder** and **Shared Data** are the folders that you use most of the time. For more information, see "Folders Tree" on page 625.

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 "Inventory Tree" on page 625.
Transformations tree	Displays by default.	Displays transformations that can be dragged and dropped into SAS Data Integration Studio jobs. For more information, see "Overview of Transformations" on page 32.
Basic Properties pane	Select **View** ⇨ **Basic Properties** 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 "Checkouts Tree" on page 615.

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.

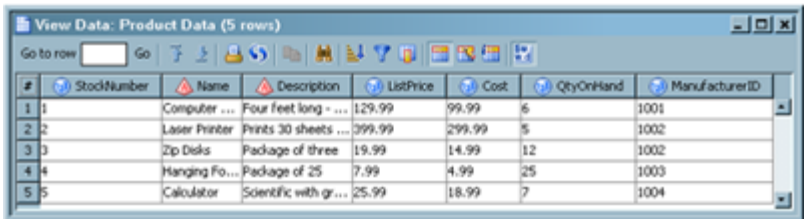
Use the edit mode to perform simple editing operations on the data in the View Data window. For example, you can overwrite the data in a cell, copy and paste rows of data, and delete data. You can even create completely new tables. However, this editing mode is enabled only on SAS tables that are stored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



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.

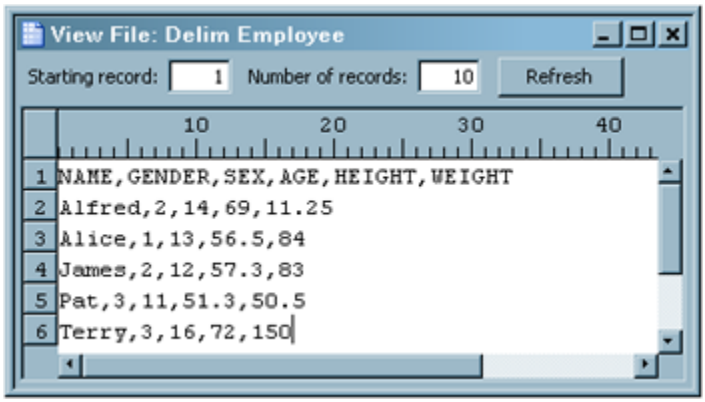
View File Window

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*



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 A1.13 *New Object Wizards*

Wizard	Description
New Folder	Adds a folder in the **Folders** 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 A1.14 *Cube Wizards*

Wizard	How to Display the Wizard	Description
Cube Designer (add mode)	Right-click an appropriate destination folder in the **Folders** tree, and then from the desktop select **New** ⇨ **Cube**.	Creates and registers a SAS OLAP cube.
Cube Designer (update mode)	Right-click a cube, and then select **Edit Cube Structure**.	Maintains a cube.
Aggregation Tuning	Right-click a cube, and then select **Aggregation Tuning**.	Updates aggregations.
View Cube	Right-click a cube, and then select **View Cube**.	Displays the cube.
Export Code	Right-click a cube, and then select **Export Code**.	Saves the code for the cube to a file.
Calculated Members	Right-click a cube, and then select **Maintain** ⇨ **Calculated Members**.	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 **Maintain** ⇨ **Change OLAP Schema**.	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 A1.15 *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 **Export** ⇨ **SAS Package**.	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 **Import** ⇨ **SAS Package**.	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 A1.16 *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 **Export** ⇨ **Metadata**.	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 **Import** ⇨ **Metadata**.	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

Usage Notes

General Usage Notes **645**
Avoid Double Quotation Marks in DBMS Table and Column Names 645
Concurrent Queries to Teradata Tables Can Fail 645
Create as View Option Works Only When It Is Possible to Create a View 645
Data Transfer Does Not Work for DBMS Tables with Special Characters in Table Names 646
DBMS-Specific Functions Work Only with Explicit Pass-Through 646
Do Not Include the << and >> Signs in the Column Descriptions of a Table That Could Be Used in an Information Map 646
Do Not Use MLE Library Tables as Targets in SAS Data Integration Studio Jobs 646
Enhanced Validation for Generated Code Can Be Turned Off 646
External File Wizard: Cannot Sort Displayed Data by Column 647
Jobs with Implicit Data Transfers between Different Hosts Fail 647
Limit Line Lengths in the Expression Builder to 128 Characters 648
Maximum Integer Value for a Field in SAS Data Integration Studio 648
Microsoft Queue Writer Transformation Does Not Transform Missing Date Values 648
Migrating from SAS/Warehouse Administrator to SAS Data Integration Studio . 648
Out-Of-Date Generated Transformations Are Updated Automatically When Included in Jobs That Are Deployed or Redeployed 648
SAS Data Integration Studio Jobs Cannot Output HP Neoview Tables That Have Keys 648
SAS Metadata Bridges 649
Saving Metadata Changes in the Corresponding Physical Target 649
Saving Temporary Output Tables to a Permanent Location 649
Sign-on Scripts for SAS/CONNECT Servers 649
SQL Join Transformations: Adding Multiple Sources with Primary Keys to an SQL Join Transformation Job 649
SQL Merge Transformations: Input and Output Issues 650
SQL Merge Transformations: Use the SQLNCLI10 Driver to Access SQL Server 2008 650
SQL Update and SQL Merge Transformations: Manual Updates Are Required If Subqueries Are Used 650
Synchpoint Option on WebSphere Queue Reader Does Not Prevent All Data Commits to Target When an Error Occurs 650
Transformations That Use PROC SQL Rename SAS Reserved Words Unless Case Sensitive Names and Special Characters Are Recognized 651
Unrestricted Users Cannot Perform Tasks That Require Logins from the Metadata Server 651

Update Table Metadata Cannot Be Used for Some Tables 651
Update Table Metadata on z/OS Platforms 651
Updates Might Be Needed to the Names of Primary Keys for Oracle Tables 651
Very Large Tables or Files in WebSphere Message Queues Can Cause Memory Overruns 652

Usage Notes for Register Tables Wizards and the New Table Wizard 653
Access to Data on z/OS Platforms 653
Access to Tables Using ODBC DB2 z/OS Pass-through 653
Case and Special Characters in DBMS Names 653
Case and Special Characters in SAS Names 653
Control Whether SAS Formats and Informats Are Automatically Applied to Table Columns 653
Generic Register Tables Wizard: When to Use 654
Importing Foreign Keys for DBMS Tables 654
Importing Foreign Keys for SAS Tables 654
Importing Keys and Indexes from SAS/SHARE Libraries 654
LIBNAME Options Required for Support of Case and Special Characters in the Names for Keys and Indexes 654
In a Register Tables Wizard, Limit Excel Connections to a Single User Name . . 655
Limitations of Register Tables Wizards for MySQL and Informix 655
Metadata for a Library and Its Tables Must Be Stored in the Same Metadata Repository 655
Microsoft Windows Administrative Rights Required to Connect to OLE DB Data Sources 655
ODBC Informix Library: Preserving Case in Table Names 656
Registering SAS/SHARE Tables 656
Registering a Sybase Table with a Clustered Index 656
Separate Logon Credentials for Each Authentication Domain for Database Servers 656
Setting Table Options in the New Table Wizard 657
Table Names over 32 Characters Are Truncated in the New Table Wizard 657
Teradata Register Tables Wizard Hangs Unless a User ID and Password Can Be Supplied 657
Unrestricted Users Cannot Perform Tasks That Require Logon Credentials from the Metadata Server 658

Usage Notes for the View Data Window 658
About Usage Notes for the View Data Window 658
Cannot Display Tables in an HDFS Library or a LASR Library 658
Cannot View Cubes in SAS Data Integration Studio 658
Cannot View Tables in Libraries That Are Not Defined in a Current Repository 658
Close the View Data Window to Unlock a SAS Table 659
Default Parameter Values Are Used When Parameterized Tables Are Viewed . . 659
Error When Viewing SAP R/3 Tables 659
Libraries for Any User-Defined Formats Must Be Available 659
Permanent Data Set Formats Are Unaffected by the Apply Metadata Formats Option in the View Data Window 660
Setting Multiple Column Heading Label Options 660
Tables Might Not Be Editable Due to a Referential Integrity Issue 660
Table Options Will Be Ignored When You Create a Table with View Data 661
Using Columns with Leading or Trailing Spaces 661
View Data Queries Construct SELECT Statements 661
Viewing DBMS Tables Immediately After a Job Executes 661
Viewing Formatted Data in Fixed-Width External Files 662
Viewing Tables in a SAS WORK Library 662

Viewing Teradata Tables 662

Usage Notes for Iterative Jobs 662
About Usage Notes for Iterative Jobs 662
Iterative Jobs and Parameterized Jobs Behave as Completely Separate Jobs 662
Iterative Processing Places Limits on the Number of Nesting Levels and Jobs . . . 663
Parameter Values That Include Special Characters Require Special Handling . . . 663
Control Table Jobs Display an Out of Order Warning 664

Prerequisites for Running a Job When a DataFlux Server Is Used for Authentication 664
Overview 664
Verify How Users Are Authenticated on the DataFlux Data Management Server 664
Deploy the Job, Service, or Profile 664
Register a User on the DataFlux Authentication Server 665
Grant Privileges on the DataFlux Data Management Server 666
Update or Add a Data Management Server Definition in SAS Management Console 668
Update User Accounts in SAS Management Console 668
How the Connections Work 669
Next Tasks 669

Usage Notes for Loaders 669
Cloudera Impala: Usage Notes for Loaders 669
DBMS Notes for the SCD Type 2 Loader Transformation 670
MINMAXVARLIST Options Are Supported by Both SPD Server Loaders and SPD Server Tables 670
Setting the Autocommit Option in Table Loader Scenarios with a Vertica DBMS Target 670

General Usage Notes

Avoid Double Quotation Marks in DBMS Table and Column Names

SAS Data Integration Studio cannot successfully generate code for a job that includes a DBMS table in which the double quotation mark is used in the table name or the column names.

Concurrent Queries to Teradata Tables Can Fail

You should use the **CONNECTION=GLOBAL** LIBNAME option whenever you have difficulty running concurrent queries to Teradata tables using the Teradata Access Engine. This setting is recommended when these queries fail. A race condition can occur when there are two concurrent queries to a table that uses a LIBNAME with a connection that is not global. This global connection can prevent a disconnect at the end of the SQL procedure.

Create as View Option Works Only When It Is Possible to Create a View

In the process flow for a job, you can display the properties window for an output table, click the **Physical Storage** tab, and select the **Create as view** option. If this option is

selected, the table is created as a view, if that is possible. If it is not possible to create a view, a physical table is created even when the **Create as view** option is selected. For example, the Extract transformation generates implicit pass-through code. If the format of the output table is a DBMS other than SAS, the creation of a view might not be permitted in the target DBMS. Deselect this option to specify that the table should be created as physical table instead of a view.

Data Transfer Does Not Work for DBMS Tables with Special Characters in Table Names

If you create a process flow in a SAS Data Integration Studio job that transfers a DBMS table from one computer to another and the table name includes special characters, the transfer fails. Neither implicit nor explicit data transfer works in this case.

DBMS-Specific Functions Work Only with Explicit Pass-Through

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. When you are specifying expressions for columns in DBMS tables, the **Functions** tab of the Expression Builder might display a folder that is named after the DBMS, such as **Teradata**. Keep in mind that functions unique to a DBMS resolve only in the context of explicit pass-through processing. Otherwise, you will receive an error when the job is executed. One example of explicit pass-through processing would be in the context of the SQL Join transformation when the explicit pass-through option turned on.

Do Not Include the << and >> Signs in the Column Descriptions of a Table That Could Be Used in an Information Map

You can enter labels for columns in a SAS data set. These labels can be easier for users to manipulate and remember than the original column names. When the SAS data set is registered in a metadata repository, the labels are translated into the **Description** field for the column.

If you are using these SAS data sets to build information maps, then the column descriptions must not contain double less-than signs (<<) or double greater-than signs (>>). In addition, no description should end in a single greater-than sign (>). If the column description contains these characters, then these columns are not directly usable in expressions in an information map. You can use SAS Management Console or SAS Data Integration Studio to fix this problem by changing the description of the column.

Do Not Use MLE Library Tables as Targets in SAS Data Integration Studio Jobs

Tables can be stored in a library that was assigned or pre-assigned by using the metadata LIBNAME engine (MLE). Do not specify MLE Library table as a target in a SAS Data Integration Studio job. If you use an MLE library table as a target in a job, executing the job could have an unexpected impact on the job's metadata.

Enhanced Validation for Generated Code Can Be Turned Off

When SAS Data Integration Studio generates code for a job, it checks for the following conditions, among others:

- No column mappings are defined for a transformation.
- The number of source columns is not equal to the number of target columns in a transformation.

Previous releases of SAS Data Integration Studio did not check for these two conditions.

In most cases, if no column mappings are defined for a transformation, code generation fails with an error. This rule does not apply to the User-Written Code transformation and the Mining Results transformation. It also does not apply to any transformation for which code is not generated (the transformation is turned off or it retrieves user-written code). If you need a transformation that does not define mappings, use a User-Written Code transformation or specify user-written code on the **Process** tab for a transformation.

If the number of source columns is not equal to the number of target columns in a transformation, a warning message is displayed in the SAS log for the job. You can take action, if appropriate.

Perform the following steps to temporarily turn off the new validations until the job can be fixed:

1. Exit SAS Data Integration Studio. (You must close SAS Data Integration Studio before attempting to edit the file in the next step.)
2. Locate the file named **wadef.txt**.
3. Edit this file in a simple text editor. Append the following line to the end of the file: `913Behavior=Y`
4. Restart SAS Data Integration Studio. Most of the enhanced code generation error detections are now disabled.

After you have corrected the job to take advantage of the updated error checking, you can remove the line that turns off the error detection. Remember to close SAS Data Integration Studio before attempting to remove the line from the wadef.txt file.

Note: Turning off the enhanced error checking should be a temporary change only.

You should modify existing jobs to take advantage of the enhanced error detection at your earliest opportunity. Future releases of SAS Data Integration Studio might not enable you to turn off error detection for generated code.

External File Wizard: Cannot Sort Displayed Data by Column

The External File wizards have a Column Definition page. The bottom half of that page has a **Data** tab that displays data from the external file, if you have specified enough information about the file. When viewing data in the **Data** tab, you cannot click a column heading and sort the displayed data by the values in that column. The ability to sort the columns is currently not supported.

Jobs with Implicit Data Transfers between Different Hosts Fail

Do not use a loader transformation to specify an input table from one type of host (such as a PC) and an output table on a different type of host (such as MVS). This job flow will fail. To enable a transfer between different hosts, you must insert a Data Transfer transformation before the loader. This step ensures that the data is already in the target format before it is loaded into the target table.

Limit Line Lengths in the Expression Builder to 128 Characters

Limit line lengths in the Expression Builder to 128 characters to avoid problems caused by server line-length restrictions.

Maximum Integer Value for a Field in SAS Data Integration Studio

In SAS Data Integration Studio, for an integer column with the SAS format w., the maximum allowable value is 2,147,483,646 (Integer.MAX_VALUE - 1). A double (floating point) column with the SAS format BESTw. does not have this limitation.

Microsoft Queue Writer Transformation Does Not Transform Missing Date Values

The Microsoft Queue Writer transformation does not transform missing date values to some other value. If missing date values are encountered, then an error occurs. Tables that are transmitted on the Microsoft MQ messaging system should be handled in accordance with the following recommendations:

1. Use the Data Validation transformation to clean up the missing date values before you send any tables via a Microsoft queue writer.
2. Send tables in transactions on transactional queues. If a non-transactional queue is used, then all the rows of the table are sent until a row with any error (such as a missing date value) is encountered. At this point, the error prevents any further rows of the table being sent.

Migrating from SAS/Warehouse Administrator to SAS Data Integration Studio

For information about migrating from SAS/Warehouse Administrator to SAS Data Integration Studio, see Migration: Converting from SAS/Warehouse Administrator to SAS Data Integration Studio, which is available at the following website: **http://support.sas.com/rnd/migration/planning/software/etlstatement.html**.

Out-Of-Date Generated Transformations Are Updated Automatically When Included in Jobs That Are Deployed or Redeployed

If you include an out-of-date generated transformation in a job that is deployed or redeployed, SAS Data Integration Studio automatically updates the transformation in the deployed job. The updated transformation code is generated instead of the original code, so the results of the job can be affected.

SAS Data Integration Studio Jobs Cannot Output HP Neoview Tables That Have Keys

The HP Neoview DBMS does not support constraints (keys) on tables. Accordingly, SAS Data Integration Studio jobs cannot output HP Neoview tables that have keys.

SAS Metadata Bridges

SAS Data Integration Studio enables you to import and export metadata in formats that are supported by a SAS Metadata Bridge, such as Common Warehouse Metamodel (CWM) format. Make sure that you have the appropriate bridge. Ask your SAS representative for details.

Saving Metadata Changes in the Corresponding Physical Target

When you update a SAS Data Integration Studio job that has produced output tables at least once, any changes that you make to the column metadata for its tables are not reflected in the physical tables immediately. The column changes are not seen until you select **Drop Target** on the **Load Technique** tab of the Loader transformation for the tables, and you successfully execute the job again. **Drop Target** is not selected by default.

Saving Temporary Output Tables to a Permanent Location

Most transformations in a SAS Data Integration Studio job send their output to a temporary output table. By default, temporary output tables are stored in the WORK library. Perform the following steps to change the library where the temporary output table is stored for a transformation:

1. Right-click the icon for the temporary output table in a process flow (green circle). Select **Properties** from the pop-up menu.
2. Click the **Physical Storage** tab.
3. Change the library and other physical storage information as desired.

Sign-on Scripts for SAS/CONNECT Servers

SAS Data Integration Studio uses a SAS/CONNECT server to submit generated SAS code to computers that are remote from the default SAS Application Server. A SAS/CONNECT server can also be used for interactive access to remote libraries.

For SAS Data Integration Studio to generate the appropriate code for scripted sign on to a SAS/CONNECT server, you must specify a valid user ID and password in the sign-on script. The sign-on script is specified in the metadata for the SAS/CONNECT server in SAS Management Console.

SQL Join Transformations: Adding Multiple Sources with Primary Keys to an SQL Join Transformation Job

You can add more than one source table with a primary key to a job that includes the SQL Join transformation. However, only the first primary key match is shown in the SQL Join query. The auto-join feature of SQL Join assumes that primary keys must have foreign key relationships with the other tables participating in the join before auto-join finds them.

If two tables have different primary keys, and you want both keys to participate in the join, and you want auto-join to find both of them for you, you must make foreign key associations to represent the relationships between the two tables. Then, you do not have

to enter one key manually. This feature ensures that the key relationships found for the tables are consistent.

SQL Merge Transformations: Input and Output Issues

The SQL Merge transformation cannot accept a temporary output table as a source table. You must use a table from one of the supported database management systems. In addition, you cannot use the same table as both the input and output table for the SQL merge transformation.

SQL Merge Transformations: Use the SQLNCLI10 Driver to Access SQL Server 2008

When an SQL Merge transformation writes to a table in SQL Server 2008 format, use the SQLNCLI10 driver (SQL Server Native Client 10.0), not the older SQLOLEDB driver, to access SQL Server 2008.

SQL Update and SQL Merge Transformations: Manual Updates Are Required If Subqueries Are Used

The SQL Update transformation and the SQL Merge transformation do not support table aliases on the target table for the transformation. However, the Subquery interface for these transformations generates a table alias (‘t’) in the code for the subquery. To specify a subquery for a table that is the target of an SQL Update or SQL Merge transformation, for databases other than Oracle, edit the generated code as follows:

1. Add a subquery as usual to an SQL Update transformation or an SQL Merge transformation in a job. For example, open the properties window for an SQL Merge transformation. Click the **Source** tab. Select **Subquery** in the **Source** control to display the Subquery Builder. Then, click the **Filter and Sort** tab to specify a filter for the subquery.
2. Open the properties window for the transformation.
3. On the **Code** tab, select **All user written** from the **Code Generation Mode** control.
4. Find and delete the table alias 't' after `libname.tablename` in the UPDATE statement.
5. Click **OK** to save your changes.

If you want to specify a subquery for an Oracle table that is the target of an SQL Update or SQL Merge transformation, contact SAS Technical Support.

Synchpoint Option on WebSphere Queue Reader Does Not Prevent All Data Commits to Target When an Error Occurs

If the synchpoint option in the Advanced Setting window for the WebSphere Queue Reader transformation is set to **Yes**, you might expect that any error while reading messages from the queue would prevent any data from being written from the queue to the target. Instead, all of the messages that precede the message where the error occurred are written to the target and the remaining messages are not.

For example, if an error occurs on the fifth message in a ten-message queue, messages one to four are written to the target, and messages five to ten are not written. In addition, an **Ended with errors** status is displayed for the job, and an error message is added to

the log. Finally, all of the messages remain in the queue when an error occurs. If no errors are present, all of these messages are removed.

Transformations That Use PROC SQL Rename SAS Reserved Words Unless Case Sensitive Names and Special Characters Are Recognized

Any transformations that generate code that uses PROC SQL rename SAS reserved words without issuing a warning. (These reserved words are sometimes used as column names.) However, you can change this behavior by selecting the **Enable case-sensitive DBMS object names** check box and the **Enable special characters within DBMS object names** check box on the **Physical Storage** tab. Then, the SAS reserved words are treated as n literals and are not renamed.

Unrestricted Users Cannot Perform Tasks That Require Logins from the Metadata Server

An unrestricted user is one of the administrative users that can be defined for a SAS Metadata Server. If SAS Data Integration Studio and related software have been configured with the SAS Deployment wizard, a default unrestricted user called **sasadm** is created .

An unrestricted user such as **sasadm** cannot access other servers by retrieving logins from the metadata server. For example, you cannot log on to SAS Data Integration Studio as an unrestricted user and access the servers that are required by the Register Tables wizards and the New Table wizard. It also means that an unrestricted user cannot use the metadata Export wizard to include or replace physical tables in a DBMS.

For details about the unrestricted user, see the *SAS Intelligence Platform: Security Administration Guide*.

Update Table Metadata Cannot Be Used for Some Tables

You can use the Update Table Metadata feature to update DBMS table names and column names that have special characters. However, you cannot use it for SAS table names and column names with special characters. A special character is any character that is not an underscore, a letter of the alphabet, or a numeric digit (such as 0 through 9). You also cannot use this feature for PC format files, such as Microsoft Excel, or for parameterized tables.

Update Table Metadata on z/OS Platforms

The Update Table Metadata feature updates table metadata so that it matches the corresponding physical table. However, if the physical table resides on a z/OS platform, the update might fail for large tables. A z/OS limit on the number of characters in a single line causes this problem.

Updates Might Be Needed to the Names of Primary Keys for Oracle Tables

When SAS Data Integration Studio creates PRIMARY KEY definitions on Oracle tables, it uses the ADD PRIMARY KEY statement. This statement does not support the naming of the Primary Key. Accordingly, the key gets a generic name generated by Oracle. This happens even if you specify a name that you want to use for the key in the

properties window for the table. The result is that the error message that is returned when a constraint is violated might be hard to understand.

If this behavior becomes a problem for you, update the transformation that specifies the primary key for the table. Switch Code Generation from Automatic to User-Written Body, and then replace the existing ADD PRIMARY KEY syntax with the ADD CONSTRAINT name PRIMARY KEY syntax.

Very Large Tables or Files in WebSphere Message Queues Can Cause Memory Overruns

When very large tables or files of 500,000 rows or more are run in a job that includes a WebSphere message queue, memory overruns can occur. These overruns occur because memory handles are not freed during WebSphere MQ calls. Instead, the memory handles are freed only when the DATA step ends. This delay in freeing the memory means that a potential exists to run the server out of memory when you write very large files to a queue. You can correct the problem by splitting the writes and reads up into smaller sets of data. You can also modify the generated code to free up memory by adding code to the generated code for the SAS job.

The following lines of SAS code must be added in order to free handles when you write records from a table to a queue. They must also be added when you read records into a table from the messages on a queue. Add the following lines immediately before the line that consists of the label **`etls_mqexit:`** in the generated SAS code:

```
if etls_hmd ^=0 then
do;
/* Free message descriptor handle */
CALL MQFREE(etls_hmd);
end;
if etls_hmap ^=0 then
do;
/* Free map descriptor handle */
CALL MQFREE(etls_hmap);
end;
if etls_hdata ^=0 then
do;
/* Free Data handle */
CALL MQFREE(etls_hdata);
end;
```

When reading messages from a queue to a table, the SAS code to free up memory must be moved if it is added. Move the following mqmap call code to a position immediately after the line **`do i=1 to .....;`**:

```
call mqmap(etls_hmap, etls_qrc, etls_desc0, ....);
if etls_qrc ^= 0 then
do;
%rcSetDS(8000);
etls_qmessage = sysmsg();
put "ERROR: MQMAP: failed with reason code: " etls_qmessage;
goto etls_mqexit;
end;
```

Usage Notes for Register Tables Wizards and the New Table Wizard

Access to Data on z/OS Platforms

Data on a z/OS platform must be stored in a UNIX System Services (USS) directory rather than in an MVS bound library. For a USS directory, the physical name of the library is the same as the directory path. For more information, see "LIBNAME Statement: z/OS" in the *SAS Companion for z/OS*.

Access to Tables Using ODBC DB2 z/OS Pass-through

To use the pass-through facility for ODBC DB2 z/OS to access tables, you must configure the password and user ID. Because the DB2 z/OS pass-through does not support the PASSWORD= and USER= options, you must configure these options on the ODBC DB2/zOS source using the ODBC Administrator.

Case and Special Characters in DBMS Names

SAS Data Integration Studio cannot access DBMS tables with case-sensitive names or with special characters in names unless all of the following name options are specified:

- in the metadata for the database library that is used to access the table:
  - set **Preserve DBMS table names** to YES
  - set **Preserve column names as in the DBMS** to YES
- in the metadata for the table itself:
  - select **Enable case-sensitive DBMS object names**
  - select **Enable special characters within DBMS object names**

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 following options in the metadata for the SAS table:

- select **Enable case-sensitive DBMS object names**
- select **Enable special characters within DBMS object names**

Double-byte character set (DBCS) column names are supported in this way, for example.

Control Whether SAS Formats and Informats Are Automatically Applied to Table Columns

You can control whether SAS formats and informats are automatically applied to table columns when you register tables or when code is generated for tables. For example, the

Library Information tab in the Register Tables wizard has a new check box: **Include formats and informats in column definitions**. If you clear this check box, formats and informats will not be registered in metadata for the columns when the table information is created. Three options control the use of formats and informats in generated code. To control the use of formats and informats globally, select **Tools** ⇨ **Options** ⇨ **Job Editor Tab**, and then set the format or informat option in the **Automatic Settings** area. To control the use of formats and informats in a job, open the properties window for the job. Then click the **Options** tab, and then set the format or informat option on the **General** pane. To control the use of formats and informats in a transformation, open the properties window for the transformation. Then click the **Options** tab, and then set the format or informat option on the **Advanced Options** pane.

Generic Register Tables Wizard: When to Use

In general, a Register Tables wizard for a specific type of data generates more useful metadata than the **Generic** Register Tables wizard. Use the **Generic** Register Tables wizard only when a Register Tables wizard for a specific type of data is not available.

Importing Foreign Keys for 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.

Importing Foreign Keys for SAS Tables

The Register Tables wizard for data in SAS format imports metadata for SAS tables, including the metadata for foreign keys. To successfully import the metadata for foreign keys in SAS tables, the following conditions must be met:

- Primary keys and foreign keys must have unique names across all SAS tables in all SAS libraries from which metadata is imported.
- In the Define Tables window in the Register Tables wizard, select the primary key table and all related foreign key tables. Otherwise, the metadata is incomplete. (If the metadata is incomplete, then all registrations must be deleted and the complete set of related tables would need to be imported again to get the complete set of metadata objects.)

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.

Importing Keys and Indexes from SAS/SHARE Libraries

You can import keys and indexes for SAS tables in a SAS/SHARE library but not for DBMS tables in a SAS/SHARE library.

LIBNAME Options Required for Support of Case and Special Characters in the Names for Keys and Indexes

The Register Tables wizard can register database tables, including the metadata for keys and indexes. However, when you select a database library in the Register Tables wizard,

to preserve case-sensitive names in table keys and indexes, and to preserve names with special characters in table keys and indexes, you must specify the **Preserve DBMS table names** option and the **Preserve column names as in the DBMS** option for the selected library.

In a Register Tables Wizard, Limit Excel Connections to a Single User Name

Dedicate a single user name to use when you register Excel data with the SAS/ACCESS Excel engine and the SAS Data Integration Studio Register Tables wizard for ODBC or OLE DB. This restriction is necessary because of connection and LIBNAME errors that are generated when you register Excel data. Close and restart SAS Data Integration Studio and try to register Excel data using a different user name. The connection error states that the connection has been opened exclusively by another user or that you need permission to view the data. In either case, a connection to the Excel data cannot be established under the second user name. At this point, you must either use the original user name or restart the application server.

Limitations of Register Tables Wizards for MySQL and Informix

Primary keys, foreign keys, and index cannot be registered for Informix, ODBC Informix, and OLE DB Informix tables.

To set preserve_tab_names, open the Advanced Options window for an Informix library either from the New Library Wizard or from the properties window for the library. Select the **Output** tab and select a value of YES for the **Preserve column names as in the DBMS** field. To set preserve_col_names, from the Advanced Options window, select the **Input/Output** tab and select a value of YES for the **Preserve DBMS table names** field.

Metadata for a Library and Its Tables Must Be Stored in the Same Metadata Repository

The metadata for a library and the metadata for the tables in the library must be stored in the same metadata repository. Other configurations are not supported in this release.

Microsoft Windows Administrative Rights Required to Connect to OLE DB Data Sources

Users must have Microsoft Windows administrative rights on the server that contains the data before they can connect to any OLE DB data source. If a user does not have these administrative rights, an attempt to connect to an OLE DB data source generates the following text:

```
ERROR: Error trying to establish connection:
Unable to load OLE DB conversion library
ERROR: Error in the LIBNAME statement.
```

This error message is displayed even when the user has a valid Microsoft Windows user ID and password to log on to the server. After the administrative rights have been granted, the user can connect to OLE DB data sources without generating the error.

ODBC Informix Library: Preserving Case in Table Names

Perform the following steps to preserve the case of table names when using an ODBC Informix library with a Register Tables wizard:

1. From the SAS Data Integration Studio desktop, right-clock the folder in the Folders tree where the metadata for the ODBC table should be saved. Then select **Register Tables**. The wizard selection window is displayed.
2. Open the **ODBC Sources** folder in the wizard selection window.
3. In the **ODBC Sources** folder, select **ODBC Informix**. The ODBC Informix Register Tables wizard is displayed. The first window enables you to select an ODBC Informix library.
4. Select the appropriate ODBC Informix library and then click **Edit**. A library properties window is displayed.
5. On the library properties window, click the **Options** tab.
6. On the **Options** tab, click **Advanced Options**. The Advanced Options window is displayed.
7. In the Advanced Options window, select the **Output** tab.
8. On the **Output** tab, select **Yes** in the **Preserve column names, as in the DBMS** field.
9. Enter the following in the **Options used in DBMS CREATE TABLE** field:

   ```
   QUOTE_CHAR=
   ```

10. Click the **Input/Output** tab.
11. Select **Yes** in the **Preserve DBMS table names** field.
12. Click **OK** to save your changes.

Registering SAS/SHARE Tables

The SAS/SHARE Register Tables wizard enables you to select a library that contains the tables to be registered. Be sure to select a SAS/SHARE client library, not a SAS/SHARE server library.

Registering a Sybase Table with a Clustered Index

If you use the Sybase source designer wizard to register a Sybase table with a clustered index, the wizard does not register the clustered index by default. To register a clustered index, select the **Clustered values** check box on the **Index** tab of the wizard.

Separate Logon Credentials for Each Authentication Domain for Database Servers

Administrators define the metadata for users and groups as part of the setup tasks for a data warehousing project. The logon metadata for each user and group includes an authentication domain.

Each user (or a group to which the user belongs) must have a user ID and password for the authentication domain that is associated with the relevant database server definition.

That user ID and password must correspond to an account that has been established with the database. Otherwise, the user cannot read any existing tables in the relational database, and the user cannot use the Register Tables wizard or the New Tables wizard to access tables in the relational database.

Accordingly, administrators must define separate logon credentials for each authentication domain that contains a database server that you need to access. For more information about defining logon metadata for users and groups, see the *SAS Intelligence Platform: Security Administration Guide*.

Setting Table Options in the New Table Wizard

The New Table wizard and the property windows for tables include a physical storage tab or window. This tab or window includes a **Table Options** button. Click that button to specify options for the current table. For details about options for SAS tables (data sets and views), see *SAS Language Reference: Dictionary*.

Table Names over 32 Characters Are Truncated in the New Table Wizard

You can create a table with a name that has more than 32 characters in the **New Table** wizard. However, the name of the table will be truncated to its first 32 characters. If another table that begins with the same first 32 characters is created in the **New Table** wizard in the same library, this new table name is also truncated to the same 32-character name. If the second table is used in a SAS Data Integration Studio job, then its data could overwrite the data in the physical table from the identically named first table.

Teradata Register Tables Wizard Hangs Unless a User ID and Password Can Be Supplied

The Register Tables wizard enable you to import metadata for one or more tables in a library. One of the first windows in the wizard enables you to select the library that contains the tables. When you select a library, a connection is made to a SAS Application Server. The server accesses the selected library and lists any tables that are associated with that library.

The Teradata Register Tables wizard cannot connect to a Teradata database library unless both of the following conditions are met:

- A Windows environment variable, GUILOGON, is defined and set to NO on the computer where SAS/ACCESS to Teradata is running. (This variable is typically set for the SAS Workspace Server component of the SAS Application Server that is used to access the Teradata database.) For details about how to define Windows environment variables, see the appropriate Windows documentation.
- Valid logon credentials are supplied to the Teradata database server.

There are two main ways to supply valid logon credentials to the Teradata database server:

- Add a default user ID and password to the metadata for the Teradata database library.
- Implement single sign-on (SSO) for the Teradata database on Windows. For details about SSO, the database administrator should see the appropriate Teradata documentation.

Unrestricted Users Cannot Perform Tasks That Require Logon Credentials from the Metadata Server

An unrestricted user is one of the administrative users that can be defined for a SAS Metadata Server. If SAS Data Integration Studio and related software have been configured with the SAS Deployment wizard, a default unrestricted user called **sasadm** is created.

An unrestricted user such as **sasadm** cannot access other servers by retrieving logon credentials from the metadata server. For example, you cannot log on to SAS Data Integration Studio as an unrestricted user and access the servers that are required by the Register Tables wizard or the New Tables wizard. It also means that an unrestricted user cannot use the Metadata Export wizard to include or replace physical tables in a DBMS.

For details about the unrestricted user, see the *SAS Intelligence Platform: Security Administration Guide*.

Usage Notes for the View Data Window

About Usage Notes for the View Data Window

These notes apply to the features that are included in the View Data window. To access this window, right-click a table or external file and click **Open** in the pop-up menu.

Cannot Display Tables in an HDFS Library or a LASR Library

See "View Data Window Cannot Display Tables in an HDFS Library or a LASR Library" on page 609 .

Cannot View Cubes in SAS Data Integration Studio

You cannot view the contents of a cube in SAS Data Integration Studio. You can use Microsoft Excel or SAS Enterprise Guide to view the data in a cube. For details, see the documentation for SAS OLAP Server.

Cannot View Tables in Libraries That Are Not Defined in a Current Repository

The View Data window cannot display data in a table or view unless the library for that table or view is defined in a current metadata repository.

For example, you cannot view the data in a SAS table if you change the SAS library to on the **Physical Storage** tab in the properties window for the table. To successfully view SAS table data in a work library, you must first reassign that library to an existing metadata library.

As another example, if the **Temp** option is checked for an SAS SPD Server library, you cannot use the View Data feature to display tables in the library.

The properties window for a SAS SPD Server library includes an **Options** tab. On the **Options** tab, there is an **Advanced Options** button. If you click the **Advanced Options** button and then select the **Server Connection Information** tab, you can specify **YES** or

NO in the **Temp** field. The **Temp** field specifies whether a temporary LIBNAME domain is created for the library.

If you specify **YES** in the **Temp** field, any data objects, catalogs, or utility files created in the library are deleted after the job is executed. Accordingly, you cannot use the view data feature in SAS Data Integration Studio to view tables in the library.

Close the View Data Window to Unlock a SAS Table

Close the View Data window on a SAS table before running a job that updates data in that table. Otherwise, the job fails, with the following error message in the SAS log:

```
A lock is not available for table, lock held by another process.
```

If this error message appears, close the View Data window and run the job again. SAS releases the lock on the table, and the job can complete successfully. This problem affects only SAS tables.

Default Parameter Values Are Used When Parameterized Tables Are Viewed

Macro variable parameters can be used in the **SASTableName** field of tables. The View Data function generates code to resolve these macro variable parameters by using the default values of the parameters. You cannot view these tables unless the View Data function can recognize the values entered in their **SASTableName** fields. Therefore, you must use a specific syntax that joins the table name to the parameter value. For example, a table named PARAMTABLE that contains a parameter named &STATE would be entered as PARAMTABLE_&STATE in the Properties window for the table. Enter the new table name into the **SASTableName** field on the **Advanced** tab.

The data displayed in the View Data window for these parameterized tables is based on the value entered in the **Default value** field when the parameter is created.

Error When Viewing SAP R/3 Tables

You might receive the Oracle error, ORA-1555, or a message similar to the SAS/ACCESS error noted below when using the View Data window to display information in an SAP R/3 table:

```
FREECPIC1 SQL error 1555 occurred when accessing table QALS
```

This error occurs because the snapshot is too old or the rollback segment is too small. When a query is submitted to Oracle, it takes a snapshot of the currently committed transactions and creates a consistent set of results in a rollback segment. If the data of a table is updated frequently and the query is a long-running query, there might not be enough information in the rollback segment to reconstruct the older data. That is why the error occurs.

To avoid this error, create more or larger rollback segments, schedule long-running queries when there are fewer concurrent transactions made, or obtain a shared lock on the table that you are querying.

Libraries for Any User-Defined Formats Must Be Available

The View Data window cannot display data with user-defined formats unless the format library is available to the SAS Application Server that is used to display the data.

Permanent Data Set Formats Are Unaffected by the Apply Metadata Formats Option in the View Data Window

When permanent formats have been applied to data in data sets, the permanent formats are displayed even when the View Data window has been set to show unformatted data. This is true whether the setting is made on the **View Data** tab in the global Options window in SAS Data Integration Studio or on the View Data window toolbar. The settings that toggle between the display of formatted and unformatted data apply only to formats that are defined in the metadata.

Setting Multiple Column Heading Label Options

The View Data window supports multiple column heading label options. The labels are displayed on the View Data toolbar in this order:

1. Show Column Name
2. Show Description
3. Show Metadata Name

When you select more than one of these options, the column labels are displayed in a fixed sequence. The labels are sequenced as follows:

- **Show Column Name** always comes before either **Show Metadata Name** or **Show Description**.
- **Show Metadata Name** always comes after **Show Column Name** and before **Show Description**.
- **Show Description** always comes after either **Show Column Name** or **Show Metadata Name**.

Therefore, the following column label combinations are possible:

- **Show Column Name** (**Show Metadata Name: Show Description**)
- **Show Column Name** (**Show Metadata Name**)
- **Show Column Name** (**Show Description**)
- **Show Metadata Name** (**Show Description**)

If none of the column heading label options are selected, the physical column name is displayed. Also, if you select the **Show Metadata Name** option or the **Show Description** option for a column that is missing a metadata name or a description, the physical column name is displayed by default.

Tables Might Not Be Editable Due to a Referential Integrity Issue

When primary keys and foreign keys are created, and these keys establish referential relationships between tables, the ability to edit these tables within the View Data window can be disabled under certain circumstances. You can open a table that has referential relationships to other tables for editing if no other tables within the relationship are open for editing. However, you cannot edit any of the other tables within the referential relationship. To clear this editing conflict between the tables, you must close all tables and then you can edit the first table that you reopen.

Table Options Will Be Ignored When You Create a Table with View Data

When you register metadata for a table in the New Table wizard, you can set its options by performing the following steps:

1. Click the **Table Options** button on the **Physical Storage** screen of the New Table wizard.
2. Enable one or more of the available options. Note that these options are also available for a completed table metadata object. The **Table Options** button is also found on the **Physical Storage** tab of the Properties window for the table.

Any table options added before the table is created are ignored, even if they are applied in the metadata. In addition, if these table options are incompatible with the DATA step that View Data uses to build the physical table, the query might fail. In that case, the table is not created, and the query ends with an error. After you have successfully created the table, you can set any options that you need.

Using Columns with Leading or Trailing Spaces

You cannot use View Data on tables whose column names have leading or trailing spaces.

View Data Queries Construct SELECT Statements

The View Data window constructs a SELECT query statement 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, the view data function 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 is displayed. The dialog box gives you the option of ignoring the column metadata that has been registered for the data store. You can use 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.

Viewing DBMS Tables Immediately After a Job Executes

Some database management systems do not commit changes as soon as they are requested. Accordingly, if a SAS Data Integration Studio job updates a table in a DBMS and you try to verify the update by using the View Data feature, the changes might not show up immediately.

If you want SAS changes to a DBMS table to show up immediately, select **YES** in the **Whether to COMMIT immediately after a transaction** field in the metadata for the DBMS library that is used to access the DBMS table.

To select this option for a DBMS library, display the property window for the library, select **Options**, and then click the **Advanced Options** button. Click the **Input/Output** tab. In the **Whether to COMMIT immediately after a transaction** field, select **YES**, and then click **OK** to save your changes.

Viewing Formatted Data in Fixed-Width External Files

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

Viewing Tables in a SAS WORK Library

In order to use the View Data window to view physical tables attached to a SAS WORK library, the SAS WORK library must be registered, and the library must be assigned to a SAS Application Server.

Viewing Teradata Tables

To display a Teradata table in the View Data window, you must connect to the database as a user with SELECT permissions. Otherwise, you get an error message saying that there are no columns in the table.

Usage Notes for Iterative Jobs

About Usage Notes for Iterative Jobs

Iterative jobs are jobs with a control loop in which one or more processes are executed multiple times.

The following usage notes apply to iterative jobs:

- “Iterative Jobs and Parameterized Jobs Behave as Completely Separate Jobs” on page 662
- “Iterative Processing Places Limits on the Number of Nesting Levels and Jobs” on page 663
- “Parameter Values That Include Special Characters Require Special Handling” on page 663
- “Control Table Jobs Display an Out of Order Warning” on page 664

Iterative Jobs and Parameterized Jobs Behave as Completely Separate Jobs

Iterative jobs and parameterized jobs are treated as completely separate jobs, even when a parameterized job is nested within an iterative job. This fact has the following implications for check-out, delete, copy and paste, and import and export behaviors:

- Check-out: A parameterized job is not checked out when the iterative job containing it is checked out. Each job must be checked out individually.
- Delete: A parameterized job is not deleted when the iterative job containing it is deleted. Each job must be deleted individually.
- Copy and paste: A parameterized job is not copied or pasted just because the iterative job that contains it is copied or pasted. Instead, the new copy of the iterative job includes an association to the original parameterized job. If you need to preserve

the nested relationship between the iterative and the parameterized job, you must copy and paste both jobs. The easiest way to perform this task is to select one job, and then hold down the CTRL key and click the other job. Next, right-click and select **Copy** from the **Properties** menu. You can then paste the copy of the nested jobs.

- Import and export: A parameterized job is not imported or exported just because the iterative job that contains it is imported or exported. Instead, the new import or export of the iterative job includes an association to the original parameterized job. If you need to preserve the nested relationship between the iterative and the parameterized job, you must import or export both jobs.

Iterative Processing Places Limits on the Number of Nesting Levels and Jobs

At this time, the following nesting scenarios are supported for iterative processing and parallel processing:

```
One level of nesting:

Job A
    Job B

Two levels of nesting:

Job A
    Job B
        Job C
```

In addition, each looping construction can contain only one job. A looping construction consists of a Loop transformation and a Loop End transformation.

Parameter Values That Include Special Characters Require Special Handling

If table names, physical paths, or other attributes in a job are specified as parameters and those parameters resolve to literal values that contain characters other than letters, numeric digits (such as 0 through 9), or underscores, then code generation for the job might fail.

If code generation for a parameterized job fails, check the log for an error message that says:

NOTE: Special characters encountered; References may require: %unquote(&<name>);

If you see this message, use the ***%unquote(&<name>);*** syntax to resolve the parameter value correctly during execution of the generated code. For example, suppose that the metadata for an external file specified its physical path as the parameter &srcpath. When code is generated for the external file, the &srcpath parameter expands to a string that included slashes and the job fails.

To enable the job to run, open the properties window for the external file, go to the field where the &srcpath parameter is specified, and enter the following string:

%unquote(&srcpath);

After this change, the resolution of the parameter value for the generated code can handle special characters, and the path no longer causes an error.

Control Table Jobs Display an Out of Order Warning

If you use a Control Table job, a warning dialog box might be displayed. This warning indicates that the transformations are out of order even though the order is appropriate and the job runs properly with no error or additional warning messages. This warning message does not appear in the logs when the job is executed.

If you encounter this warning, dismiss the dialog box and ignore the warning.

Prerequisites for Running a Job When a DataFlux Server Is Used for Authentication

Overview

These prerequisites apply if you want to incorporate a DataFlux data job, process job, data service, or profile into the flow for a SAS Data Integration Studio job. The job, service, or profile must be deployed to a DataFlux Data Management Server. It is assumed that this server is secured with a DataFlux Authentication Server, as described in the next section.

Verify How Users Are Authenticated on the DataFlux Data Management Server

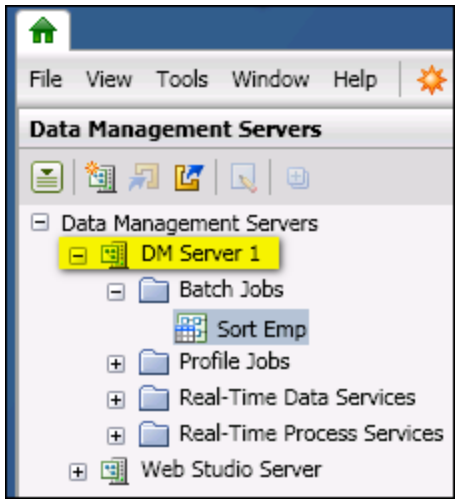
SAS Data Integration Studio can access DataFlux jobs, services, or profiles if they have been deployed to a DataFlux Data Management Server. In production environments, this server is usually secured. It can be secured with either a DataFlux Authentication Server or a SAS Metadata Server. SAS offerings that include SAS Data Integration Studio 4.8 and later typically use the SAS Metadata Server to authenticate users on a DataFlux Data Management Server. The administrator who maintains your data management environment should know which authentication method is being used to secure the server where the DataFlux jobs, services, or profiles have been deployed.

If authentication is handled by a DataFlux Authentication Server, then follow the steps in the next sections. If authentication is handled by a SAS Metadata Server, then see "Prerequisites for Running a DataFlux Job or Profile in a SAS Data Integration Studio Job" on page 338.

Deploy the Job, Service, or Profile

A DataFlux Data Management Studio user deploys jobs, services, or profiles to a DataFlux Data Management Server. He or she should ensure that the objects can be executed on the server and that they deliver the expected results. The next display shows a job, Sort Emp, that has been deployed to a server called DM Server 1.

Display A2.1 *Data Job Deployed to a DataFlux Data Management Server*

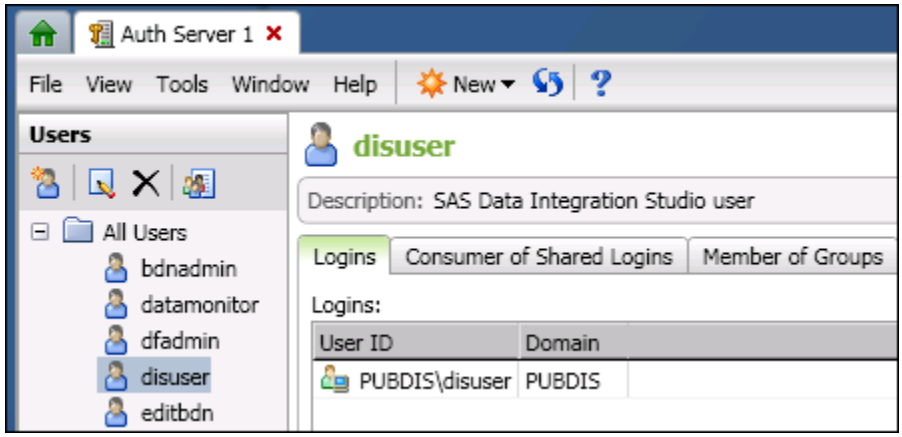


For information about deploying jobs, services, and profiles to a Data Management Studio, see the chapters for data jobs, process jobs, and profiles in the *DataFlux Data Management Studio User's Guide*.

Register a User on the DataFlux Authentication Server

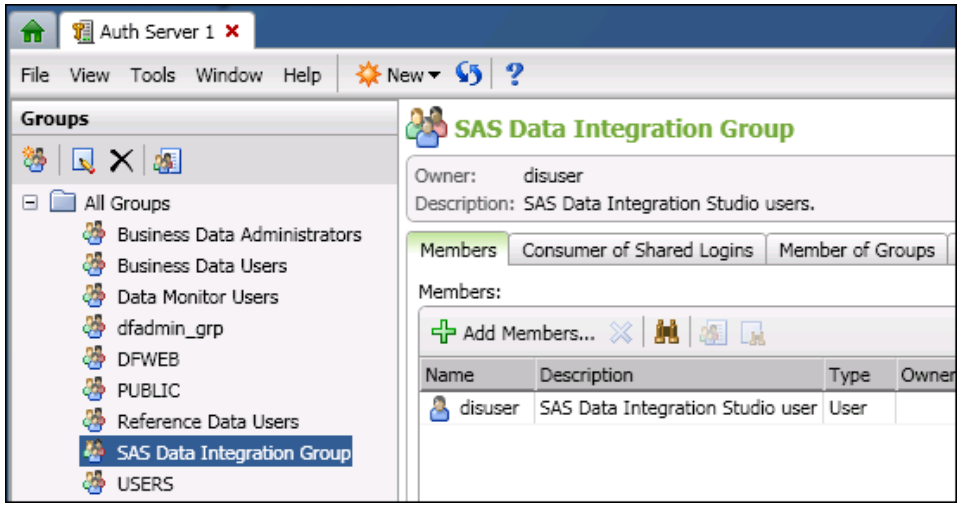
This task is performed on the Administration riser in DataFlux Data Management Studio. The administrator for the DataFlux Authentication Server adds a user account to that server. This user account must have credentials that can be authenticated by the DataFlux Authentication Server. Later, these login credentials are added to the SAS Data Integration Studio user's account in SAS Management Console. These login credentials enable the SAS Data Integration Studio user to list and execute the deployed objects on the DataFlux Data Management Server. The example account that is shown in the next display is for a user called disuser.

Display A2.2 *User Account on the DataFlux Authentication Server*



(Optional but recommended). On the DataFlux Authentication Server, create a group for SAS Data Integration Studio users. Add these users to that group. Having a group for SAS Data Integration Studio users make it easier to grant privileges to deployed objects. The example group that is shown in the next display is called SAS Data Integration Group. Note that disuser is a member of this group.

Display A2.3 *Group Account on the DataFlux Authentication Server*



For details about adding users and groups, see the *DataFlux Authentication Server User's Guide* and the *DataFlux Authentication Server Administrator's Guide*.

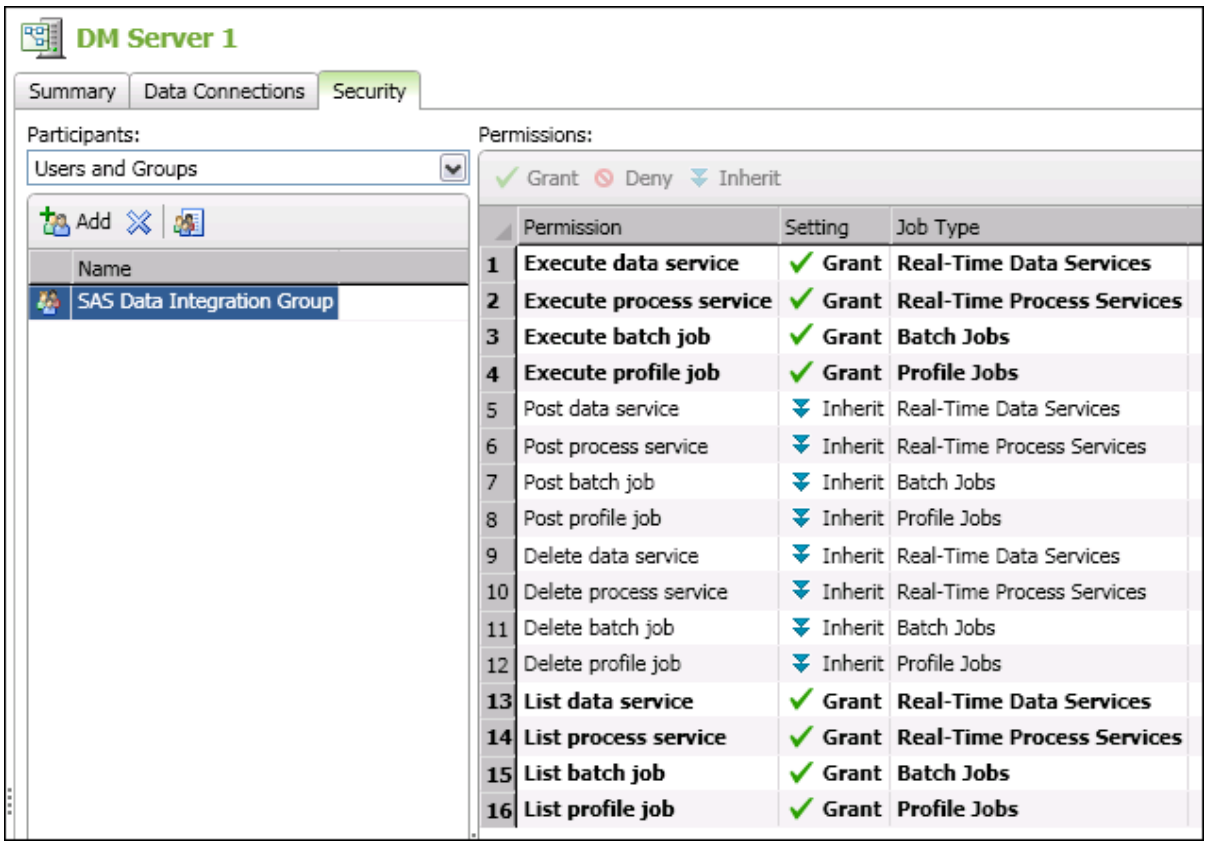
Grant Privileges on the DataFlux Data Management Server

This task is performed on the Data Management Servers riser in DataFlux Data Management Studio. It is performed by the administrator for the DataFlux Data Management Server where jobs, services, or profiles have been deployed. This administrator grants the appropriate users or group the general permission to list and execute deployed objects on the data management server. He or she also grants the appropriate user or group access to these specific objects.

Note: Both the List permission and the Execute permission must be granted to users or groups who execute jobs, services, or profiles on a DataFlux Data Management Server.

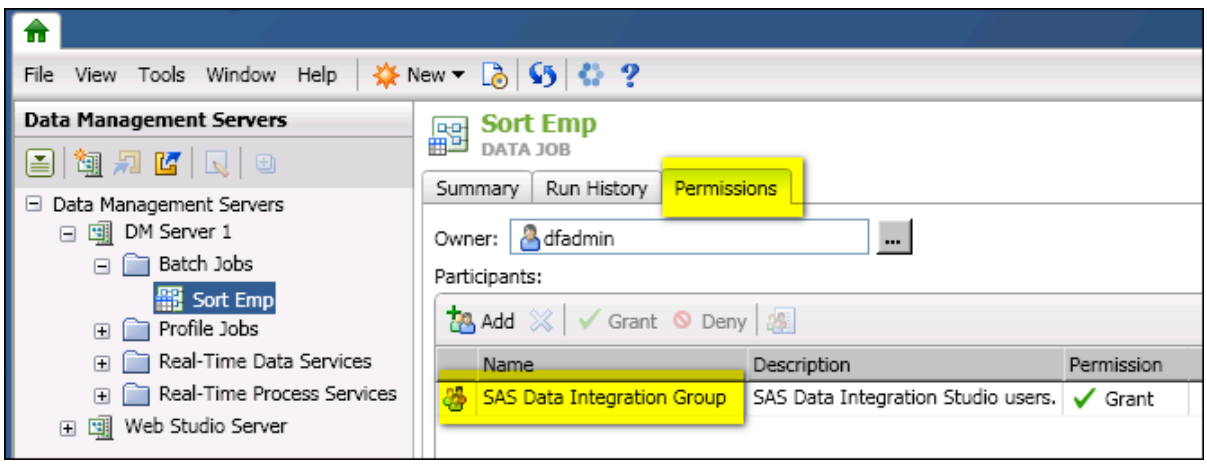
For example, you can grant List and Execute permissions to the SAS Data Integration Group, as shown in the next display.

Display A2.4 *Granting List and Execute Permissions for Jobs, Services, and Profiles*



Next, identify the individual jobs, services, or profiles on the server that SAS Data Integration Studio users should be able to execute. Grant the appropriate user or group access to these specific objects. For example, you can grant permissions so that the SAS Data Integration Group can access Sort Emp, as shown in the next display.

Display A2.5 *Granting Permission to Access Individual Jobs, Services, and Profiles*



For more information about these tasks, see the "Security Administration" chapter in the *DataFlux Data Management Server Administrator's Guide*.

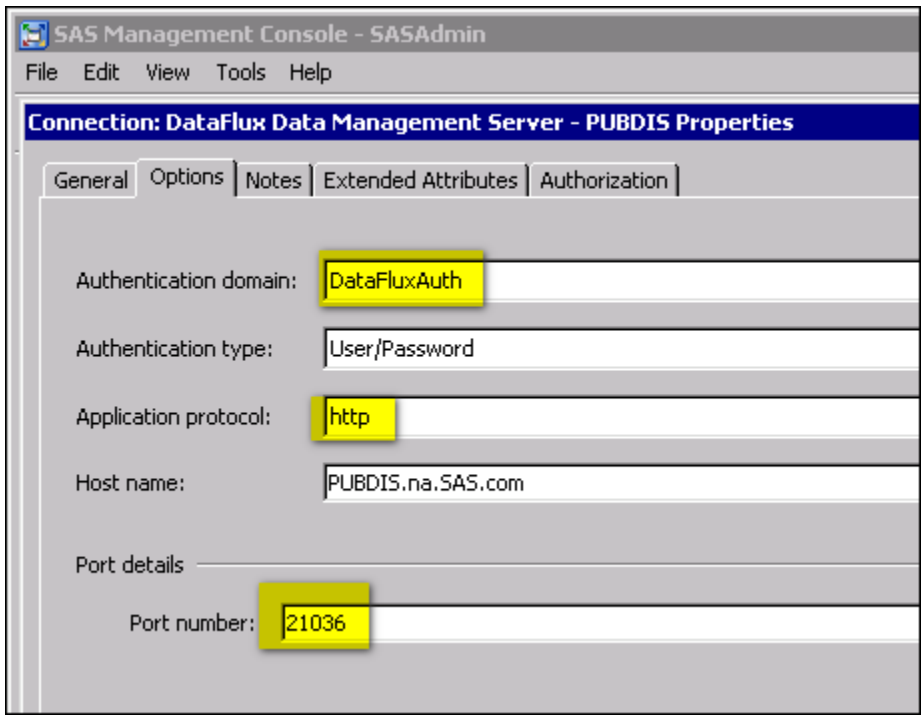
Update or Add a Data Management Server Definition in SAS Management Console

This task is performed in SAS Management Console. It is performed by the administrator who is in change of maintaining server definitions and other metadata for SAS Data Integration Studio. If the SAS Data Quality Server and the DataFlux Data Management Platform were installed on your site as part of a SAS Enterprise bundle, a server definition is automatically created in SAS Management Console for the DataFlux Data Management Server. An administrator should review this definition and check the following items:

- **Authentication domain**. This field should specify a SAS authentication domain for the DataFlux Data Management Server where the jobs, services, or profiles have been deployed. If this field specifies the default SAS domain (**DefaultAuth**), replace the default with a new domain for the DataFlux context, such as **DataFluxAuth**.
- **Other fields**. The values in the other fields should be appropriate for the DataFlux Data Management Server where objects have been deployed.

This definition enables SAS Data Integration Studio to connect to the DataFlux Data Management Server. An example definition is shown in the next display.

Display A2.6 *HTTP Server Definition for the DataFlux Data Management Server*



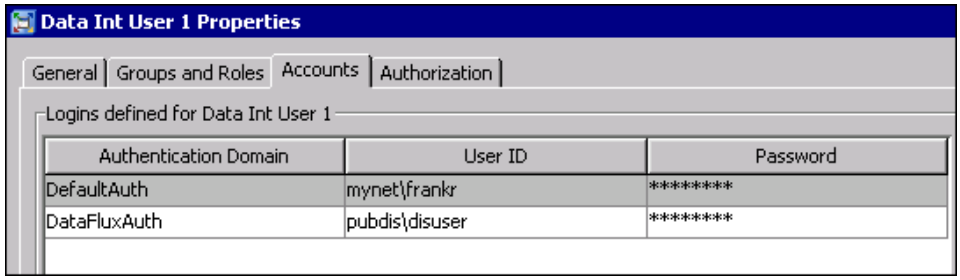
For more information about updating or adding server definitions, see the Help for the Server Manager in SAS Management Console. See also the SAS Data Integration Studio chapter of the *SAS Intelligence Platform: Desktop Administration Guide*.

Update User Accounts in SAS Management Console

This task is performed in SAS Management Console by the administrator who is in change of maintaining metadata for SAS Data Integration Studio. The administrator

updates the user account for the SAS Data Integration Studio user who must execute DataFlux Data Management Studio jobs and other deployed objects. Add an appropriate DataFlux Authentication Server user and login for this user, such as the disuser account that is described in "Register a User on the DataFlux Authentication Server" on page 665. Specify the SAS authentication domain from the HTTP Server definition above, such as the DataFluxAuth domain that is shown in the next display.

Display A2.7 *Add a DataFlux Authentication Server User and Login to a SAS Metadata Identity*



For more information about adding logins to a user definition, see the Help for the User Manager in SAS Management Console.

How the Connections Work

A user logs in to SAS Data Integration Studio as usual. If he or she tries to connect to the DataFlux Data Management Server, the HTTP Server definition for that server is accessed. Since this definition specifies the DataFluxAuth domain, the connection searches the user's account in SAS Management Console and finds the login for that domain, and the connection is made to the DataFlux Data Management Server.

The DataFlux Data Management Server checks for the appropriate permissions, and if the appropriate permissions have been granted to this user, the job, service, or profile is executed.

Next Tasks

After you have met the prerequisites above, you can do the following tasks:

- "Using a DataFlux Job or Profile in a SAS Data Integration Studio Job" on page 355
- "Using a DataFlux Data Service in a SAS Data Integration Studio Job" on page 351

Usage Notes for Loaders

Cloudera Impala: Usage Notes for Loaders

See "Usage Notes for Table Loader and Cloudera Impala Tables" on page 610 .

DBMS Notes for the SCD Type 2 Loader Transformation

In jobs that include the SCD Type 2 Loader and a Sybase target, or a Netezza target, SQL pass-through is not supported, even when you specify **Yes** for the Use SQL Passthru option.

In jobs that include the SCD Type 2 Loader and a Sybase target, the job runs only if you include the value **UNDOPOLICY=NONE** in the **SQL Options** field.

When the SCD Type 2 Loader Transformation is used to load Netezza or Neoview tables, the Autocommit option should be turned off in the DBMS library. To turn off this option, display the properties window for the library, select **Options** ⇨ **Advanced Options**, and then select the **Input/Output** tab. Set the **Whether to COMMIT immediately after a transaction** field to **No**.

MINMAXVARLIST Options Are Supported by Both SPD Server Loaders and SPD Server Tables

You can set the MINMAXVARLIST option for either an SPD Server Loader transformation or an SPD Server table. However, the scope of the option depends on the type of object that you select.

When you set the MINMAXVARLIST option for an SPD Server Loader transformation, the option is applied to any table that is loaded with the transformation. To set the option on a loader, open the property window for the SPD Server Loader transformation. Then, click the **Options** tab and enter **minmaxvarlist=xxx** in the **Advanced Data Table Options** field (where **xxx** denotes the range of variables that you want to specify).

When you set the MINMAXVARLIST option for an SPD Server table, the option is applied to the table in any job when the transformation that uses the table supports table options. To set the option on a table, open the properties window for the SPD Server Table. Then, click the **Physical Storage** tab and click **Table Options**. You can then enter **minmaxvarlist=xxx** in the **Additional Options** field (where **xxx** denotes the range of variables that you want to specify).

Setting the Autocommit Option in Table Loader Scenarios with a Vertica DBMS Target

When you use the Table Loader or SCD Type 2 loader transformations with a Vertica DBMS table target, you must set the AUTOCOMMIT option to **No**. However, you must set the AUTOCOMMIT option to **Yes** if the **Load Style** field on the **Load Technique** tab is set to **Update/Insert** and the **Matching Rows** field is set to either **Modify by Column(s)** or **Skip Matching Rows**.

If you receive the error message “ERROR: One or more values have been changed in the record to be updated”, then you must set the AUTOCOMMIT option to **No**.

Perform the following steps to set the AUTOCOMMIT option for a Vertica library:

1. Right-click the library in the Inventory pane and open its Properties window.
2. Click **Options**.
3. Click **Advanced Options**.
4. Click **Other Options**.
5. Enter **AUTOCOMMIT=YES** into the **Option(s) to be appended** field.

6. Click **OK** to save the option.

Appendix 3

Miscellaneous Transformations

Using a Business Rule Flow in a Job **675**
Overview 675
Problem 675
Solution 676
Tasks 676
Usage Notes 679

Creating a Table That Appends Two or More Source Tables **679**
Overview 679
Problem 679
Solution 680
Tasks 680

Creating a Publish to Archive Report from Table Data **682**
Overview 682
Problem 683
Solution 683
Tasks 683

Validating Product Data **689**
Overview 689
Problem 690
Solution 690
Tasks 690

Creating a Publish to Email Report from Table Data **693**
Overview 693
Problem 694
Solution 694
Tasks 694

Integrating a SAS Enterprise Miner Model with Existing SAS Data **700**
Overview 700
Problem 700
Solution 701
Tasks 701

Creating a Publish to Queue Report from Table Data **706**
Overview 706
Problem 706
Solution 706
Tasks 707

Extracting Data from a Source Table **712**
Overview 712

Problem . . . 712
Solution . . . 712
Tasks . . . 713

Creating Reports from Table Data . . . 715
Overview . . . 715
Problem . . . 715
Solution . . . 715
Tasks . . . 715

Create a Table That Ranks the Contents of a Source . . . 721
Overview . . . 721
Problem . . . 721
Solution . . . 721
Tasks . . . 722

Create Two Tables That Are Subsets of a Source . . . 724
Overview . . . 724
Problem . . . 724
Solution . . . 724
Tasks . . . 724

Moving Data Directly from One Machine to Another Machine . . . 728
Overview . . . 728
Problem . . . 728
Solution . . . 728
Tasks . . . 729

Creating Standardized Statistics from Table Data . . . 732
Overview . . . 732
Problem . . . 732
Solution . . . 732
Tasks . . . 733

Creating Transposed Data from Table Data . . . 736
Overview . . . 736
Problem . . . 736
Solution . . . 736
Tasks . . . 737

Converting a SAS or DBMS Table to an XML Table . . . 741
Overview . . . 741
Problem . . . 741
Solution . . . 741
Tasks . . . 742

Using ODS to Specify Output from the XML Writer . . . 746
Problem . . . 746
Solution . . . 746
Tasks . . . 747

Using SOAP to Access a Third-Party Web Service . . . 747
Problem . . . 747
Solution . . . 747
Verify Prerequisites . . . 747
Create a Job That Uses a SOAP Transformation to Access a Web Service . . . 748

Using REST to Access a Third-Party Web Service . . . 750
Problem . . . 750
Solution . . . 750
Verify Prerequisites . . . 750

Create a Job That Uses a REST Transformation to Access a Web Service 750

Generating Enterprise Decision Management Output 751
Overview 751
Problem 752
Solution 752
Tasks 753
Usage Notes 757

Running Conditional Processes 757
Overview 757
Problem 757
Solution 758
Tasks 758

Using a Business Rule Flow in a Job

Overview

The Business Rules transformation enables you use the business rule flow packages that are created in SAS® Business Rules Manager in the context of a SAS Data Integration Studio job. You can import business rule flows, specify flow versions, map source table columns to required input columns, and set business rule options.

The transformation enables you to map your source data and output data into and out of the rules package. Then, the SAS Data Integration Studio job applies the rules to your data as it is run. When a job that contains a rules package is run, statistics are collected about the following:

- how many rules were triggered
- the number of invalid and valid data record values
- other details

You can use this information to further refine your data as it flows through your transformation logic.

You must satisfy the following prerequisites before you can use the Business Rules transformation in a SAS Data Integration Studio job:

- One or more business rule flows must have been created in SAS® Business Rules Manager. The outputs created for one of these business rule flows determine the outputs for the Business Rule transformation that the flow is used in a job.
- These flows must be available in the Folders tree in SAS Data Integration Studio. A typical path through the Folders tree is **Folders** ⇨ **Products** ⇨ **SAS Business Rules Studio** ⇨ **Business Rule Flows 1**.

Problem

You want to use a business rule flow to structure the analysis of some of your data.

Solution

You can create a SAS Data Integration Studio job that contains the Business Rules transformation. Perform the following tasks:

- "Verify Prerequisites" on page 676
- "Create the Job" on page 676
- "Configure the Job" on page 677
- "Run the Job and Review the Results" on page 678

Tasks

Verify Prerequisites

The SAS Business Rules Manager must be installed on the server where the job that contains the Business Rules transformation is executed. Verify that this prerequisite has been satisfied.

Create the Job

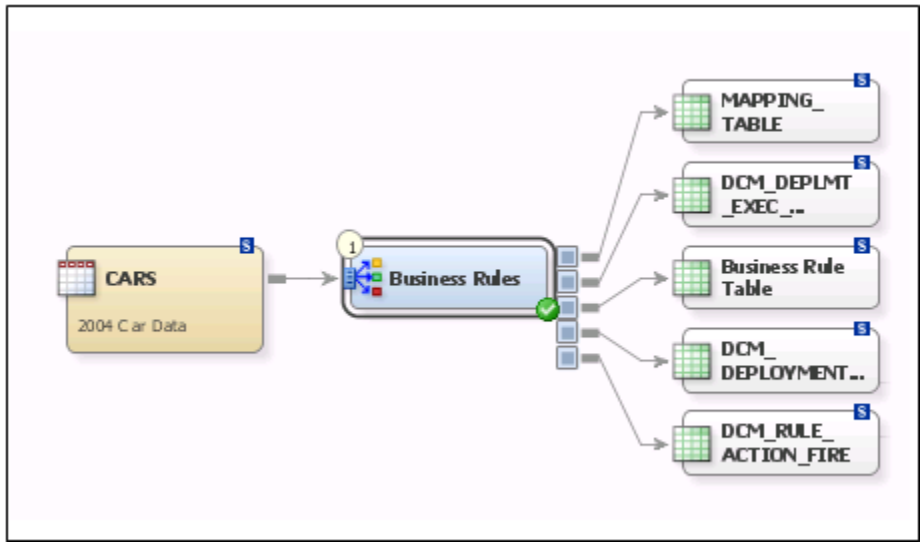
Perform the following steps to create a job that uses a business rule flow:

1. Create an empty SAS Data Integration Studio job.
2. Navigate to the **SAS Business Rules Studio** folder on the **Folders** tab, and open the folder that contains the business rule flow that you want to use.
3. Select the business rule flow and drop it in the empty job on the **Diagram** tab in the Job Editor window. A Business Rules transformation populated with the selected business rule flow is displayed on the tab.

   Note: You can also drop a Business Rules transformation directly onto the **Diagram** tab. However, you must then open the **Business Rule** tab in the transformation and select the flow in the **Business rule flow** field.
4. Locate the source table that contains the data that you want to process in the business rule flow. The source table for the sample job is named CARS.
5. Drop the source table onto the **Diagram** tab and connect it to the Business Rules transformation.
6. Right-click the Business Rules transformation and click **Expand** in the pop-up menu. This step expands the outputs so that you can easily review them.

The following display shows the completed job flow:

Display A3.1 *Business Rules Flow*

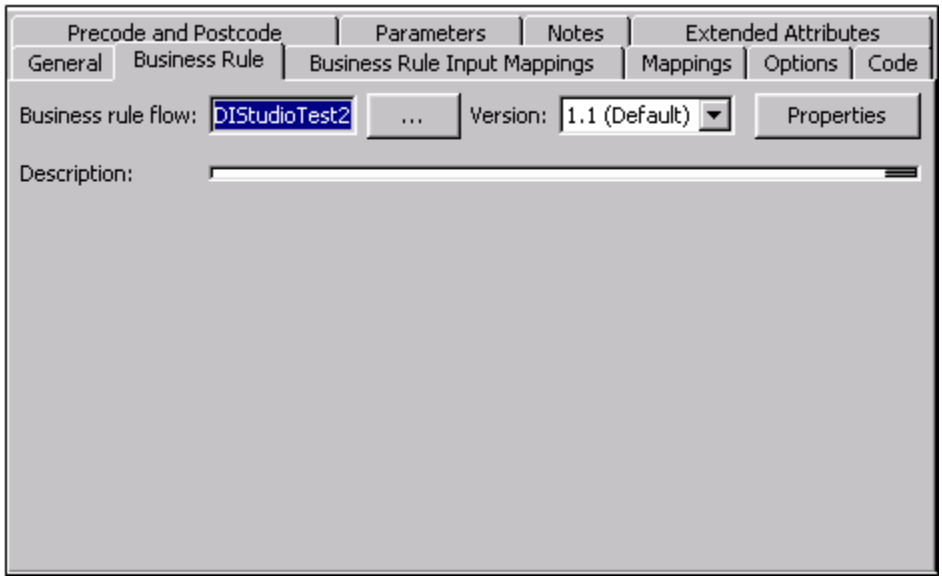


Configure the Job

You need to ensure that the business rule flow was properly populated and mapped when you created the job.

First, you should check the **Business Rule** tab, as shown in the following display:

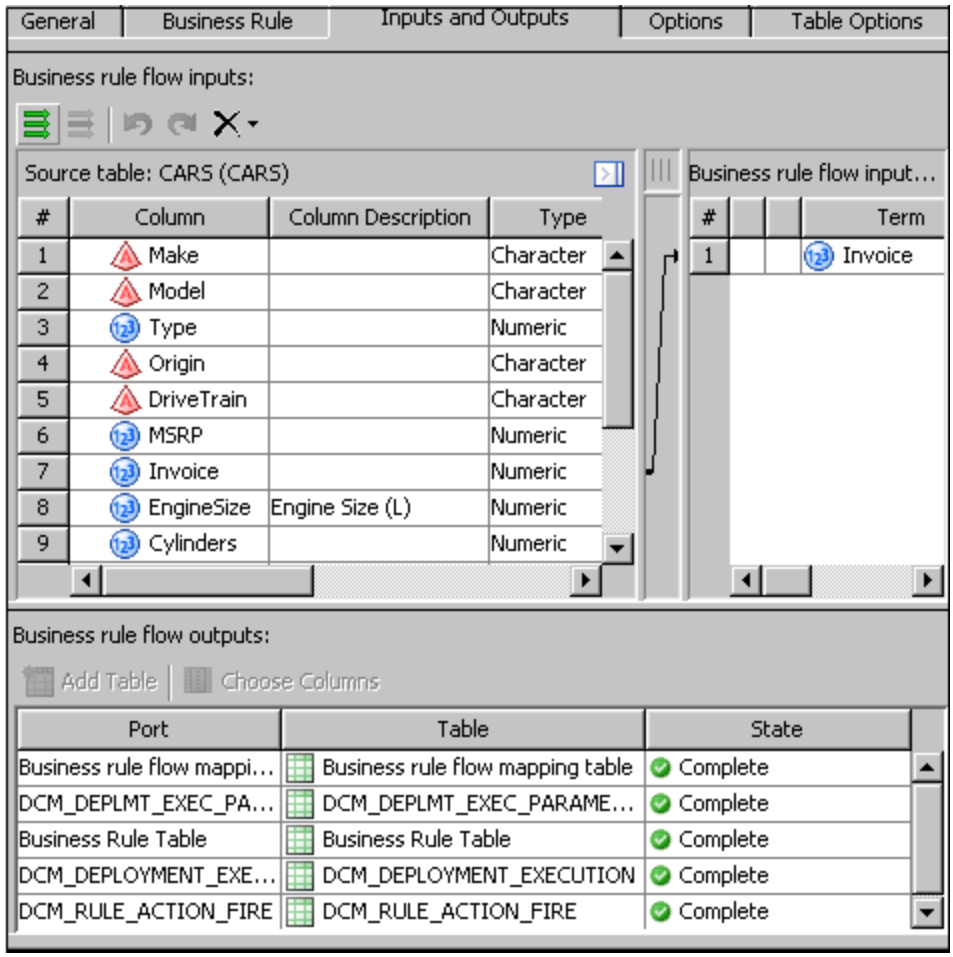
Display A3.2 *Business Rule Tab*



You can see that version 1.1 the DIStudioTest2 flow has been populated into the Business Rules transformation. If necessary, you can select a new business rule flow and version in the appropriate fields.

You should also review the **Inputs and Outputs** tab, as shown in the following display:

Display A3.3 *Inputs and Outputs Tab*



Make sure that the Inputs column in the source table is mapped to the corresponding term in the business rule flow input. You also should examine the business rule flow outputs at the bottom of the tab. If necessary, you can add tables and select additional columns.

Run the Job and Review the Results

Run the job and verify that the job completes without error. Then, open the output tables to verify that the expected data is populated.

Note: By default the Business Rules transformation will generate a rule fire action table. With large data volumes or frequent rule triggers, the rule fire action table can grow large enough to impact system performance. If that happens, you can turn off the generation of the rule fire action table. To do so, open the properties window for the Business Rule transformation. Then, click the **Options** tab and the **Business Rule Flow Advanced Options** section of the tab. Select **No** in the drop-down menu in the **Generate rules fired table** field.

Usage Notes

The temporary work tables for the Business Rule transformation should be left in SAS data set format. Otherwise, the transformation might fail.

Transformations in a job typically create temporary work tables as they execute. The default work tables are SAS data sets in the SAS WORK library. You can redirect these temporary tables to an alternative format and location, as described in "Redirecting Temporary Output Tables" on page 196. However, the temporary work tables for the Business Rule transformation should be left in SAS data set format.

If a job with a Business Rule transformation fails with errors, try turning off the **Enable optional metadata macro variables** option for the job. This option is required when the job is executed on a SAS 9.1 Workspace Server, but it is not required if the job is executed on a newer server. To turn off this option for one job, perform the following steps.

1. Open the job in the process editor.
2. Right-click the canvas of the job and select **Properties**. The properties window for the job appears.
3. Click the **Options** tab.
4. Deselect the **Enable optional metadata macro variables** option.
5. Click **OK** to save your changes.
6. Run the job.

If the job now runs without error, consider turning off the **Enable optional metadata macro variables** option at the global level so that this option is turned off for all new jobs. To turn off this option for all new jobs, perform the following steps.

1. Select **Tools** ⇨ **Options** from the main menu.
2. Click the **Code Generation** tab.
3. Deselect the **Enable optional metadata macro variables** option.
4. Click **OK** to save your changes.

Creating a Table That Appends Two or More Source Tables

Overview

Use the Append transformation to create a single target by appending (concatenating) two or more sources.

Note: You cannot append a table to itself.

Problem

You want to combine data from several source tables into a single target table.

Solution

You can use the Append transformation in a SAS Data Integration Studio job to combine the source tables into the target table. For example, you can create a job similar to the sample job featured in this topic. This sample job combines several months of sales data into a table that contains quarterly sales data. The sample job includes the following tasks:

- "Create and Populate the Job" on page 680
- "Run the Job and View the Output" on page 681

Tasks

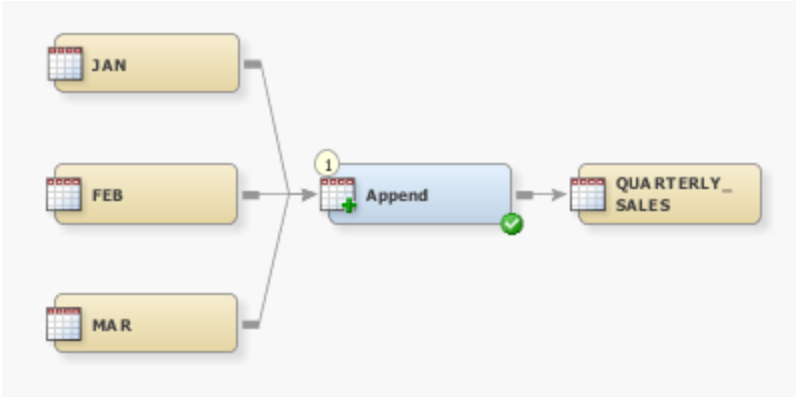
Create and Populate the Job

Perform the following steps to create and populate the job:

1. Create an empty SAS Data Integration Studio job.
2. Select and drag an Append 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 the source tables from the Inventory tree. Then, drop them before the Append transformation on the **Diagram** tab.
4. Drag the cursor from the source tables to the input port of the Append transformation. This action connects the sources to the transformation.
5. Because you want to have a permanent target table to contain the output for the transformation, 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 following display shows a sample process flow diagram for a job that contains the Append transformation.

Display A3.4 *Sample Process Flow*



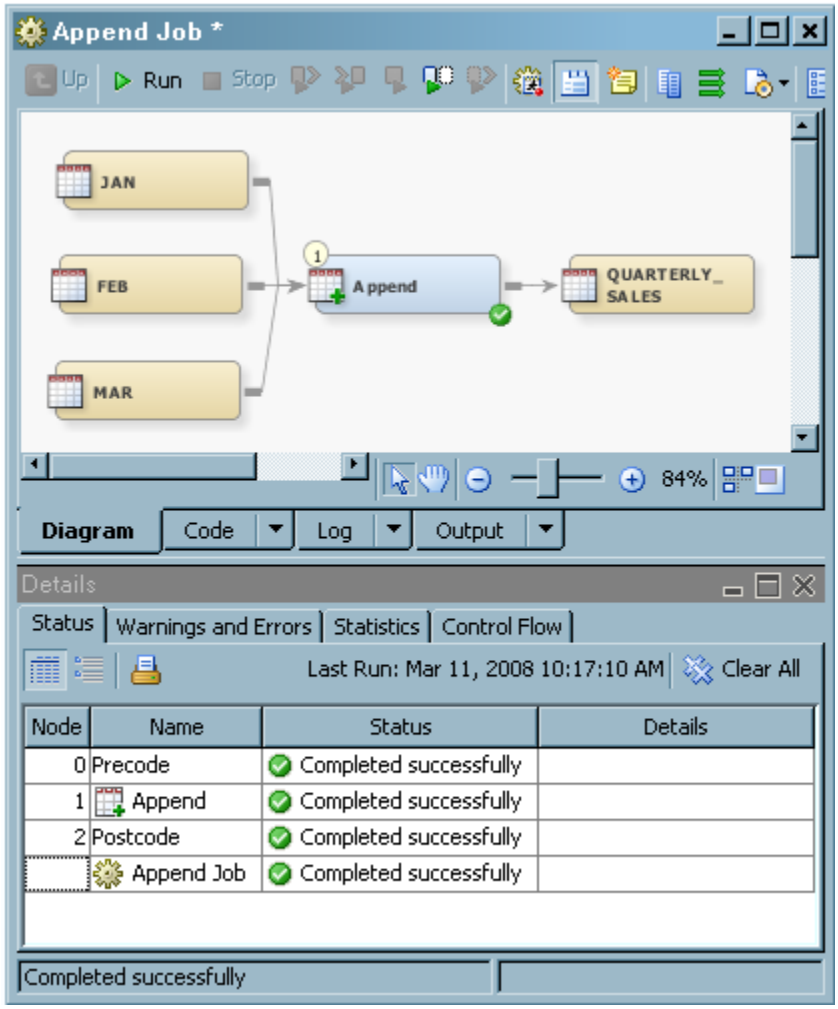
The source tables for the sample job are named Jan, Feb, and Mar. The target table for the sample job is named QUARTERLY_SALES. It contains sales information for specific customers for the months of January, February, and March.

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. The following display shows a successful run of a sample job.

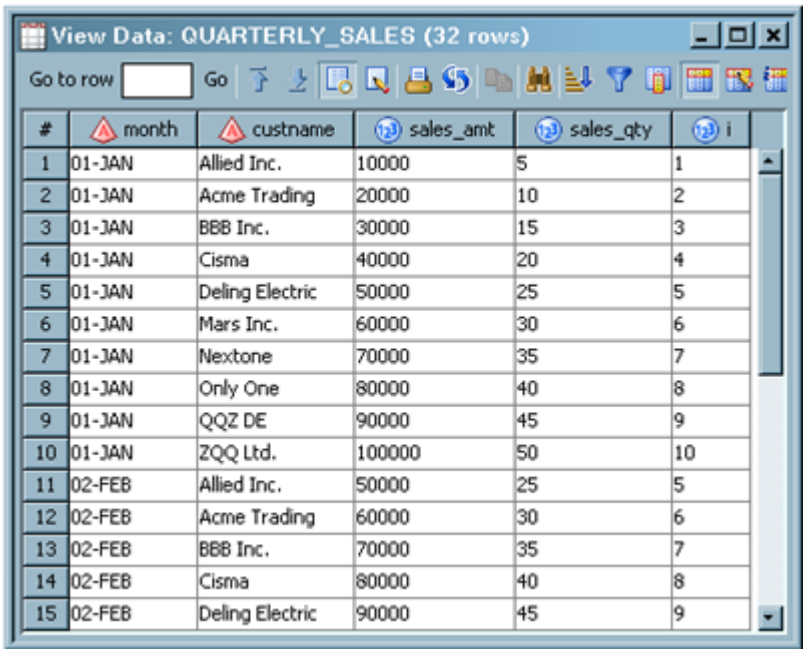
Display A3.5 *Sample Completed Job*



2. If error messages display, read and respond to the messages as needed.

3. Right-click the target table in the **Diagram** tab. Then, click **Open** in the pop-up menu. The following display shows the target table data for the sample job.

Display A3.6 *Sample Target Table in the View Data Window*



#	month	custname	sales_amt	sales_qty	i
1	01-JAN	Allied Inc.	10000	5	1
2	01-JAN	Acme Trading	20000	10	2
3	01-JAN	BBB Inc.	30000	15	3
4	01-JAN	Cisma	40000	20	4
5	01-JAN	Deling Electric	50000	25	5
6	01-JAN	Mars Inc.	60000	30	6
7	01-JAN	Nextone	70000	35	7
8	01-JAN	Only One	80000	40	8
9	01-JAN	QQZ DE	90000	45	9
10	01-JAN	ZQQ Ltd.	100000	50	10
11	02-FEB	Allied Inc.	50000	25	5
12	02-FEB	Acme Trading	60000	30	6
13	02-FEB	BBB Inc.	70000	35	7
14	02-FEB	Cisma	80000	40	8
15	02-FEB	Deling Electric	90000	45	9

Note that the first ten rows in the sample target table display data from the Jan table. The next five rows display data from the Feb table. The remaining rows display the rest of the data from the source tables.

Creating a Publish to Archive Report from Table Data

Overview

Use a Publish to Archive transformation to create an HTML report and an archive file so that the report can be re-created using SAS Package Retriever. You can control many aspects of how the report is created, including the following:

- the title of the report
- the location of the report and the archive
- which columns are analyzed

The Publish to Archive transformation uses the Publishing Framework feature of SAS Integration Technologies. This framework provides a complete and robust publishing environment for enterprise-wide information delivery. It consists of SAS CALL routines, application programming interfaces (APIs), and graphical user interfaces that enable both users and applications to publish SAS files (including data sets, catalogs, and database views), other digital content, and system-generated events to a variety of destinations such as e-mail addresses, message queues, publication channels and subscribers, WebDAV-compliant servers, and archive locations.

The Publishing Framework also provides tools that enable both users and applications to receive and process published information. For example, users can receive packages with content, such as charts and graphs, that is ready for viewing. SAS programs can

receive packages with SAS data sets that might in turn trigger additional analyses on that data.

Problem

You want to create and print an HTML report. Then, you want to save an archived version of the data.

Solution

You can use a Publish to Archive transformation in a job that creates and archives an HTML report. For example, you can create a job similar to the sample job featured in this topic. This sample job generates a report that is based on a table that contains information about business invoices. The sample job includes the following tasks:

- "Create and Populate the Job" on page 683
- "Configure SAS Table and Reporting Options" on page 684
- "Run the Job and View the Output" on page 686

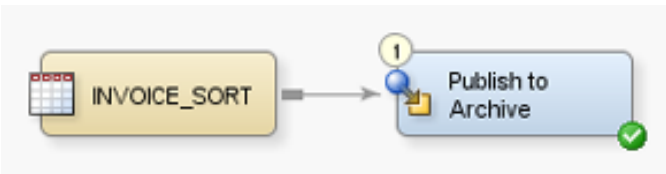
Tasks

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. Select and drag a Publish to Archive transformation from the Publish folder in the Transformations tree. Then, drop it in the empty job on the **Diagram** tab in the Job Editor window.
3. Right-click the Publish to Archive transformation and select **Ports** ⇨ **Add Input Port**.
4. Select and drag the source table from the Inventory tree. Then, drop it before the Publish to Archive transformation on the **Diagram** tab.
5. Drag the cursor from the source table to the input port of the Publish to Archive transformation. This action connects the source to the transformation.
6. Ensure that the output of the job can be sent to the **Output** tab of the Job Editor window. (If the **Output** tab is not displayed, enable it with the **Show Output tab** check box on the **General** tab of the **Options** item in the **Tools** menu.) The following display shows a sample process flow diagram for a job that contains the Publish to Archive transformation.

Display A3.7 *Sample Process Flow*



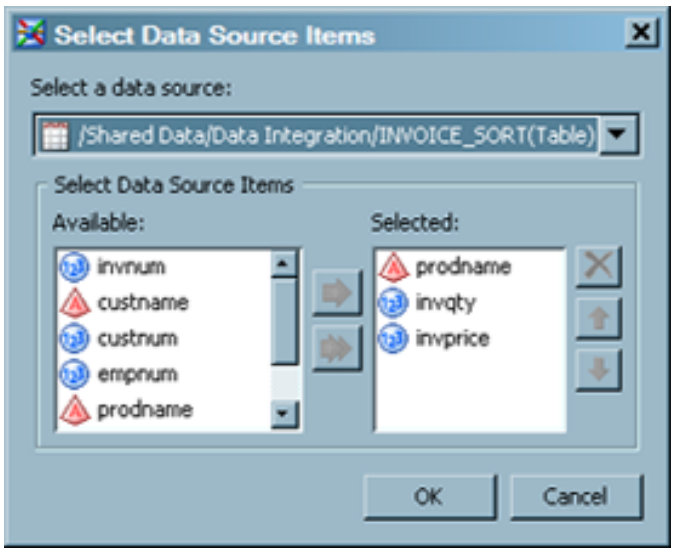
Note that the source table for the sample job is named INVOICE_SORTED.

Configure SAS Table and Reporting Options

Use the **Options** tab in the properties window for the Publish to Archive transformation to configure the SAS tables that are generated in the job and shape the output of your analysis. Note that the **Options** tab is divided into two parts, with a list of categories on the left-hand side and the options for the selected category on the right-hand side. Perform the following steps to set the options that you need for your job:

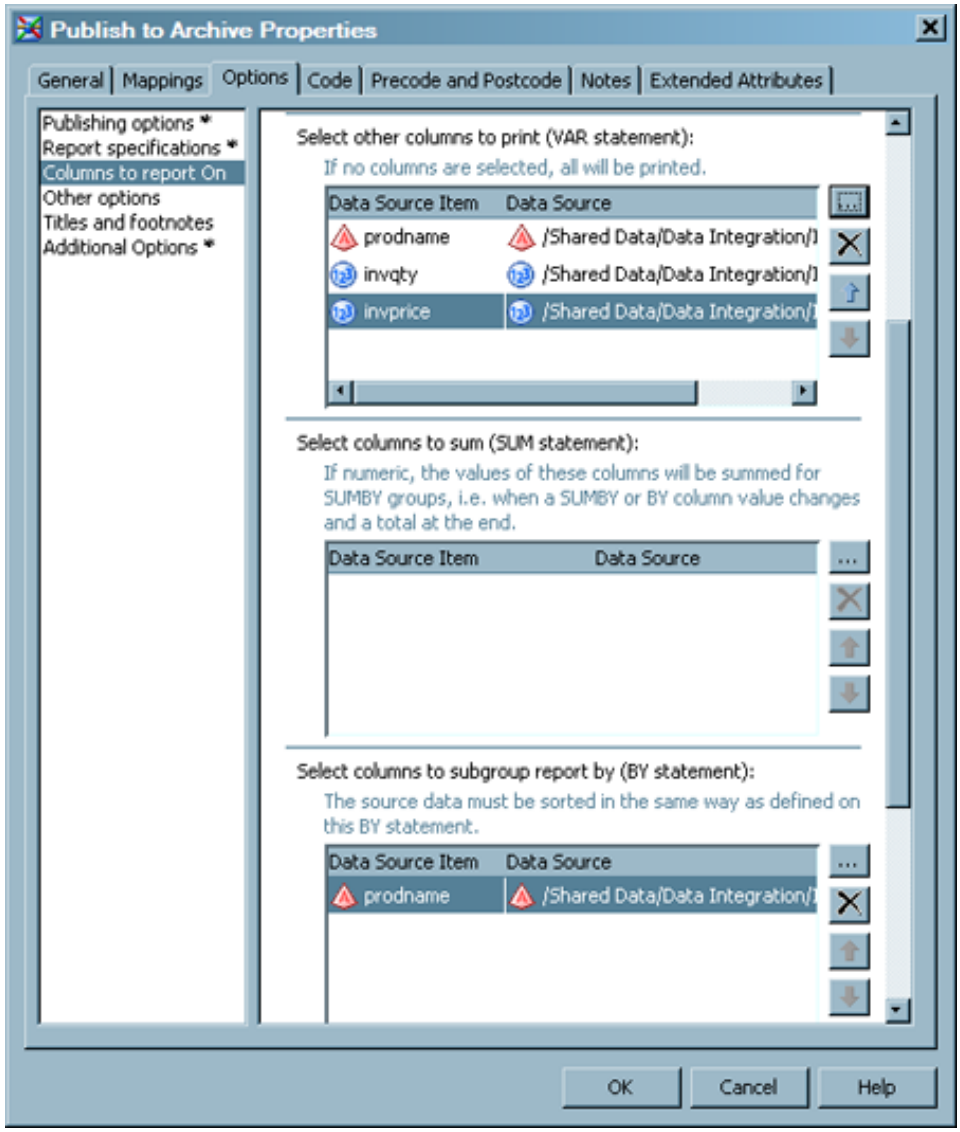
1. Open the properties window for the Publish to Archive transformation on the **Diagram** tab in the Job Editor window. Then, click the **Options** tab.
2. Click **Columns to report on** to access the Columns to report on columns page. Use the column selection prompts to access the columns that you need in the SAS tables included in your job. For example, you can click ... for the **Select other columns to print (VAR statement)** to access the Select Data Source Items window, as shown in the following display.

Display A3.8 *Sample Select Data Source Items Window*



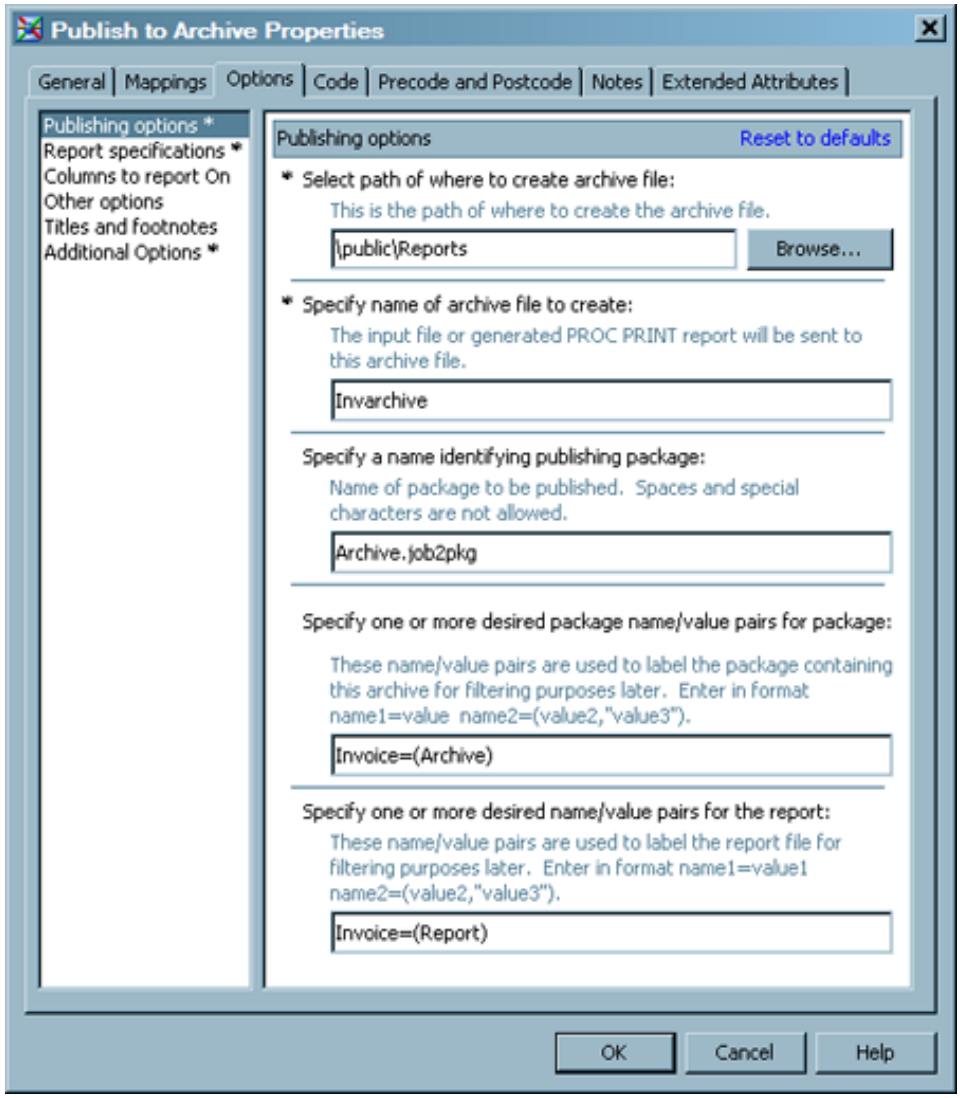
In the sample job, the VAR statement columns are prodname, invqty, and invprice. The column assignment options for the sample job are shown in the following display.

Display A3.9 *Sample Options Properties*



3. Set the publication options for the report on the Publishing options page. The options for the sample job are shown in the following display.

Display A3.10 *Sample Publication Options*



Note that the options to set a path for the archive file and specify the archive filename are required.

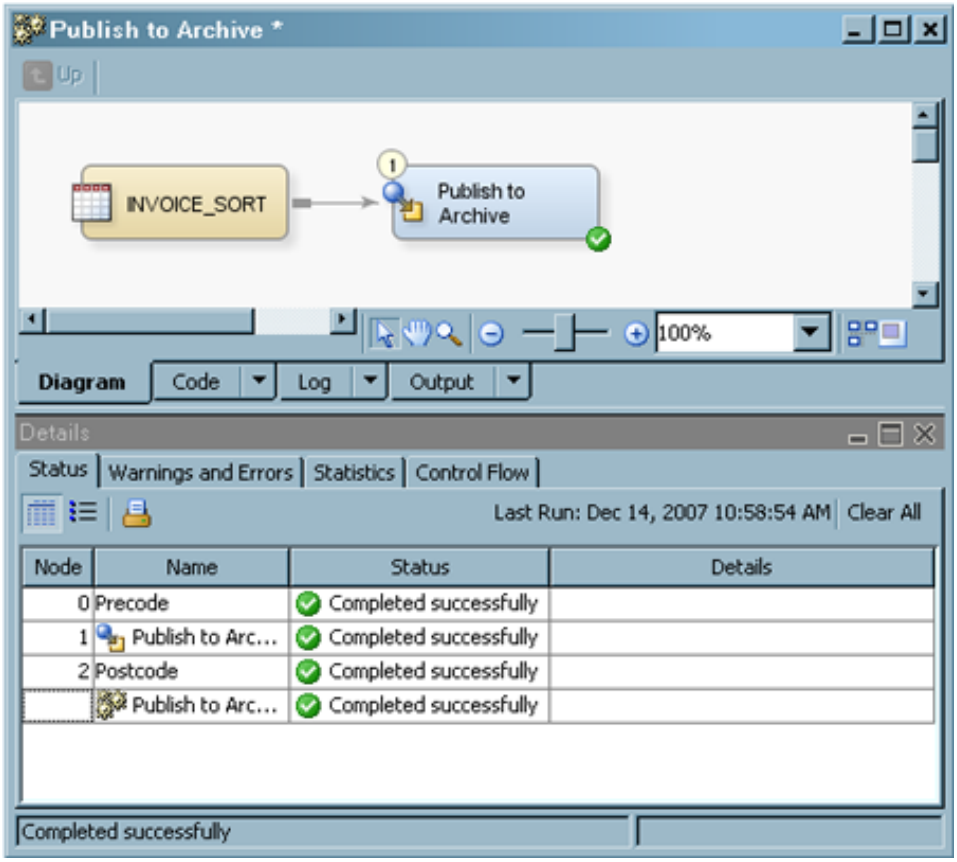
4. Set the remaining options for your report in the appropriate fields. For example, the path and filename for the report generated by the sample job are specified in the **Specify path and filename for generated report** field on the Report specifications page. (Make sure that you click **Generate PROC PRINT from input table** in the **Generate PROC PRINT or send existing report** field). A title for the sample job report is set on the Titles and Footnotes page.

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. The following display shows a successful run of a sample job.

Display A3.11 *Successfully Completed Sample Job*



2. If error messages are displayed on the **Status** tab, read and respond to the messages as needed.
3. To view the generated reports, click the **Output** tab in the Job Editor window. The following display shows the report for the sample job.

Display A3.12 *Sample Output in the Output Tab*

Invoice Analysis Archive

---------- prodname=flippers ----------

Obs	prodname	invqty	invprice
1	flippers	15	$19
2	flippers	15	$19
3	flippers	30	$18
4	flippers	20	$19
5	flippers	5	$20
6	flippers	10	$20
7	flippers	15	$19
8	flippers	25	$19

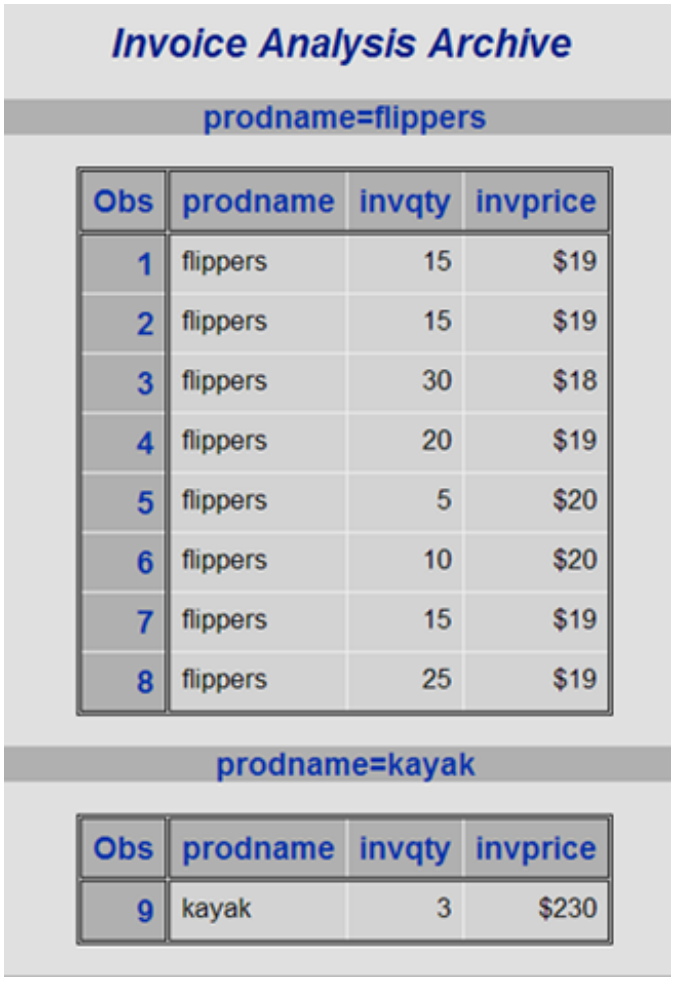
---------- prodname=kayak ----------

Obs	prodname	invqty	invprice
9	kayak	3	$230

You can find the archive file (invarchive.spk) in the directory that you specified. You can access the archive file with the SAS Package Retriever. The archive file is a compressed file that has to be unzipped for the report to be viewed.

4. Navigate to the HTML file using the path that you configured on the Report specifications page on the **Options** tab. Sample HTML output is shown in the following display.

Display A3.13 *Sample PDF Output*



Invoice Analysis Archive

prodname=flippers

Obs	prodname	invqty	invprice
1	flippers	15	$19
2	flippers	15	$19
3	flippers	30	$18
4	flippers	20	$19
5	flippers	5	$20
6	flippers	10	$20
7	flippers	15	$19
8	flippers	25	$19

prodname=kayak

Obs	prodname	invqty	invprice
9	kayak	3	$230

Validating Product Data

Overview

Use a Data Validation transformation to improve the quality of operational data before you load that data into a data warehouse or data mart. You can detect error conditions and specify actions that alleviate those errors. Error conditions include blank or missing values, duplicate values, and invalid values. The actions that you can take in response to erroneous values include stopping the job, changing the value, or writing the row to an error table instead of to the target.

Custom validation enables you to apply source values to user-written expressions. You then define the actions that are taken in response to true and false results. Custom actions include the replacement of source values in the target. Replacement values can be generated by a second expression, or they can be obtained from a translation table.

Each of the validation actions sends information to an exception report, which you can create on the **Error and Exception Tables** tab. You can specify the name and path of the exception report on the **Status Handling** tab.

Problem

You want to create a job that validates operational data before that data is loaded into a data warehouse or data mart.

Solution

You can use a Data Validation transformation to improve data quality by identifying and acting on duplicate values, invalid values, and missing values. Perform the following tasks to create the job:

- "Create and Populate the Job" on page 690
- "Configure Data Validation Settings" on page 691
- "Run the Job and View the Output" on page 692

You can also develop your own validation process that translates source values by using expressions or translation tables. The expressions can include the data quality functions that are available in the Expression Builder. In this example, source data on product revenues is validated before it is loaded into an enterprise data warehouse. Source rows with duplicate product numbers or with invalid product names are written to an error table, and valid rows are written to a table in the warehouse.

Tasks

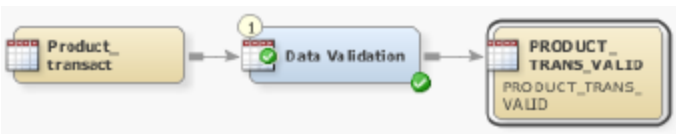
Create and Populate the Job

Perform the following steps to create and populate the job:

1. Create an empty SAS Data Integration Studio job.
2. Select and drag a Data Validation 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.
3. Select and drag the source table from the Inventory tree. Then, drop it before the Data Validation transformation on the **Diagram** tab. The source table for this sample job is Product_transact.
4. Drag the cursor from the source table to the input port of the Data Validation transformation. This action connects the source to the transformation.
5. Because you want to have a permanent target table to contain the output for the transformation, right-click the temporary work table attached to the transformation. Then click **Replace** in the pop-up menu. Finally, use the Table Selector window to select the target table for the job. The target table must be registered in SAS Data Integration Studio.

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

Display A3.14 *Sample Process Flow*



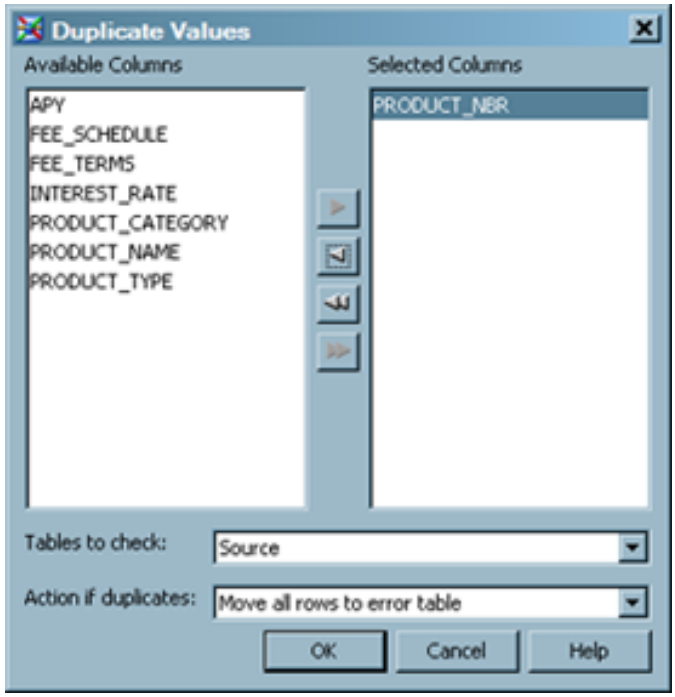
In the sample job, the lookup table is a SAS data set named PRODUCT_FACT. It contains the same columns as the source and target and contains valid values in the PRODUCT_NAME column. The source table is named Product_transact, and the target table is named PRODUCT_TRANS_VALID. When the job is run, invalid values are found in source rows that contain a product name that does not appear in the lookup table. These rows are written to an error table instead of the target table. The error table is new, so a new table is created.

Configure Data Validation Settings

You can use the properties windows for the Data Validation transformation to perform the necessary configuration to validate your data. For example, the locations of the error and exception tables are specified in the sample job. Then, status handling conditions and actions are specified. Finally, the processing of duplicate and invalid values is configured for the job. Perform the following steps to complete this configuration:

1. Open the properties window for Data Validation transformation on the **Diagram** tab in the Job Editor window.
2. Click the **Error and Exception Tables** tab. Then, select the **Create error table** and **Create exception table** check boxes to create error and exception tables. Click the **Choose columns** buttons for both tables to select appropriate columns.
3. Click the **Status Handling** tab. Then, click **New** to add a new data exception row to the table.
4. Select **Email Exception Report** from the Action column for the table to access the Action Options window. The exception report stores messages that describe the actions that take place when you run the job.
5. Enter the e-mail address of the error report destination in the **Email Address** field. Click **OK** to save the address.
6. Click the **Duplicate Values** tab. Then, click **New column** to display the Duplicate Values window.
7. Move the appropriate column from the **Available Columns** field to the **Selected Columns**. The sample job uses the PRODUCT_NBR column.
8. Select appropriate values in the **Tables to check** and **Action if duplicates** fields. The sample job retains the default values of **Source** and **Move all rows to error table**. These values ensure that source rows with duplicate values are moved to the error table that is specified on the **Error and Exception Tables** tab. The Duplicate Values window for the sample job is shown in the following display.

Display A3.15 *Sample Duplicate Values Window*



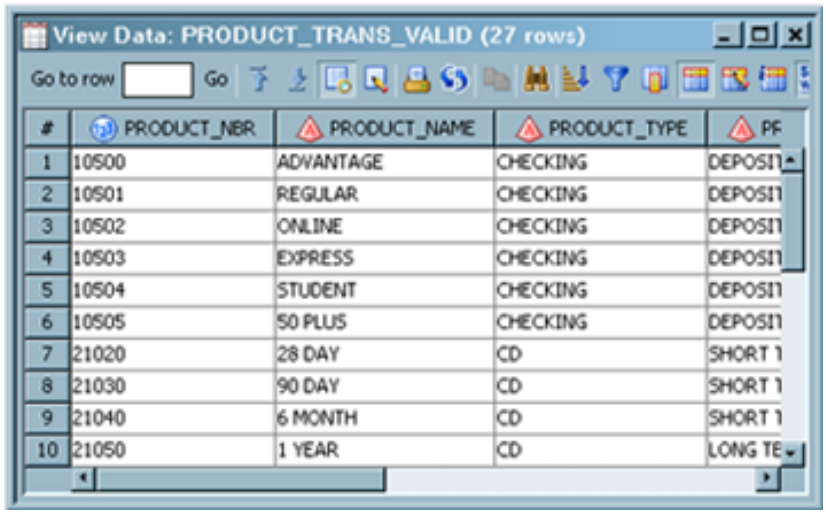
9. Click the **Invalid Values** tab. Then, click **New column** to display the Invalid Values window.
10. Select the appropriate column in the **Column Name** field. The sample job uses the PRODUCT_NAME column.
11. Click the **Lookup a table and a column** button to access the Lookup Table and Column window.
12. Navigate to the appropriate table and select the column that you need. The sample job uses the PRODUCT_FACT table and the PRODUCT_NAME column. Click **OK** to save the table and column settings and return to the Invalid Values window.
13. Make sure that the value in the **Action if invalid** field is correct. The sample job keeps the default **Move row to error table** value.
14. Click **OK** to save the invalid values settings. Then, click **OK** again to save the properties window settings for the Data Validation transformation.

Run the Job and View the Output

Perform the following steps to run the job:

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.
3. To view rows that are validated and displayed in the target table, right-click the table and click **Open**. The following display shows the target table for the sample job.

Display A3.16 *Sample Validated Data*



#	PRODUCT_NBR	PRODUCT_NAME	PRODUCT_TYPE	PF
1	10500	ADVANTAGE	CHECKING	DEPOSIT
2	10501	REGULAR	CHECKING	DEPOSIT
3	10502	ONLINE	CHECKING	DEPOSIT
4	10503	EXPRESS	CHECKING	DEPOSIT
5	10504	STUDENT	CHECKING	DEPOSIT
6	10505	50 PLUS	CHECKING	DEPOSIT
7	21020	28 DAY	CD	SHORT T
8	21030	90 DAY	CD	SHORT T
9	21040	6 MONTH	CD	SHORT T
10	21050	1 YEAR	CD	LONG TE

4. Check the error table, which contains the rows that fail validation. The following display shows the error table for the sample job.

Display A3.17 *Sample Error Table*

VIEWTABLE: TMP3.prod_error_table

	PRODUCT_NBR	PRODUCT_NAME	PRODUCT_TYPE	PRODUCT_CATEGORY	INTEREST_RATE
1	21000	MONEY MRKT	MONEY MARKET	SAVINGS	0.014
2	21010	REG SAVINGS	SAVINGS	SAVINGS	0.01
3	23150	TRAD IRA	IRA	RETIREMENT	0.0425
4	23170	EDUC IRA	IRA	EDUCATIONAL	0.0425
5	30030	HOME EQTY	HOME EQUITY LOAN	BORROWING	0.07

5. Verify that the e-mail notification of the data exception has been sent. The following display shows a portion of the notification for the sample job.

Display A3.18 *Sample E-mail Notification*

14Mar08:15:33:20 Data Exception

You can also review the exceptions table, which contains the rows that generate exceptions.

Creating a Publish to Email Report from Table Data

Overview

Use a Publish to Email transformation to create and e-mail an HTML report. You can control many aspects of how the report is created, such as the following:

- the title of the report
- the location of the report and the archive

- which columns are analyzed
- where the report is e-mailed

The following types of output are available:

- a report that can be viewed within an e-mail message
- a report in an archive (.spk) file as an e-mail attachment
- notification that a report was published

The Publish to Archive transformation uses the Publishing Framework feature of SAS Integration Technologies. This framework provides a complete and robust publishing environment for enterprise-wide information delivery. It consists of SAS CALL routines, application programming interfaces (APIs), and graphical user interfaces that enable both users and applications to publish SAS files (including data sets, catalogs, and database views), other digital content, and system-generated events to a variety of destinations such as e-mail addresses, message queues, publication channels and subscribers, WebDAV-compliant servers, and archive locations.

The Publishing Framework also provides tools that enable both users and applications to receive and process published information. For example, users can receive packages with content, such as charts and graphs, that is ready for viewing. SAS programs can receive packages with SAS data sets that might in turn trigger additional analyses on that data.

Note that e-mail must be enabled for the SAS Workspace Server that executes the job that includes the Publish to Email transformation. For more information, administrators should see the section called "Add or Modify E-Mail Settings for SAS Application Servers" in the *SAS Intelligence Platform: Application Server Administration Guide*.

Problem

You want to create and print an HTML report. Then, you want to send it by e-mail to a designated recipient.

Solution

You can use a Publish to Email transformation in a job creates and emails an HTML report. For example, you can create a job similar to the sample job featured in this topic. This sample job generates a report that is based on a table that contains information about business invoices and emails it to a specified address. The sample job includes the following tasks:

- “Create and Populate the Job” on page 694
- “Configure SAS Table and Reporting Options” on page 695
- “Run the Job and View the Output ” on page 697

Tasks

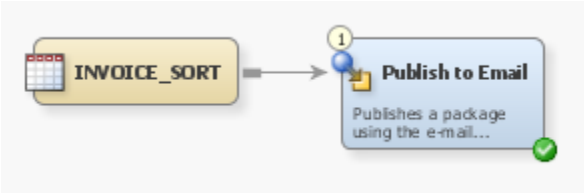
Create and Populate the Job

Perform the following steps to create and populate the job:

1. Create an empty SAS Data Integration Studio job.

2. Select and drag a Publish to Email transformation from the Publish folder in the Transformations tree. Then, drop it in the empty job on the **Diagram** tab in the Job Editor window.
3. Right-click the Publish to Email transformation and select **Ports** ⇨ **Add Input Port**.
4. Select and drag the source table out of the Inventory tree. Then, drop it before the Publish to Email transformation on the **Diagram** tab.
5. Drag the cursor from the source table to the input port of the Publish to Email transformation. This action connects the source to the transformation.
6. Ensure that the output of the job can be sent to the **Output** tab of the Job Editor window. (If the **Output** tab is not displayed, enable it with the **Show Output tab** check box in the **General** tab of the **Options** item in the **Tools** menu.) The following display shows a sample process flow diagram for a job that contains the Publish to Email transformation.

Display A3.19 *Sample Process Flow*



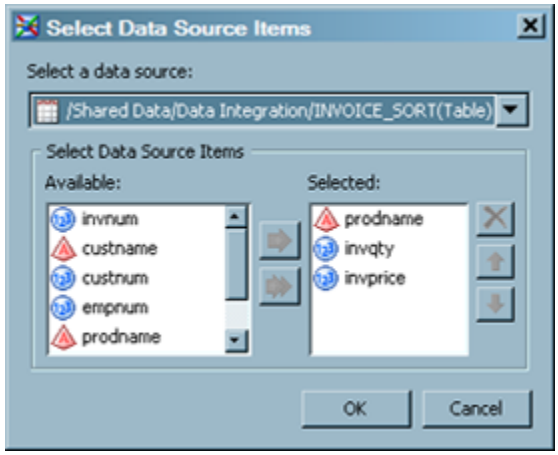
Note that the source table for the sample job is named INVOICE_SORT.

Configure SAS Table and Reporting Options

Use the **Options** tab in the properties window for the Publish to Email transformation to configure the SAS tables that are generated in the job and shape the output of your analysis. Note that the **Options** tab is divided into two parts, with a list of categories on the left-hand side and the options for the selected category on the right-hand side. Perform the following steps to set the options that you need for your job:

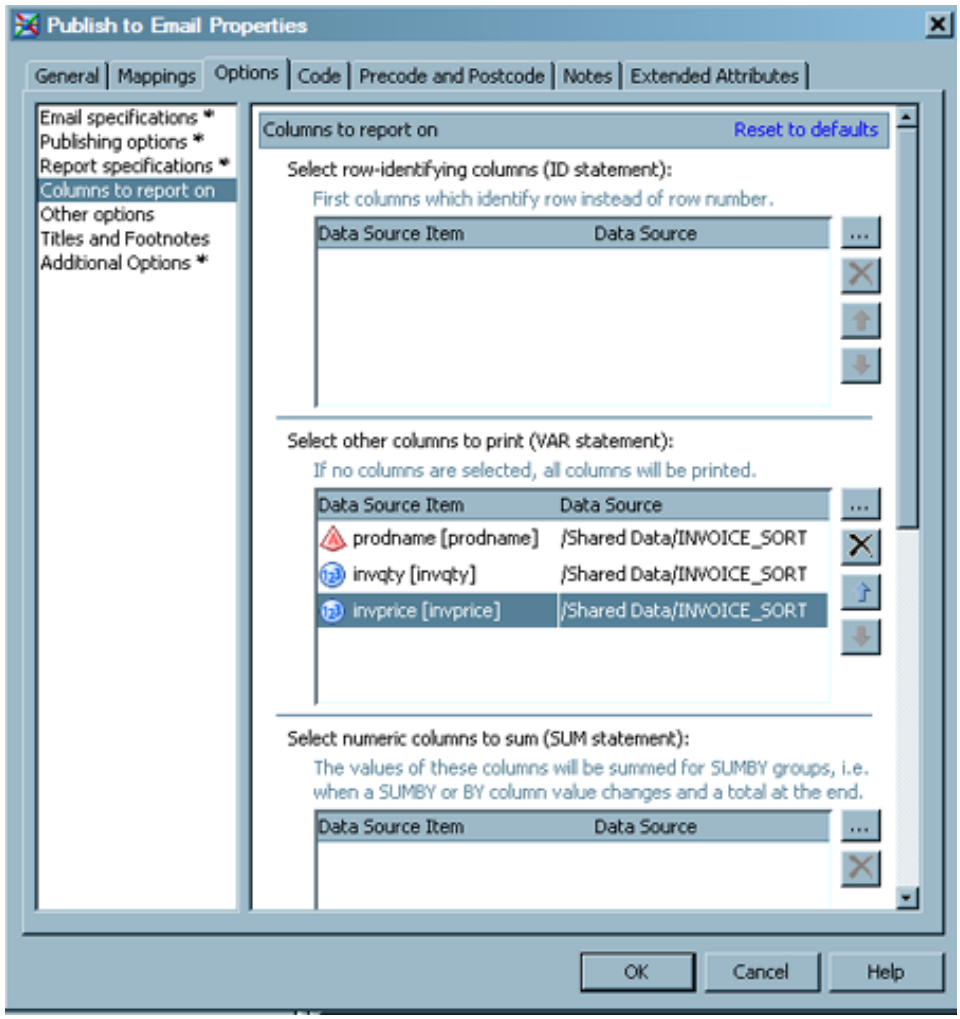
1. Open the properties window for the Publish to Email transformation in the **Diagram** tab in the Job Editor window. Then, click the **Options** tab.
2. Click **Email specifications** to access the e-mail specifications page. Enter the e-mail address for the report destination in the **Specify email addresses** field. Click **Adds a new item** to add the address to the list.
3. Click **Columns to report on** to access the Columns to report on columns page. Use the column selection prompts to access the columns that you need in the SAS tables included in your job. For example, you can click  for the **Select other columns to print (VAR statement)** to access the Select Data Source Items window, as shown in the following display.

Display A3.20 *Sample Select Data Source Items Window*



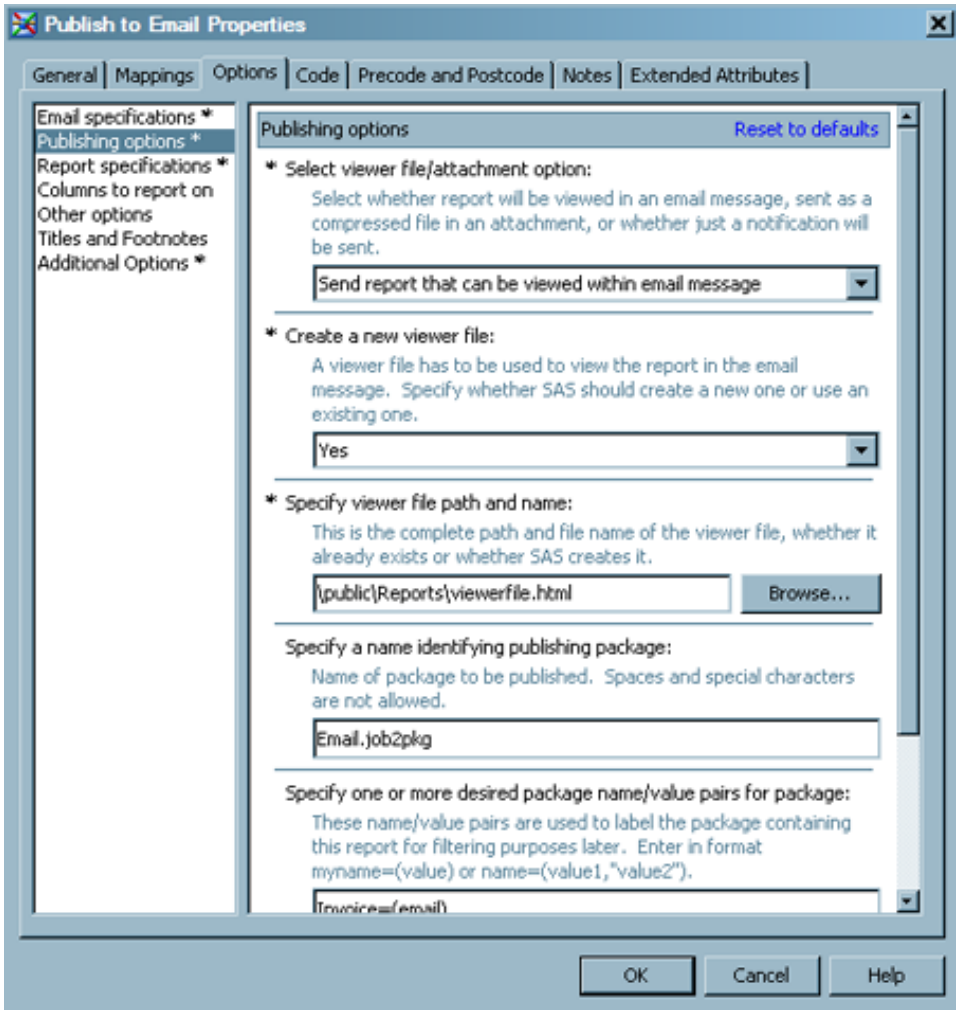
In the sample job, the VAR statement columns are prodname, invqty, and invprice. The column assignment options are shown in the following display.

Display A3.21 *Sample Options Properties*



4. Set the publication options for the report on the Publishing options page. The options for the sample job are shown in the following display.

Display A3.22 *Sample Publication Options*



Note that the first three options on this page are required. They configure the viewer file for the e-mail used to transmit the report. In the sample job, the output is sent in an e-mail message (email.html). You can also send the output in an archive file that is attached to an e-mail message or send an e-mail notification that the output was published.

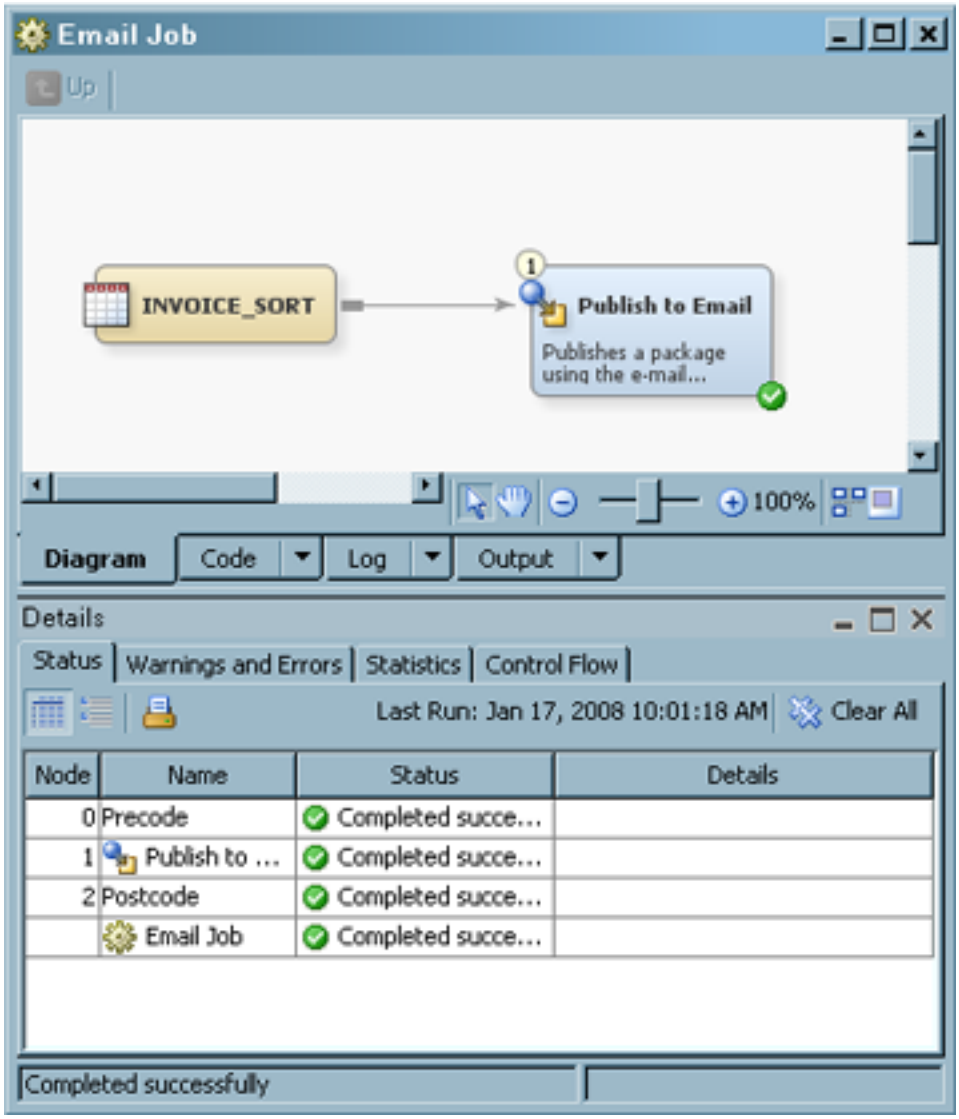
5. Set the remaining options for your analysis in the appropriate fields. For example, the path and filename for the report generated by the sample job are specified in the **Specify path and filename for generated report** field on the Report specifications page. (Make sure that you click **Generate PROC PRINT from input table** in the **Generate PROC PRINT or send existing report** field). A title for the sample job report is set on the Titles and Footnotes page.

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. The following display shows a successful run of a sample job.

Display A3.23 *Sample Publish to Email Job*



2. If error messages are displayed on the **Status** tab, read and respond to the messages as needed.

3. To view the generated reports, click the **Output** tab in the Job Editor window. The following display shows a portion of the report for the sample job.

Display A3.24 *Sample E-mail Output*

```
                          The SAS System

------------------------ prodname=flippers ---------------------

                Obs    prodname    invqty    invprice

                  1    flippers      15        $19
                  2    flippers      15        $19
                  3    flippers      30        $18
                  4    flippers      20        $19
                  5    flippers       5        $20
                  6    flippers      10        $20
                  7    flippers      15        $19
                  8    flippers      25        $19

-------------------------- prodname=kayak ---------------------

                Obs    prodname    invqty    invprice

                  9     kayak         3        $230
```

4. The same output is saved to a directory that you designate in the Publishing Option section of the **Options** tab. It is also sent to the e-mail address that you designate in the e-mail specifications section of the **Options** tab. The following display shows a portion of the e-mailed report for the sample job.

The SAS System

prodname=flippers

Obs	prodname	invqty	invprice
1	flippers	15	$19
2	flippers	15	$19
3	flippers	30	$18
4	flippers	20	$19
5	flippers	5	$20
6	flippers	10	$20
7	flippers	15	$19
8	flippers	25	$19

prodname=kayak

Obs	prodname	invqty	invprice
9	kayak	3	$230

Integrating a SAS Enterprise Miner Model with Existing SAS Data

Overview

You can use a Mining Results transformation to integrate a SAS Enterprise Miner model with data sources in your SAS Data Integration Studio data warehouse. Using the Mining Results transformation enables you to associate a SAS Enterprise Miner model with a job and use that job to create an output table that applies the model to the source data.

Note: One or more SAS Enterprise Miner models must be registered in your metadata repository before you can use the Mining Results transformation.

Problem

You want to create a job that creates a target table from a SAS Enterprise Miner model.

Solution

You can use the Mining Results transformation to create a job that creates a target table from a SAS Enterprise Miner model.

In this example, a statistician uses SAS Enterprise Miner and historical home equity data to build a data mining model to predict if a customer might default on a home equity loan. After the statistician builds the model, it is registered in a SAS metadata repository that a SAS Data Integration Studio developer can use. Additional customer data is collected using SAS Data Integration Studio. The new data has the same customer information but does not contain the predictions about a customer's probability for defaulting on a home equity loan. The SAS Data Integration Studio developer applies the SAS Enterprise Miner model to the new data source to generate the prediction of the customer's probability to default on a home equity loan.

To use the Mining Results transformation, your SAS Enterprise Miner models must be registered in the same metadata repository that contains the sources for your job. You can use the SAS Enterprise Miner Configuration Wizard to associate your SAS Enterprise Miner metadata repository with the metadata server that is used for your SAS Data Integration Studio application. See the SAS Enterprise Miner online Help for more information about how to use this wizard.

Note: It is recommended that you group your models in trees to make them easier to find when using the Mining Results transformation.

For best performance, the target of a Mining Results transformation should have only those columns that are required. These columns include the required input variables form the source table and the output results that are specified in the SAS Enterprise Miner model.

Perform the following steps:

- "Create and Populate the Job" on page 701
- "Associate the SAS Enterprise Miner Model with the Job" on page 702
- "Run the Job and View the Output" on page 704

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. Select and drag a Mining Results transformation from the Transformations tree in the **Data Transforms** folder. Then, drop it in the empty job on the **Diagram** tab in the Job Editor window.
3. Drag the cursor from the source table to the input port of the Mining Results transformation. This action connects the source to the transformation. The following display shows the process flow for the sample job.

Display A3.25 *Mining Results Process Flow Diagram*



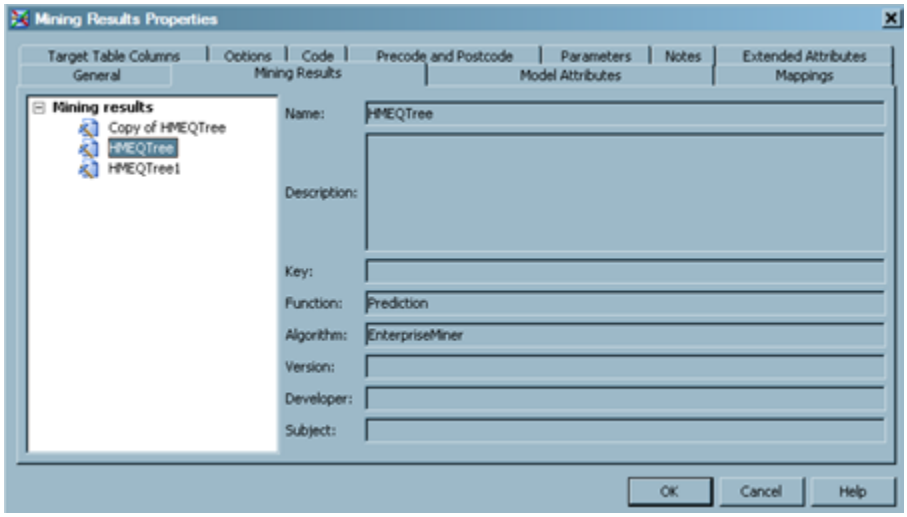
Note that the source file for the sample job is named HMEQ.

Associate the SAS Enterprise Miner Model with the Job

Perform the following steps to associate the SAS Enterprise Miner model with the Mining Results transformation:

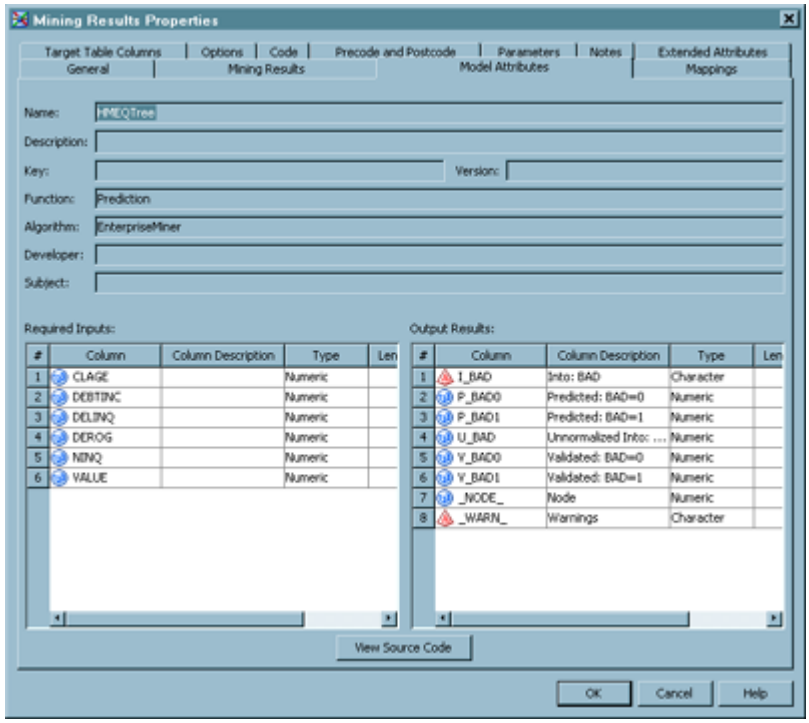
1. Display the transformation properties window.
2. Click the **Mining Results** tab.
3. Select the data mining tree model. Note that the **Mining Results** folder always contains all tree models, regardless of their assigned trees. Therefore, you can always find all available trees in the folder. Metadata about the selected tree model is displayed on the tab. The tree model used in the sample job in HMEQTree. The **Mining Results** tab for the sample job is shown in the following display.

Display A3.26 *Sample Mining Results Tab*



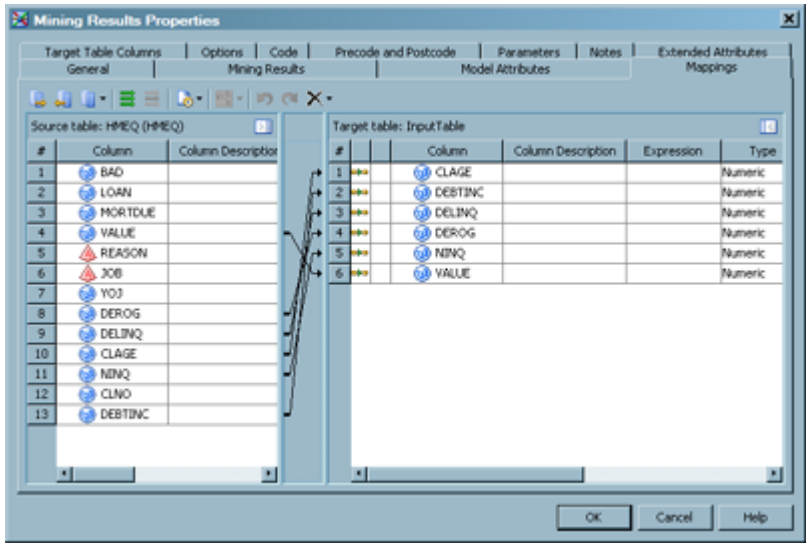
4. Click the **Model Attributes** tab. Use this tab to view the required input variables and output results that were defined for the model in SAS Enterprise Miner. You can also click **View Source Code** to view the SAS source code that is generated by the model. The **Model Attributes** tab for the sample job is shown in the following display.

Display A3.27 *Sample Model Attributes Tab*



5. Click the **Mapping** tab. The columns in the source and table tables are displayed.
6. Manually map the columns in the source table to the same columns in the target table. The mappings in the sample job are shown in the following display.

Display A3.28 *Sample Mapping Tab*

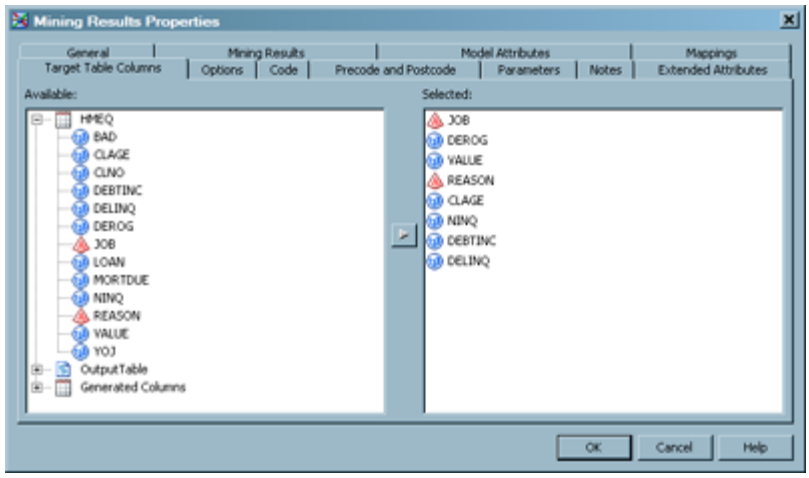


Note: The **Input Table** label on the **Target table** field is not displayed until a permanent target table is added to the job.

7. Click the **Target Table Columns** tab. In this tab, the **Available** field displays all columns in the source table and output table. It also displays all output results and any generated columns that are defined for the model.

8. Move the columns that you want to include in the target table into the **Selected** field. The **Target Table Columns** tab for the sample job is shown in the following display.

Display A3.29 *Sample Target Table Columns*



By default, only the columns that you must select from the source table are the required inputs and the model key columns, such as customer ID. From the output table, you can select columns starting with EM_ , P_, I_, or D_ that represent the SAS Enterprise Miner model predictions. You might want to include the **ModelKey** column and **Date** or **DateTime** columns also.

Note: For better performance, you should include only the required input variables and an identifier (such as Job) from the source table.

9. Click **OK** to save the settings and close the properties window.

10. Right-click the temporary output table attached to the Mining Results transformation. Then, click **Register Table** 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 following display shows a process flow for the sample job that includes a permanent target table.

Display A3.30 *Sample Process Flow with Target Table*



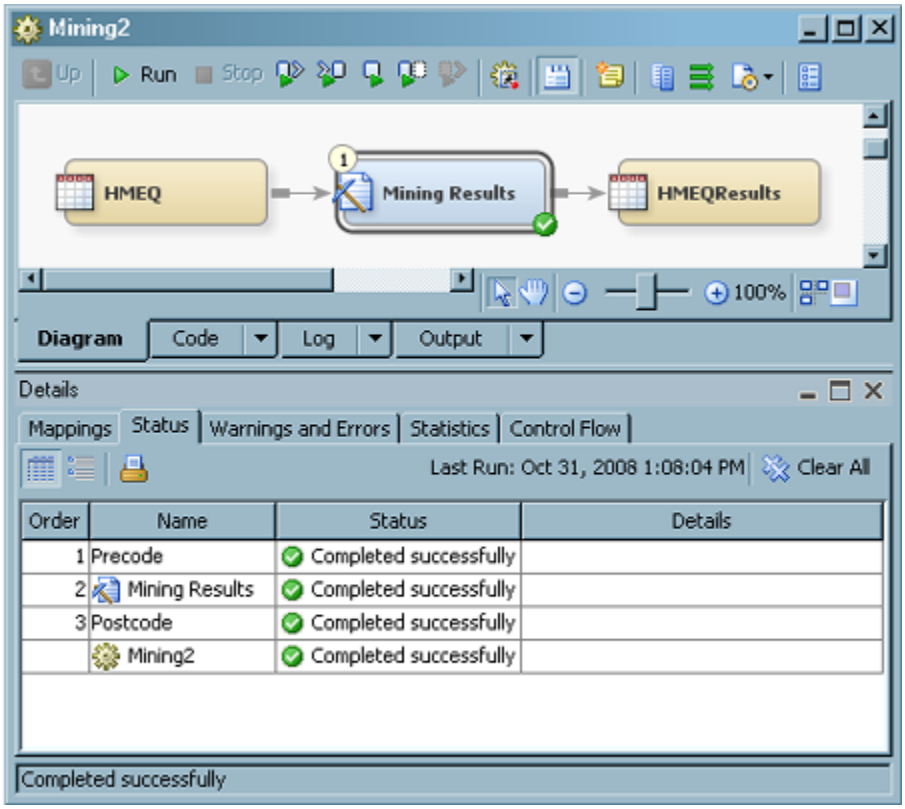
In the sample job, this permanent target table is named HMEQResults.

Run the Job and View the Output

Perform the following steps to run the job and verify that the job created the desired output:

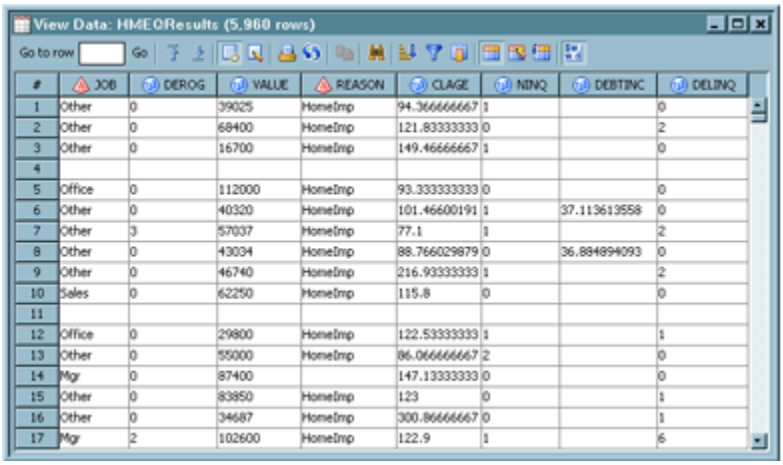
1. Right-click on an empty area of the job, and click **Run** in the pop-up menu. SAS Data Integration Studio generates code for the job and submits it to the SAS Application Server for execution. The following display shows a successful run of a sample job.

Display A3.31 *Completed Mining Results Job*



2. If error messages display, read, and respond to the messages as needed.
3. Right-click the target table in the **Diagram** tab. Then, click **Open** in the pop-up menu. The following display shows the target table data for the sample job.

Display A3.32 *Sample Target Table in the View Data Window*



#	JOB	DEROG	VALUE	REASON	CLAGE	NINQ	DEBTINC	DELINQ
1	Other	0	39025	HomeImp	94.36666667	1		0
2	Other	0	68400	HomeImp	121.8333333	0		2
3	Other	0	16700	HomeImp	149.4666667	1		0
4								
5	Office	0	112000	HomeImp	93.33333333	0		0
6	Other	0	40320	HomeImp	101.46600191	1	37.113613558	0
7	Other	3	57037	HomeImp	77.1	1		2
8	Other	0	43034	HomeImp	88.766029879	0	36.884894093	0
9	Other	0	46740	HomeImp	216.9333333	1		2
10	Sales	0	62250	HomeImp	115.8	0		0
11								
12	Office	0	29800	HomeImp	122.5333333	1		1
13	Other	0	55000	HomeImp	86.066666667	2		0
14	Mgr	0	87400		147.1333333	0		0
15	Other	0	83850	HomeImp	123	0		1
16	Other	0	34687	HomeImp	300.8666667	0		1
17	Mgr	2	102600	HomeImp	122.9	1		6

Creating a Publish to Queue Report from Table Data

Overview

Use the Publish to Queue transformation to create an HTML report that is sent to an MQSeries queue or a Microsoft MQ queue. Here are some of the many aspects that you can control when creating reports:

- the title of the report
- the location of the report and the archive
- the columns that are analyzed

The Publish to Archive transformation uses the Publishing Framework feature of SAS Integration Technologies. This framework provides a complete and robust publishing environment for enterprise-wide information delivery. It consists of SAS CALL routines, application programming interfaces (APIs), and graphical user interfaces that enable both users and applications to publish SAS files (including data sets, catalogs, and database views), other digital content, and system-generated events to a variety of destinations, such as e-mail addresses, message queues, publication channels and subscribers, WebDAV-compliant servers, and archive locations.

The Publishing Framework also provides tools that enable both users and applications to receive and process published information. For example, users can receive packages with content, such as charts and graphs, that is ready for viewing. SAS programs can also receive packages with SAS data sets that might in turn trigger additional analyses on that data.

Note: You must have MQSeries or Microsoft MQ installed before you can publish to a queue. In addition, the queue must exist before you publish to it, and you must have appropriate authorization to write to it.

Problem

You want to print an HTML report and send it to a queue using MQSeries or Microsoft MQ.

Solution

You can use the Publish to Queue transformation in a job creates an HTML report and sends it to a message queue. For example, you can create a job similar to the sample job featured in this topic. This sample job generates a report that is based on a table that contains information about business invoices. The sample job includes the following tasks:

- "Create and Populate the Job" on page 707
- "Configure Transformation Options" on page 707
- "Run the Job and View the Output" on page 710

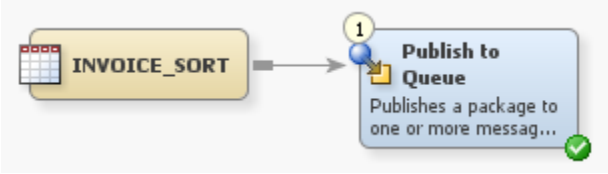
Tasks

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. Select and drag a Publish to Queue transformation from the Publish folder in the Transformations tree. Then, drop it in the empty job on the **Diagram** tab in the Job Editor window.
3. Right-click the Publish to Queue transformation and select **Ports** ⇨ **Add Input Port**.
4. Select and drag the source table out of the Inventory tree. Then, drop it before the Publish to Queue transformation on the **Diagram** tab.
5. Drag the cursor from the source table to the input port of the Publish to Queue transformation. This action connects the source to the transformation.
6. Ensure that the output of the job can be sent to the **Output** tab of the Job Editor window. (If the **Output** tab is not displayed, enable it with the **Show Output tab** check box in the **General** tab of the **Options** item in the **Tools** menu.) The following display shows a sample process flow diagram for a job that contains the Publish to Queue transformation.

Display A3.33 *Sample Process Flow*



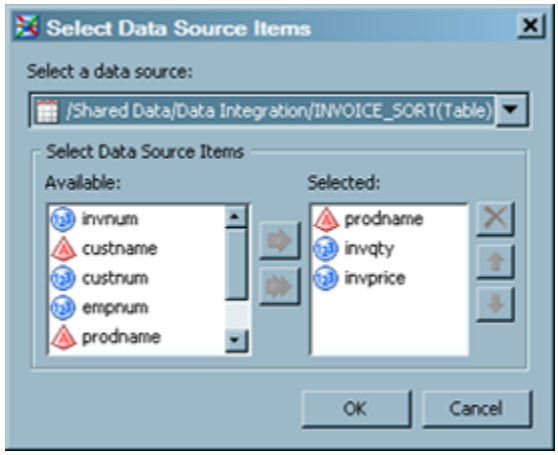
Note that the source table for the sample job is named INVOICE_SORT.

Configure Transformation Options

Use the **Options** tab in the properties window for the Publish to Queue transformation to configure the output for your analysis. Note that the **Options** tab is divided into two parts, with a list of categories on the left-hand side and the options for the selected category on the right-hand side. Perform the following steps to set the options that you need for your job:

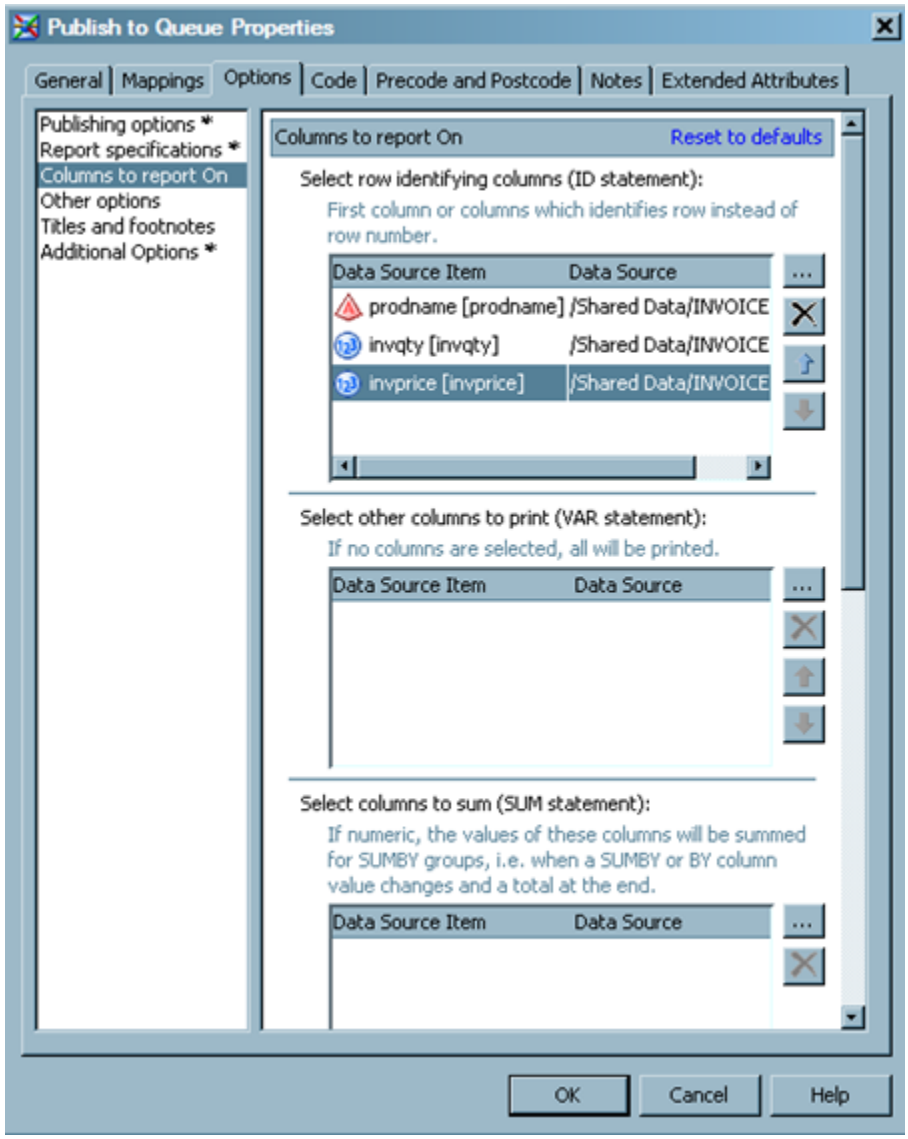
1. Open the properties window for the Publish to Queue transformation in the **Diagram** tab in the Job Editor window. Then, click the **Options** tab.
2. Click **Columns to report on** to access the Columns to report on columns page. Use the column selection prompts to access the columns that you need for your job. For example, you can click  for the **Select other columns to print (VAR statement)** to access the Select Data Source Items window, as shown in the following display.

Display A3.34 *Sample Select Data Source Items Window*



In the sample job, the VAR statement columns are prodname, invqty, and invprice. The column assignment options for the sample job are shown in the following display.

Display A3.35 *Sample Options Properties*



3. Set the publication options for the report on the Publishing options page. The options for the sample job are shown in the following display.

Display A3.36 *Sample Publishing Options*



Note that the option to specify a queue is required.

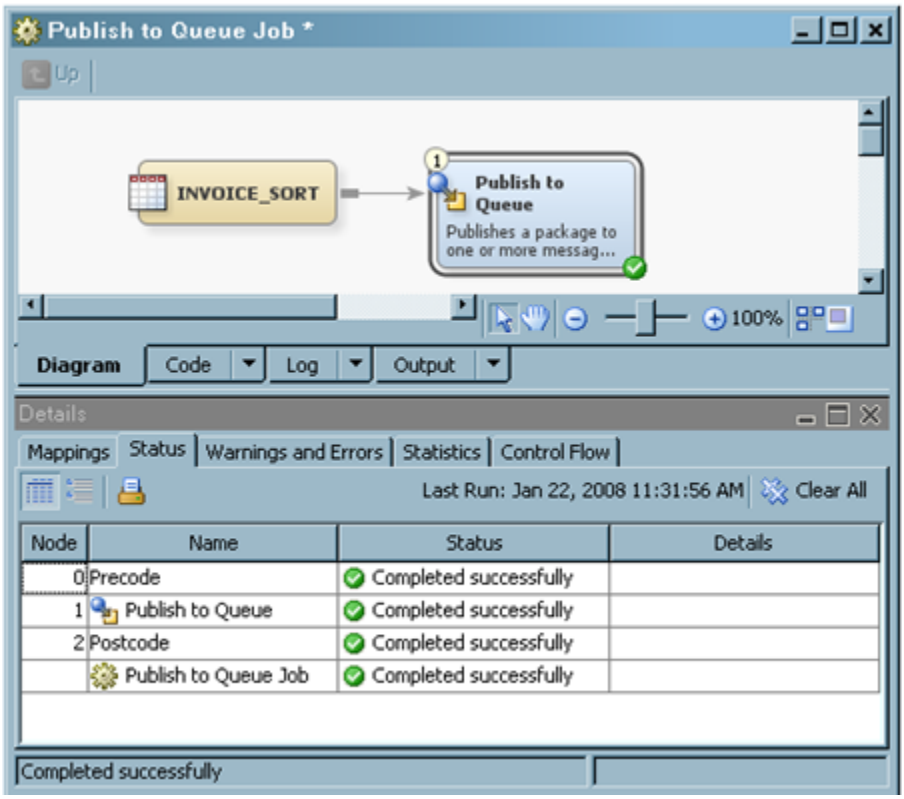
4. Set the remaining options for your report in the appropriate fields. For example, an appropriate path and filename for the report generated by the sample job must be specified in the **Specify path and filename for generated report** field on the Report specifications page. (Make sure that you click **Generate PROC PRINT from input table** in the **Generate PROC PRINTor send existing report** field). A title for the sample job report is set on the Titles and Footnotes page.

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 **Submit** in the pop-up menu. SAS Data Integration Studio generates code for the job and submits it to the SAS Application Server for execution. A completed sample job is shown in the following display.

Display A3.37 *Sample Completed Job*



2. If error messages display, read and respond to the messages as needed.
3. Click the **Output** tab. Your output should resemble the following display.

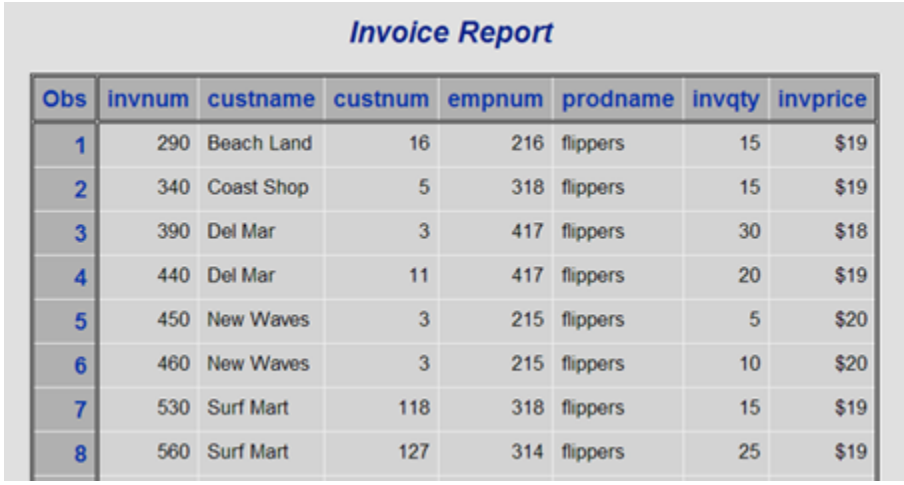
Display A3.38 *Sample Output in the Output Tab*

prodname	invqty	invprice	invnum	custname	custnum
flippers	15	$19	290	Beach Land	16
	15	$19	340	Coast Shop	5
	30	$18	390	Del Mar	3
	20	$19	440	Del Mar	11
	5	$20	450	New Waves	3
	10	$20	460	New Waves	3
	15	$19	530	Surf Mart	118
	25	$19	560	Surf Mart	127
kayak	3	$230	400	Del Mar	3

The packaged file (queue.spk) is also sent to the designated queue using MQSeries.

4. Navigate to the HTML file using the path that you configured on the Report specifications page on the **Options** tab. Sample HTML output is shown in the following display.

Display A3.39 *Sample HTML Output*



Invoice Report

Obs	invnum	custname	custnum	empnum	prodname	invqty	invprice
1	290	Beach Land	16	216	flippers	15	$19
2	340	Coast Shop	5	318	flippers	15	$19
3	390	Del Mar	3	417	flippers	30	$18
4	440	Del Mar	11	417	flippers	20	$19
5	450	New Waves	3	215	flippers	5	$20
6	460	New Waves	3	215	flippers	10	$20
7	530	Surf Mart	118	318	flippers	15	$19
8	560	Surf Mart	127	314	flippers	25	$19

Extracting Data from a Source Table

Overview

A SAS Extract transformation is a transformation that you can typically use to create one subset from a source. You can also use it to create columns in a target that are derived from columns in a source. For example, you can add a column to the target that concatenates two columns from the source or that calculates a value that is based on a column in the source.

Problem

You want to select a set of rows from a source table and write those rows to a target table.

Solution

You can use the Extract transformation in a SAS Data Integration Studio job to create jobs that require the data to be filtered or columns to be created from expressions. For example, you can create a job similar to the sample job featured in this topic. This sample job extracts only the rows that contain information about female employees from a table that contains information about both male and female employees. The sample job includes the following tasks:

- "Create and Populate the Job" on page 713
- "Specify Selection Conditions for the Target" on page 713
- "Run the Job and View the Output" on page 713

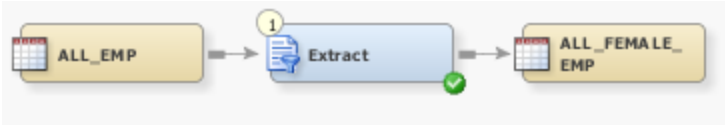
Tasks

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. Select and drag an Extract 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.
3. Select and drag the source table from the Inventory tree. Then, drop it before the Extract transformation on the **Diagram** tab.
4. Drag the cursor from the source table to the input port of the Extract transformation. This action connects the source to the transformation.
5. Because you want to have a permanent target table to contain the output for the transformation, 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.

Display A3.40 *Sample Process Flow*



The source table for the sample job is named ALL_EMP. The target table is named ALL_FEMALE_EMP. The preceding display shows the sample process flow.

Specify Selection Conditions for the Target

Use the tabs in the properties window for the Extract transformation to configure the output in the target table. Perform the following steps to configure the output:

1. Open the properties window for the Extract transformation on the **Diagram** tab in the Job Editor window. Then, click the **Where** tab.
2. Enter an appropriate WHERE condition in the **Expression Text** field. The following WHERE condition is entered in the sample job:

   ```
   Sex='F'
   ```

   For more information about using the **Where** tab, see "Creating or Configuring a WHERE Clause" on page 463.
3. Set the other criteria for your data extraction. The sample job uses the **Order by** tab to sort on the Age column in ascending order.
4. Click **OK** to save the selection criteria for the target table included in the job.

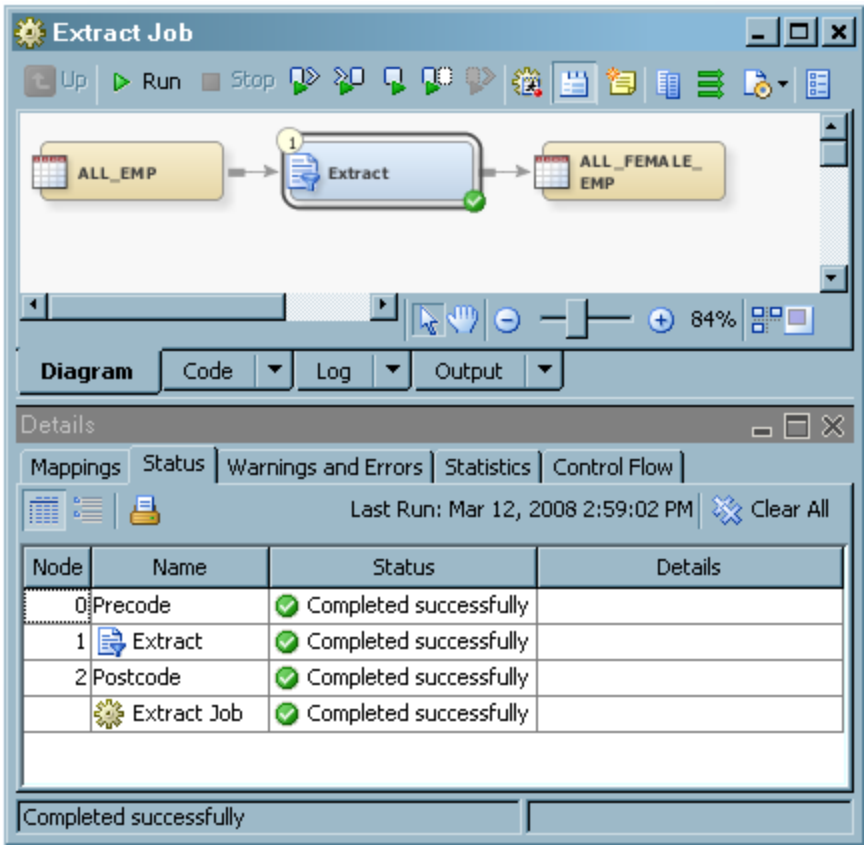
Run the Job and View the Output

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

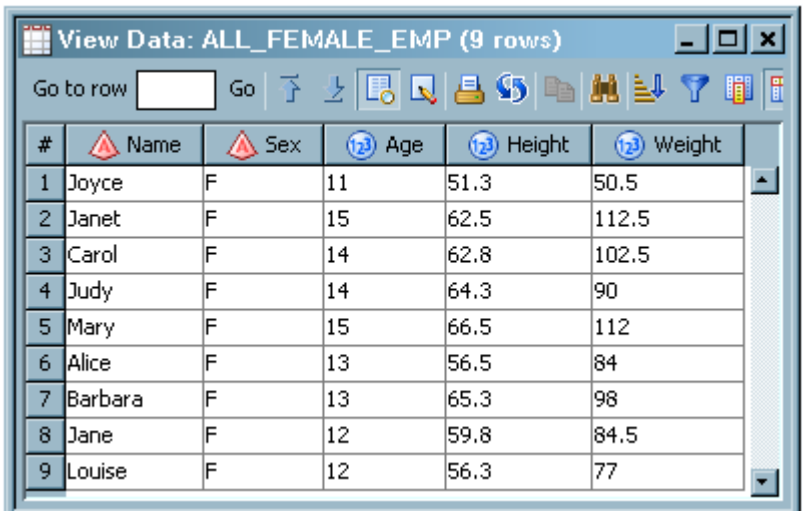
Application Server for execution. The following display shows a successful run of a sample job.

Display A3.41 *Sample Completed Job*



2. If error messages display, read and respond to the messages as needed.
3. 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 A3.42 *Sample Target Table in the View Data Window*



#	Name	Sex	Age	Height	Weight
1	Joyce	F	11	51.3	50.5
2	Janet	F	15	62.5	112.5
3	Carol	F	14	62.8	102.5
4	Judy	F	14	64.3	90
5	Mary	F	15	66.5	112
6	Alice	F	13	56.5	84
7	Barbara	F	13	65.3	98
8	Jane	F	12	59.8	84.5
9	Louise	F	12	56.3	77

Note that all of the employee dependents listed in the output are female. They are sorted by age in ascending order.

Creating Reports from Table Data

Overview

The List Data transformation provides an interface to the PRINT procedure. The PRINT procedure prints the observations in a SAS data set, using all or some of the variables. You can create a variety of reports ranging from a simple listing to a highly customized report that groups the data and calculates totals and subtotals for numeric variables. You can also use it to write the contents of a table (including a temporary output table) to a report. The PRINT procedure enables you to control many aspects of how the report is created, including the following:

- the title of the report
- how the observations in the report are grouped
- which columns are summed
- which columns are displayed and in what order

Generally, the List Data generated transformation comes at the end of a process flow diagram and prints data from the last table in the job. However, the transformation produces a temporary output table whose contents are identical to the contents of the input table, so you can use the transformation to create a report that is based on the temporary output table.

Problem

You want to print the data from a table in a report. For example, you can create a sort job and print the results in a PDF report.

Solution

You can use the List Data transformation as an interface to the PRINT procedure in a job that generates a report. For example, you can create a job similar to the sample job featured in this topic. This sample job sorts the data in a table that contains information about employees by sex and name. Note that the output for this job is sent to the **Output** tab in the Job Editor window and to an ODS document that is configured in the job. The sample job includes the following tasks:

- "Create and Populate the Job" on page 715
- "Configure Analytical Options" on page 716
- "Configure Reporting Options" on page 718
- "Run the Job and View the Output" on page 719

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 or open an existing job. For example, you can open a sort job.
2. Select and drag a List Data transformation from the Output folder in the Transformations tree. Then, drop it in the sort job on the **Diagram** tab in the Job Editor window.
3. Ensure that the output of the job can be sent to the **Output** tab of the Job Editor window. If the **Output** tab is not available, enable it by selecting **Tools Options ⇨ Show Output tab** in the menu bar.
4. Drag the cursor from the target table to the input port of the List Data transformation. This action connects the target to the transformation. The following display shows a sample process flow diagram for a job that contains the List Data transformation.

Display A3.43 *Sample Process Flow*



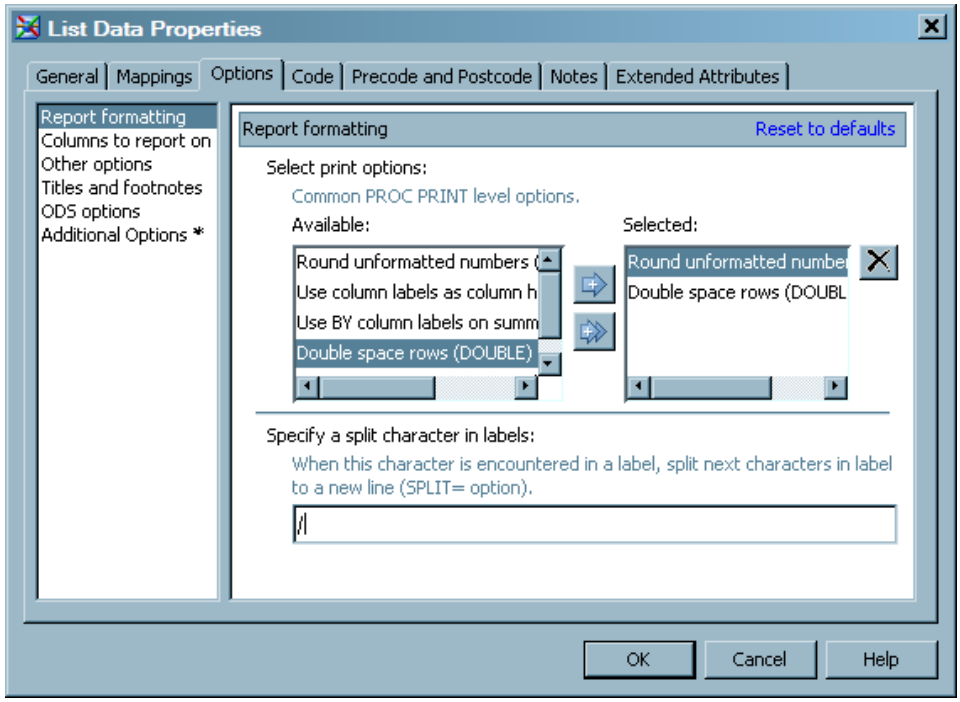
Note that the source table for the List Data transformation in the sample job is named ALL_FEMALE. This table is the target table for the Sort transformation.

Configure Analytical Options

Use the **Options** tab in the properties window for the List Data transformation to configure the output for your analysis. Note that the **Options** tab is divided into two parts, with a list of categories on the left side and the options for the selected category on the right side. Perform the following steps to set the options that you need for your job:

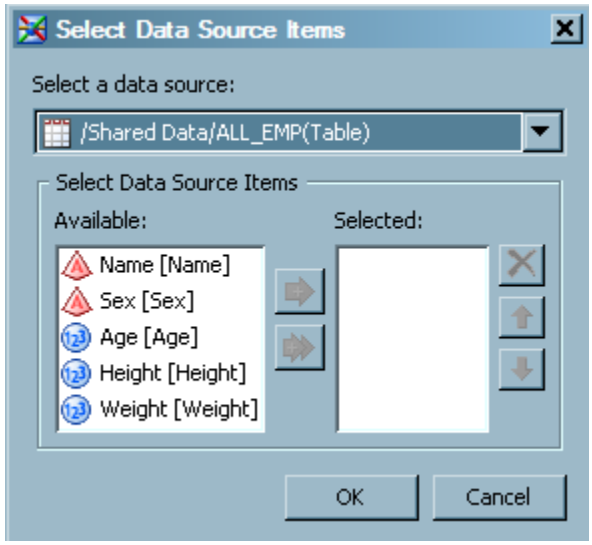
1. Open the properties window for the List Data transformation in the **Diagram** tab in the Job Editor window. Then, click the **Options** tab.
2. Click **Report formatting** to access the Report formatting page and select the formatting options for your report. For example, you can select the options shown in the following display:

Display A3.44 *Sample Report Formatting Options*



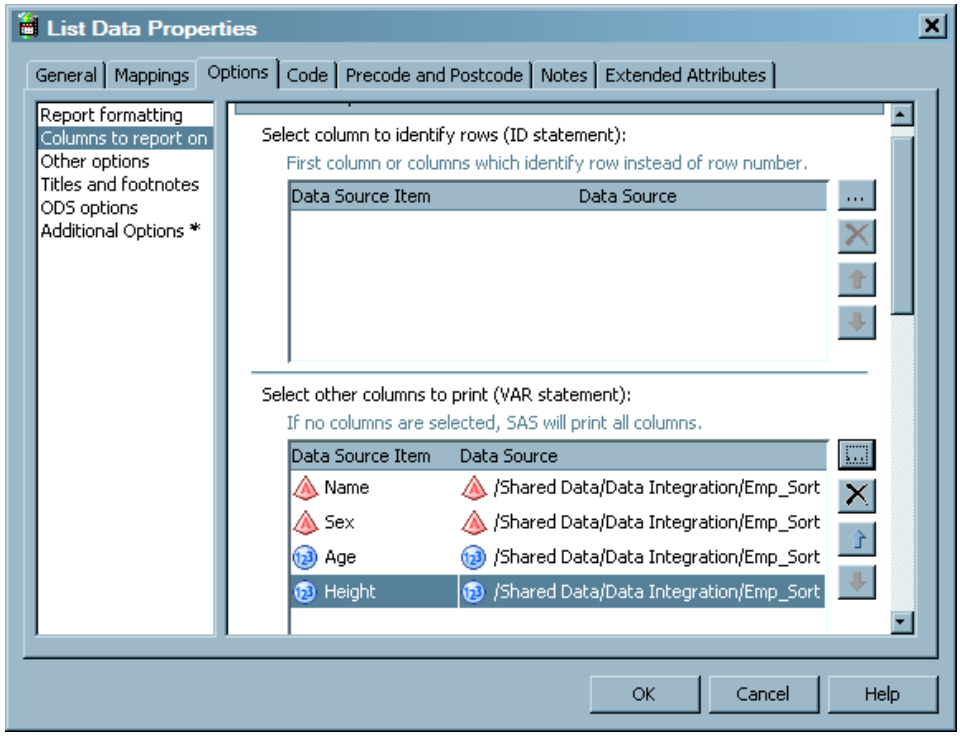
3. Click **Columns to report on** to access the Columns to report on page. Use the column selection prompts to access the columns that you need for your job. For example, you can click ... for the **Select other columns to print (VAR statement)** field to access the Select Data Source Items window, as shown in the following display:

Display A3.45 *Select Data Source Items Window*



Once you have selected the columns you need, the **Select other columns to print (VAR statement)** is populated, as shown in the following display:

Display A3.46 *Sample Selected Columns to Print*



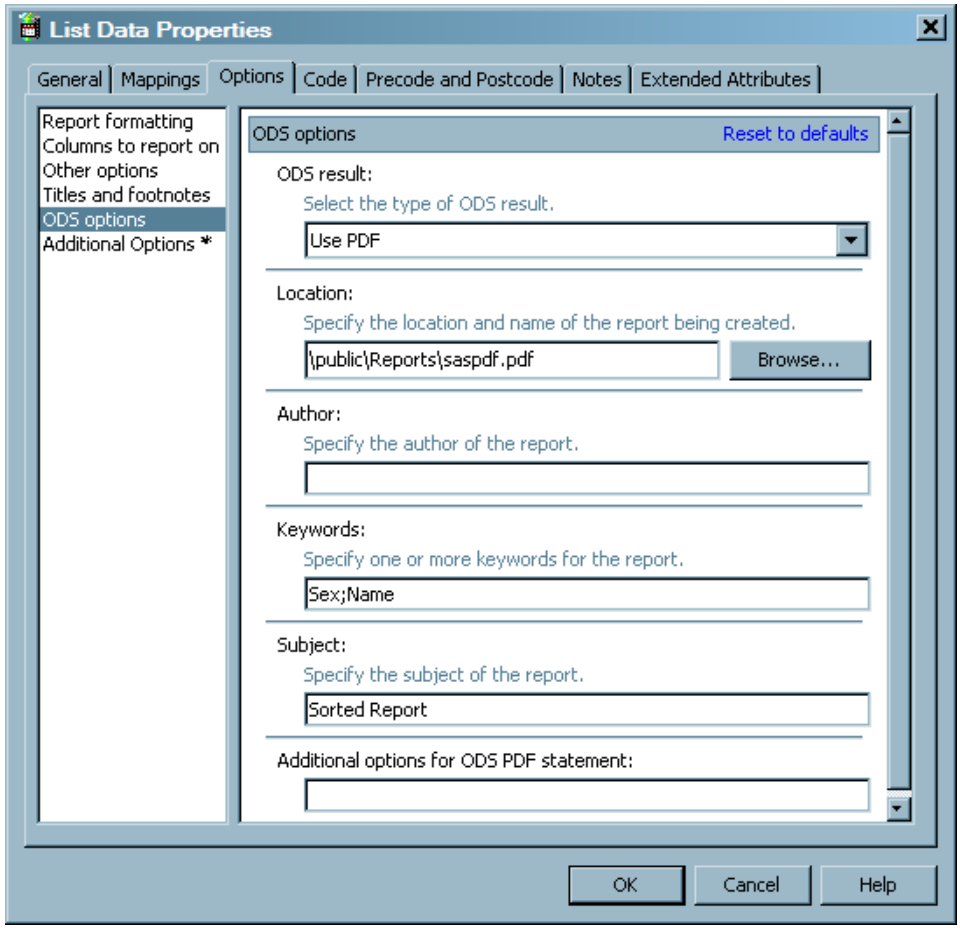
Note: You can specify additional PROC PRINT options and statements on the Other options page.

Configure Reporting Options

Use the remaining option pages to create and save an HTML, RTF, or PDF version of the output from the List Data transformation. Perform the following steps to set options for the document:

1. Click **ODS options** to access the ODS options page. You can choose between HTML, RTF, and PDF output and enter appropriate settings for each. The sample job uses PDF output. Therefore, a location, a set of keywords, and the subject of the report are added to the fields that are displayed when **Use PDF** is selected in the **ODS result** field. (The path specified in the **Location** field is relative to the SAS Application Server that executes the job.) These fields are shown in the following display:

Display A3.47 *Sample ODS Options*



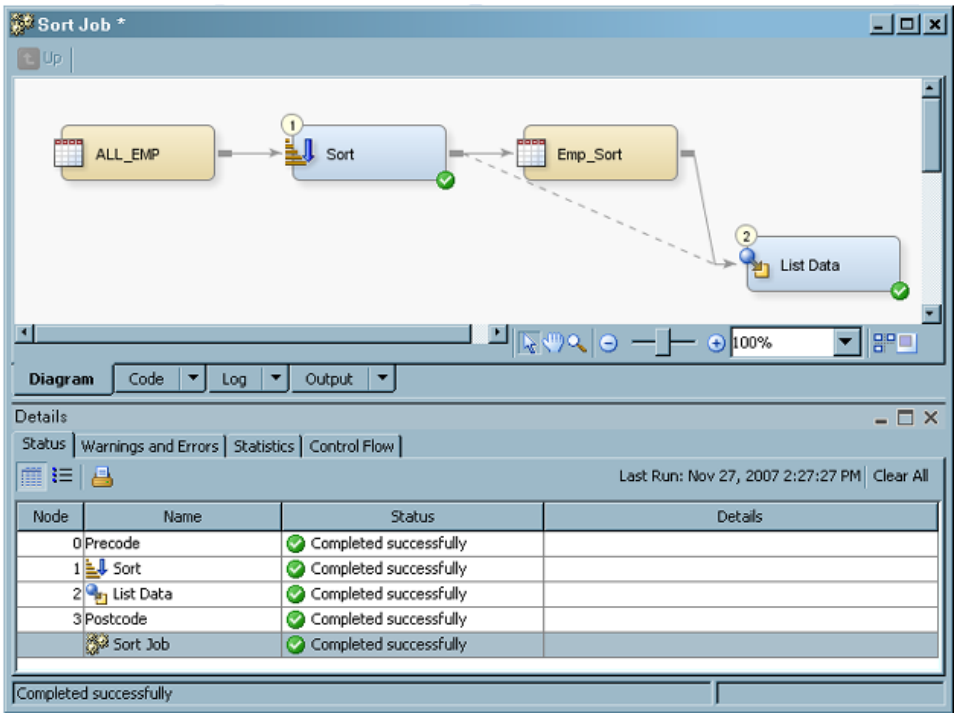
2. Click **OK** to save the settings for the **Options** tab.

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. The following display shows a successful run of a sample job:

Display A3.48 *Sample Completed Job*



2. If error messages display on the **Status** tab, read and respond to the messages as needed.
3. To view the data listing, click the **Output** tab in the Job Editor window.

Display A3.49 *Sample Output in the Output Tab*

```
                    The SAS System

 Obs    Name       Sex    Age    Height

   1    Alice       F      13     56.5

   2    Barbara     F      13     65.3

   3    Carol       F      14     62.8

   4    Jane        F      12     59.8

   5    Janet       F      15     62.5

   6    Joyce       F      11     51.3

   7    Judy        F      14     64.3

   8    Louise      F      12     56.3
```

4. Open the PDF document that you created and saved earlier. The following display illustrates a sample report based on the correlations data.

Display A3.50 *Sample PDF Output*

The SAS System

Obs	Name	Sex	Age	Height
1	Alice	F	13	56.5
2	Barbara	F	13	65.3
3	Carol	F	14	62.8
4	Jane	F	12	59.8
5	Janet	F	15	62.5
6	Joyce	F	11	51.3
7	Judy	F	14	64.3
8	Louise	F	12	56.3

Create a Table That Ranks the Contents of a Source

Overview

A Rank transformation uses the RANK procedure so you can rank one or more numeric variables in the source and store the ranks in the target.

Problem

You want to rank a set of numeric data according to some criteria. For example, you might want to rank a set of regional offices according to actual net profitability.

Solution

Create a job in which a Rank transformation reads numeric data, ranks that data according to some criteria, and then writes the ranked data to a target table. This sample job reads a table that contains profitability figures for a set of regional offices. Then it ranks them according to actual net profitability and writes the ranked data to a target table. The sample job includes the following tasks:

- "Create and Populate the Job" on page 722
- "Select Rank Variables" on page 722
- "Run the Job and View the Output" on page 723

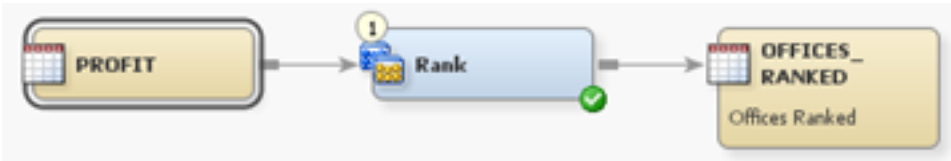
Tasks

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. Select and drag the Rank transformation from the **Data** folder in the Transformations tree. Drop it in the empty job on the **Diagram** tab in the Job Editor window.
3. From the Inventory tree, select and drag the source table. Then, drop it before the Rank transformation on the **Diagram** tab.
4. Drag the cursor from the source table to the input port of the Rank transformation. This action connects the transformation to the source.
5. A temporary work table appears after the Rank transformation. You can use this temporary table or delete it and add a target table. In this example, the temporary table is deleted.
6. From the Inventory tree, select and drag the target table. Then, drop it after the Rank transformation on the **Diagram** tab.
7. Drag the cursor from an output port of the Rank transformation to the input port of the target table. This action connects the transformation to the target. Note that if you have not deleted the temporary work table, you cannot make this connection. The following example shows the sample process flow.

Display A3.51 *Sample Process Flow*

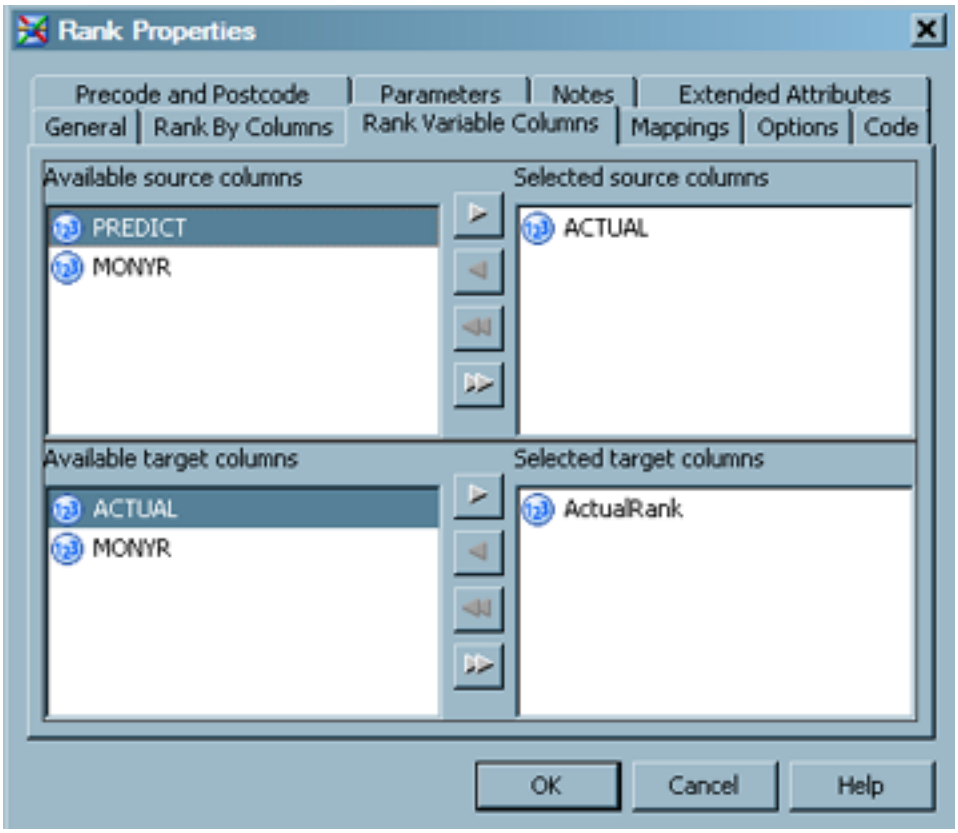


Select Rank Variables

Use the **Rank Variable Columns** tab in the properties window for the Rank transformation to specify how the information in the target table is to be ranked. The left pane on the **Rank Variable Columns** tab displays the column variables in the source. (For the rank to succeed, you must select the same number of columns for the target table as you did from the source table.) Perform the following steps to specify how the information in the target table is to be ranked:

1. Click the column that you want to select in the **Available source columns** field. Then move it to the **Selected source columns** field. The source column selected in the sample job is named ACTUAL.
2. Click the column that you want to select in the **Available target columns** field. Then move it to the **Selected target columns** field. The source column selected in the sample job is named ActualProfit.
3. Click **OK** to save the selection criteria for the target table and close the properties window for the Rank transformation. The following display shows the **Rank Variables Columns** tab for the sample job.

Display A3.52 *Rank Source and Target Information*

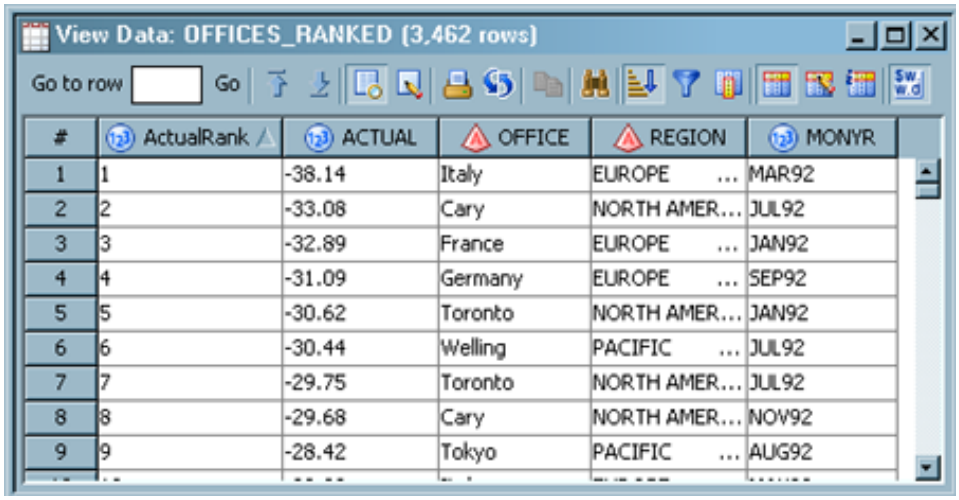


Run the Job and View the Output

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

1. Right-click on an empty area of the job, and click **Submit** 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.
3. 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 A3.53 *Data in Offices Ranked Table*



#	ActualRank	ACTUAL	OFFICE	REGION	MONYR
1	1	-38.14	Italy	EUROPE ...	MAR92
2	2	-33.08	Cary	NORTH AMER...	JUL92
3	3	-32.89	France	EUROPE ...	JAN92
4	4	-31.09	Germany	EUROPE ...	SEP92
5	5	-30.62	Toronto	NORTH AMER...	JAN92
6	6	-30.44	Welling	PACIFIC ...	JUL92
7	7	-29.75	Toronto	NORTH AMER...	JUL92
8	8	-29.68	Cary	NORTH AMER...	NOV92
9	9	-28.42	Tokyo	PACIFIC ...	AUG92

Create Two Tables That Are Subsets of a Source

Overview

A Splitter transformation is a transformation that creates one or more subsets of a source. You can also use it to create one or more copies of a source.

Problem

You want to select two or more sets of rows from a source table and write each set to a different target table.

Solution

You can use the SAS Splitter transformation in a SAS Data Integration Studio job to support 1-N outputs and one input. For example, you can create a job similar to the sample job featured in this topic. This sample job splits a source table that contains employee data into two target tables: one for female employees and another for male employees. The sample job includes the following tasks:

- "Create and Populate the Job" on page 724
- "Specify Selection Conditions for the Target Tables" on page 725
- "Run the Job and View the Output" on page 726

Tasks

Create and Populate the Job

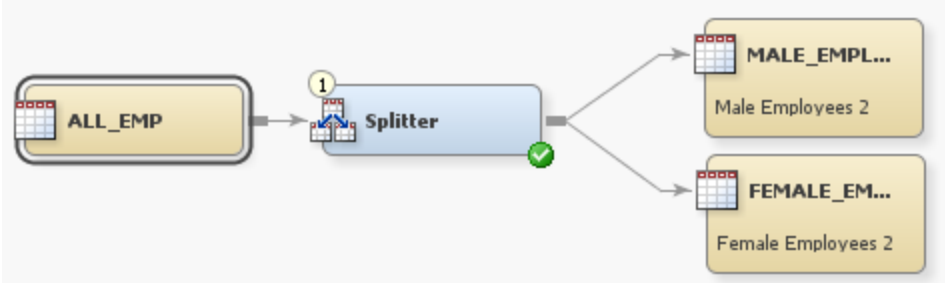
Perform the following steps to create and populate a new job:

1. Create an empty SAS Data Integration Studio job.

2. From the **Data** folder in the Transformations tree, select and drag a Splitter 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 Splitter transformation on the **Diagram** tab.
4. Drag the cursor from the source table to the input port of the Splitter transformation. This action connects the transformation to the source.
5. Because you want to have permanent target tables to contain the output for the transformation, right-click each of the temporary work tables attached to the transformation and click **Replace** in the pop-up menu. Then, use the Table Selector window to select the target tables for the job. The target tables must be registered in SAS Data Integration Studio.

The following display shows the sample process flow.

Display A3.54 *Splitter Process Flow Diagram*



In the display, the source table is named ALL_EMP and the permanent target tables are named Female Employees 2 and Male Employees 2.

Specify Selection Conditions for the Target Tables

Use the **Row Selection** tab in the properties window for the Splitter transformation to specify the selection conditions for the two target tables. Perform the following steps:

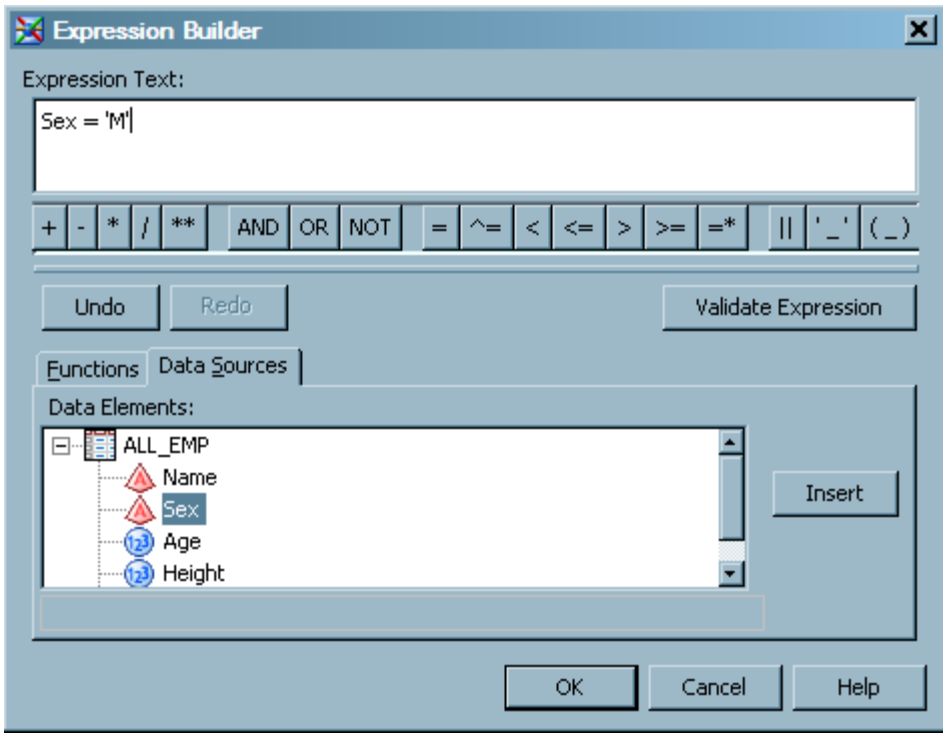
1. In the Job Editor, open the properties window for the Splitter transformation. Then, click the **Row Selection** tab. The **Target Tables** field displays the targets that have been dropped on the current Splitter transformation. (Male Employees 2 and Female Employees 2 are the target tables in the sample job.) You must define unique selection criteria for each target.
2. Click the name of the first target table in the **Target Tables** field. The first target table for the sample job is Male Employees 2.
3. Select the **Row Selection Conditions** in the **Row Selection Type** field. Note that the **Subset Data** button is activated. Click **Subset Data** to display the Expression Builder window.
4. Define the row selection criteria for the selected target table (Male Employees 2 in the sample job). You can either enter the selection criteria directly in the **Expression Text** field, or you can use the tools available on the **Functions** and **Data Sources** tabs. The selection condition for the Male Employees 2 table is

   ```
   SEX='M'
   ```

   .

The following display shows the completed **Row Selection** tab for the first target table in the sample job.

Display A3.55 *Row Selection Tab with Male Employees 2 Selection Criteria*



5. Perform the row selection conditions for the remaining target tables. In the sample job, the remaining target table is named Female Employees 2.
6. Click **OK** to save the selection criteria for the target tables and close the properties window for the Splitter transformation.

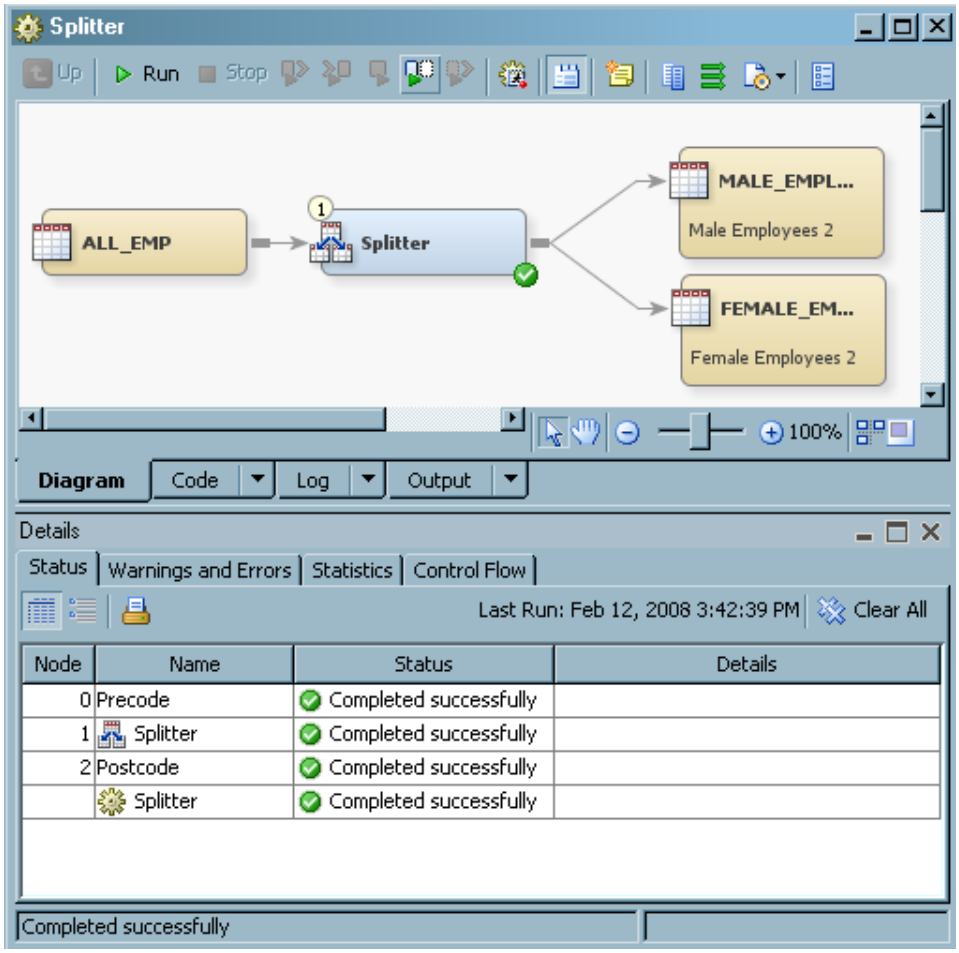
You have defined a job that selects rows for female employees and writes the rows to the target table Female Employees 2. The job also selects rows for male employees and writes the rows to the target table Male Employees 2.

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. The following display shows a successful run of a sample job.

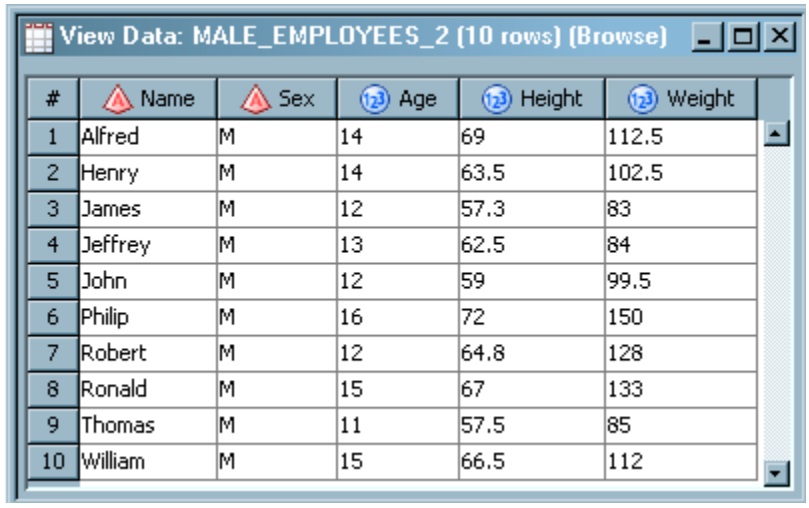
Display A3.56 *Successfully Completed Sample Job*



2. If error messages are displayed on the **Status** tab, read and respond to the messages as needed.

3. To view the target table, right-click the target and select **Open**. The following display shows the data for the Male Employees 2 target table in the sample job.

Display A3.57 *Male Employees 2 Target Table Data*



#	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69	112.5
2	Henry	M	14	63.5	102.5
3	James	M	12	57.3	83
4	Jeffrey	M	13	62.5	84
5	John	M	12	59	99.5
6	Philip	M	16	72	150
7	Robert	M	12	64.8	128
8	Ronald	M	15	67	133
9	Thomas	M	11	57.5	85
10	William	M	15	66.5	112

Moving Data Directly from One Machine to Another Machine

Overview

A Data Transfer transformation is a transformation that you can use 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

By default, SAS Data Integration Studio generates code that moves the source data from one machine (machine 1) to another (machine 2). Then, it moves the data from machine 2 to the target table on a third machine (machine 3). This action is an implicit data transfer. For large amounts of data, an implicit data transfer might not be the most efficient way to transfer data.

The following display shows the icon that is displayed on the affected transformation when implicit data transfer is used:

Display A3.58 *Implicit Data Transfer Icon*



To improve efficiency, you can add a Data Transfer transformation to the process flow diagram. The transformation enables SAS Data Integration Studio to generate code that migrates data directly from the source machine to the target machine. You can use the Data Transfer transformation with a SAS table or a DBMS table with table and column names that follow the rules for SAS names.

Problem

You need to move data directly from a source table on one machine to a target table on another machine.

Solution

You can use the Data Transfer transformation to perform a direct data transfer between source and target tables that reside on different machines. Perform the following tasks to transfer the data directly from Machine 1 to Machine 3:

- "Create and Populate the Job" on page 729
- "Verify the Storage Location for the Target Table" on page 729
- "Run the Job" on page 730
- "Verify the Result" on page 731

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. Select and drag a Data Transfer 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 the source table from the Inventory tree. Then, drop it before the Data Transfer transformation on the **Diagram** tab.
4. Drag the cursor from the source table to the input port of the Data Transfer transformation. This action connects the source to the transformation.
5. Because you want to have a permanent target table to contain the output for the transformation, 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.The following display shows a sample process flow diagram for a job that contains the Data Transfer transformation.

   Display A3.59 *Sample Data Transfer Process Flow*

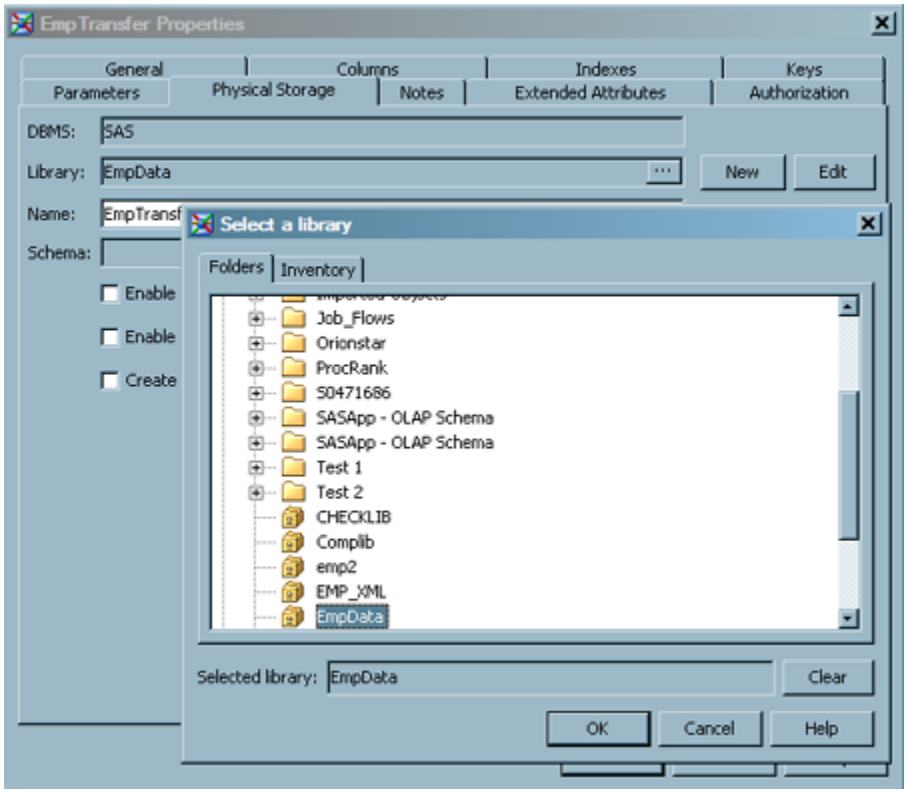
   Note that the source table for a sample job is named EMPLOYEE. It is stored on Machine 1 and registered in a current metadata repository. The target table is named EmpTransfer. It is stored in the EmpData library on Machine 3.

Verify the Storage Location for the Target Table

Perform the following steps to verify that the target table (EmpTransfer) is stored in the library on the remote machine (Machine 3):

1. Open the properties window for the target table. Then, click the **Physical Storage** tab.
2. Select the library for the target table in the **Library** field by clicking on the  to open the Select a library window.
3. The Select a library window appears with the library highlighted. The following display shows a sample Select a library window.

Display A3.60 *Sample Select a library Window*



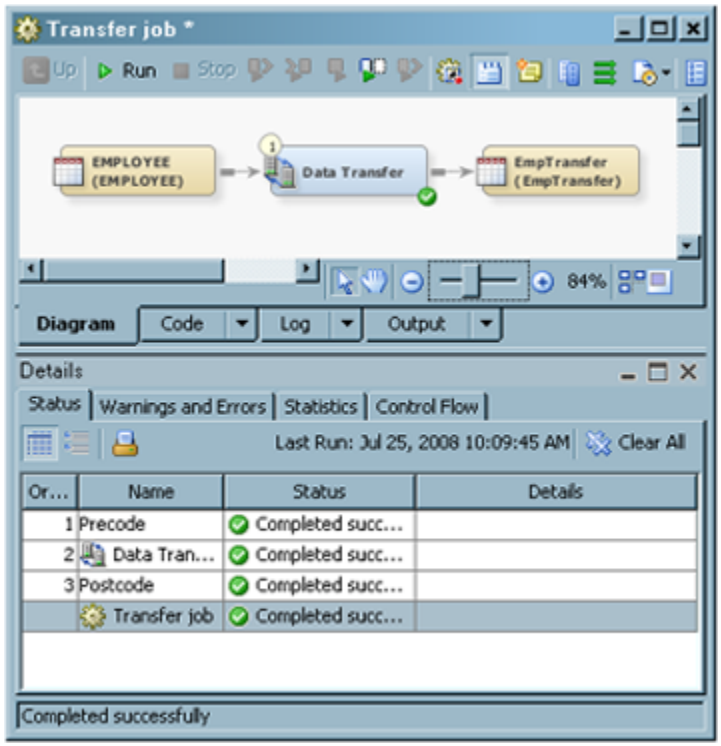
In the sample job, this library is named EmpData. The target table is stored on Machine 3.

Run the Job

Perform the following steps to run the job:

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. The following display shows a successful run of the sample job.

Display A3.61 *Sample Completed Data Transfer Job*



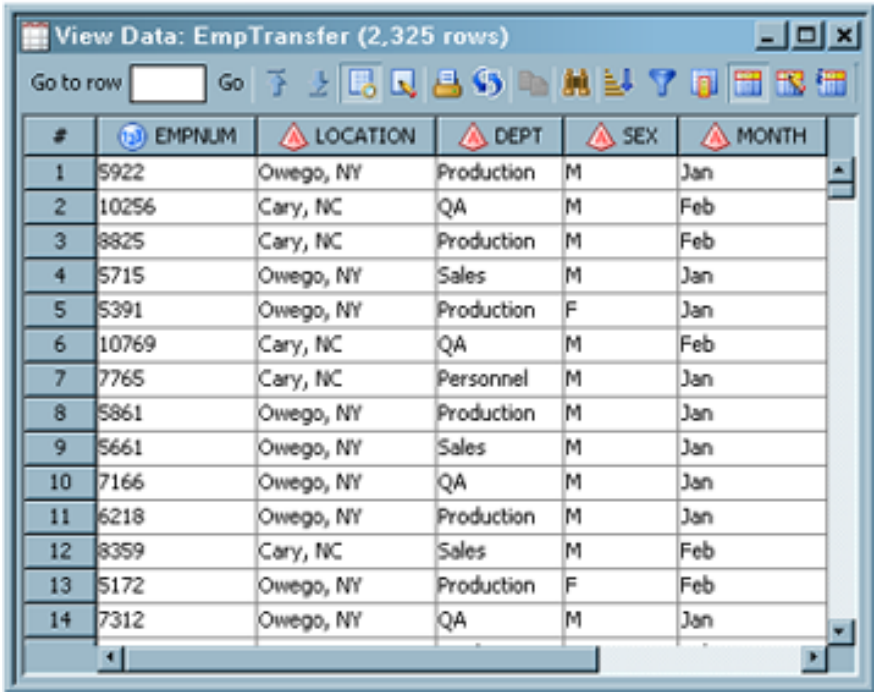
2. If the job completes without error, go to the next task. If an error messages appear, read and respond to the messages.

Verify the Result

Perform the following to verify that the job created the desired output:

1. Right-click the target table on the **Diagram** tab. Then, click **Open** in the pop-up menu. The following display shows the target table data for the sample job.

Display A3.62 *Sample Target Table in the View Data Window*



#	EMPNUM	LOCATION	DEPT	SEX	MONTH
1	5922	Owego, NY	Production	M	Jan
2	10256	Cary, NC	QA	M	Feb
3	8825	Cary, NC	Production	M	Feb
4	5715	Owego, NY	Sales	M	Jan
5	5391	Owego, NY	Production	F	Jan
6	10769	Cary, NC	QA	M	Feb
7	7765	Cary, NC	Personnel	M	Jan
8	5861	Owego, NY	Production	M	Jan
9	5661	Owego, NY	Sales	M	Jan
10	7166	Owego, NY	QA	M	Jan
11	6218	Owego, NY	Production	M	Jan
12	8359	Cary, NC	Sales	M	Feb
13	5172	Owego, NY	Production	F	Feb
14	7312	Owego, NY	QA	M	Jan

2. Compare the data in the target table to the data in the source table. If the data matches, the data transfer is successful.

Creating Standardized Statistics from Table Data

Overview

The Standardize transformation is an interface to the STANDARD procedure. The STANDARD procedure standardizes variables in a SAS data set to a given mean and standard deviation, and it creates a new SAS data set containing the standardized values. You can use it to create a target table that contains standardized data. You can control many aspects of how the target table is created, which can include the following:

- the type of standardization
- which columns are analyzed

Problem

You want to generate a target table that is standardized to a given mean and standard deviation.

Solution

You can use the Standardize transformation in a SAS Data Integration Studio job. This transformation uses the STANDARD procedure to standardize variables in a SAS data

set to a given mean and standard deviation. Then, it creates a new SAS data set that contains the standardized values.

For example, you can create a job similar to the sample job featured in this topic. This sample job generates a standardized analysis based on a table that contains information about test scores. Note that the output for this job is sent to a target table. The sample job includes the following tasks:

- "Create and Populate the Job" on page 733
- "Configure Analytical Options" on page 734
- "Run the Job and View the Output" on page 734

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. Select and drag a Standardize 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.
3. Select and drag the source table out of the Inventory tree. Then, drop it before the Standardize transformation on the **Diagram** tab.
4. Drag the cursor from the source table to the input port of the Standardize transformation. This action connects the source to the transformation.
5. Right-click the Standardize transformation, and click **Add Output Port** from the **Ports** option in the drop-down menu. This step enables you to add an output port to the transformation.
6. Because you want to have a permanent target table to contain the output for the transformation, 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.

The following display shows a sample process flow diagram for a job that contains the Standardize transformation:

Display A3.63 *Sample Process Flow*



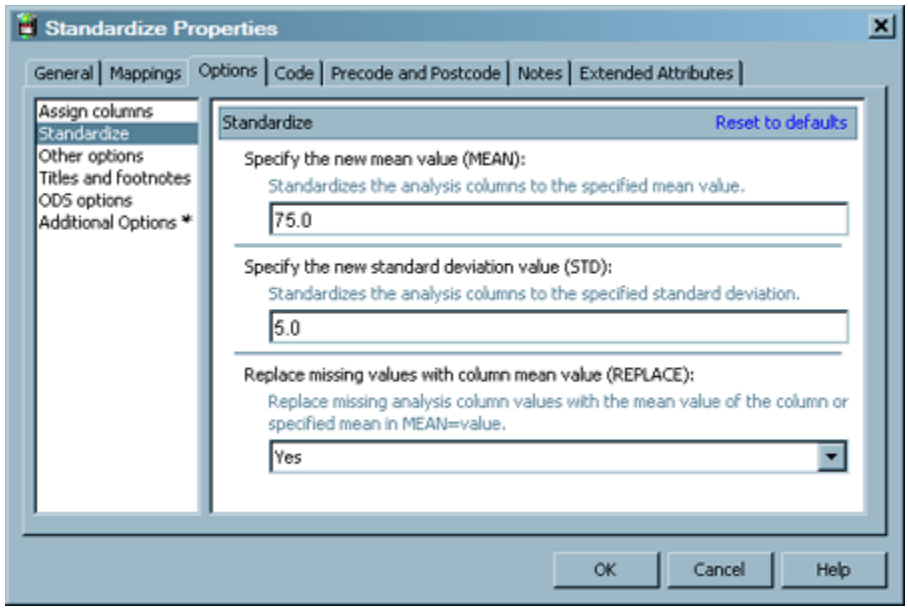
Note that the source table for the sample job is named SCORE, and the target table is named SCORE_OUT.

Configure Analytical Options

Use the **Options** tab in the properties window for the Standardize transformation to configure the output for your analysis. Note that the **Options** tab is divided into two parts, with a list of categories on the left-hand side and the options for the selected category on the right-hand side. Perform the following steps to set the options that you need for your job:

1. Open the properties window for the Correlations transformation in the **Diagram** tab in the Job Editor window. Then, click the **Options** tab.
2. Click **Standardize** to access the Standardize page. In the sample job, the mean value and the standard deviation are set in the Standardize page. You can also specify that missing values are replaced with the column mean value on the same page. These settings are shown in the following display:

Display A3.64 *Sample Standardize Options*

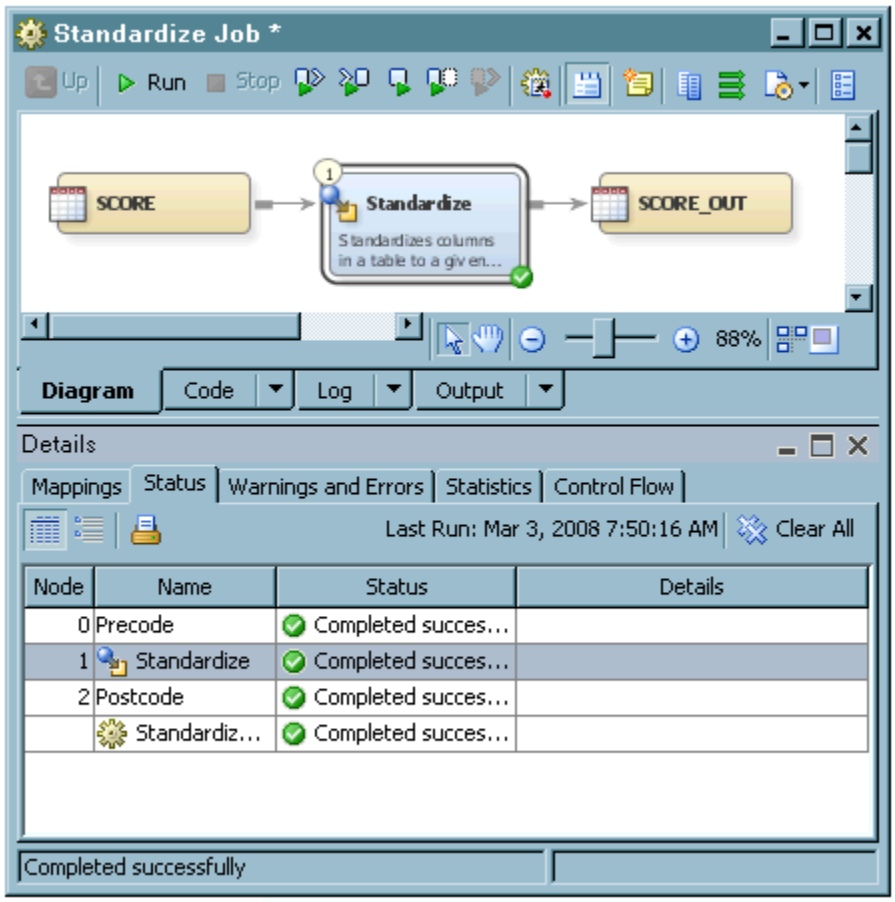


Run the Job and View the Output

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

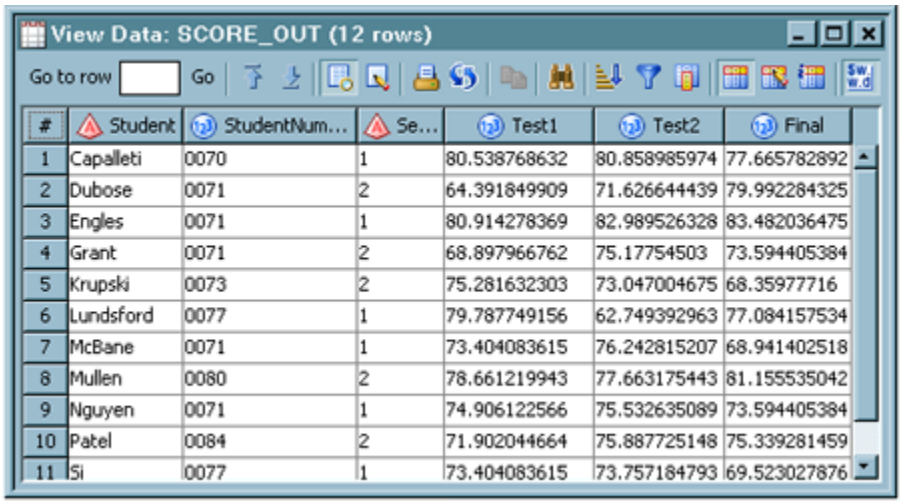
1. Right-click on an empty area of the job, and click **Run** in the pop-up menu. SAS Data Integration Studio generates code for the job and submits it to the SAS Application Server for execution. The following display shows a successful run of a sample job:

Display A3.65 *Successfully Completed Sample Job*



2. If error messages display, read and respond to the messages as needed.
3. 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 A3.66 *Output for Standardize Transformation*



#	Student	StudentNum...	Se...	Test1	Test2	Final
1	Capalleti	0070	1	80.538768632	80.858985974	77.665782892
2	Dubose	0071	2	64.391849909	71.626644439	79.992284325
3	Engles	0071	1	80.914278369	82.989526328	83.482036475
4	Grant	0071	2	68.897966762	75.17754503	73.594405384
5	Krupski	0073	2	75.281632303	73.047004675	68.35977716
6	Lundsford	0077	1	79.787749156	62.749392963	77.084157534
7	McBane	0071	1	73.404083615	76.242815207	68.941402518
8	Mullen	0080	2	78.661219943	77.663175443	81.155535042
9	Nguyen	0071	1	74.906122566	75.532635089	73.594405384
10	Patel	0084	2	71.902044664	75.887725148	75.339281459
11	Si	0077	1	73.404083615	73.757184793	69.523027876

Creating Transposed Data from Table Data

Overview

Use a Transpose transformation to create a target table that contains transposed data. You can control many aspects of how the target table is created, including the following:

- the type of data to be transposed
- which columns are analyzed

The TRANSPOSE procedure creates an output data set by restructuring the values in a SAS data set, transposing selected variables into observations. The TRANSPOSE procedure can often eliminate the need to write a lengthy DATA step to achieve the same result. Further, the output data set can be used in subsequent DATA or PROC steps for analysis, reporting, or further data manipulation. A transposed variable is a variable that the procedure creates by transposing the values of an observation in the input data set into values of a variable in the output data set.

PROC TRANSPOSE does not produce printed output. To print the output data set from the PROC TRANSPOSE step, use the List Data transformation or another SAS publishing, summary, or reporting tool.

The target table that is created by this transformation contains only the columns selected to be analyzed. In addition, there is an informational column, _NAME_, which contains the name of the column that is to be used as the transposed column. There are also additional columns needed, depending on the type of transposition you are performing.

Note: You should set **Update the table metadata for the target tables** to **Yes**. This action ensures that the proper columns are included in the target table. Alternatively, you can create the columns in the output table on the **Mapping** tab of the Transpose transformation property window. You must create the _NAME_ column and any other columns in the target table for that data to be included in the final table.

Problem

You want to create a target table that contains transposed data.

Solution

You can use the Transpose transformation in a job that transposes the data in a table and creates an ODS document that displays the transposed table. The transformation uses the TRANSPOSE procedure to load transposed data into a target table. Transposing the data in a table turns the rows in a table into columns and the columns into the rows.

For example, you can create a job similar to the sample job featured in this topic. This sample job generates a target table that contains information about baking. The values in the Age column are transposed into columns in the SAS column generated in the job. This data is then output to the target table and to a report that is generated with ODS. The sample job includes the following tasks:

- "Create and Populate the Job" on page 737
- "Configure Analytical Options" on page 737

- "Run the Job and View the Output" on page 739

Tasks

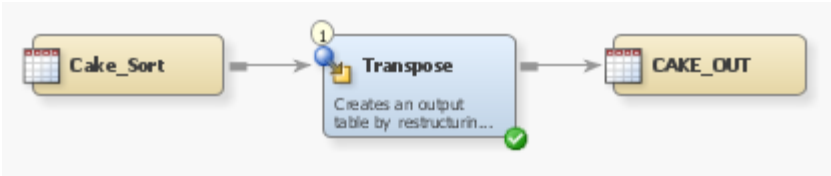
Create and Populate the Job

Perform the following steps to create and populate a new job:

1. Create an empty SAS Data Integration Studio job.
2. Select and drag a Transpose 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.
3. Select and drag the source table out of the Inventory tree. Then, drop it before the Transpose transformation on the **Diagram** tab.
4. Drag the cursor from the source table to the input port of the Transpose transformation. This action connects the source to the transformation.
5. Right-click the Transpose transformation, and click **Add Output Port** from the **Ports** option in the drop-down menu. This step enables you to add an output port to the transformation.
6. Because you want to have a permanent target table to contain the output for the transformation, 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.

The following display shows a sample process flow diagram for a job that contains the Transpose transformation:

Display A3.67 *Sample Process Flow*



Note that the source table for the sample job is named Cake_Sort and that the target table is named CAKE_OUT.

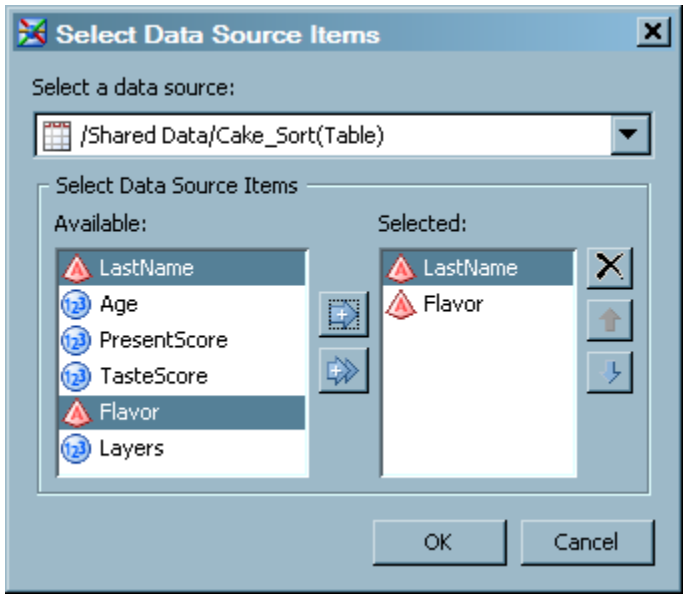
Configure Analytical Options

Use the **Options** tab in the properties window for the Transpose transformation to configure the SAS tables that are generated in the job and shape the output of your analysis. Note that the **Options** tab is divided into two parts, with a list of categories on the left side and the options for the selected category on the right side. Perform the following steps to set the options that you need for your job:

1. Open the properties window for the Transpose transformation in the **Diagram** tab in the Job Editor window. Then, click the **Options** tab.
2. Click **Assign columns** to access the Assign columns page. Use the column selection prompts to access the columns that you need in the SAS tables generated in your job.

For example, you can click ... for the **Select analysis columns (VAR statement)** to access the Select Data Source Items window, as shown in the following display:

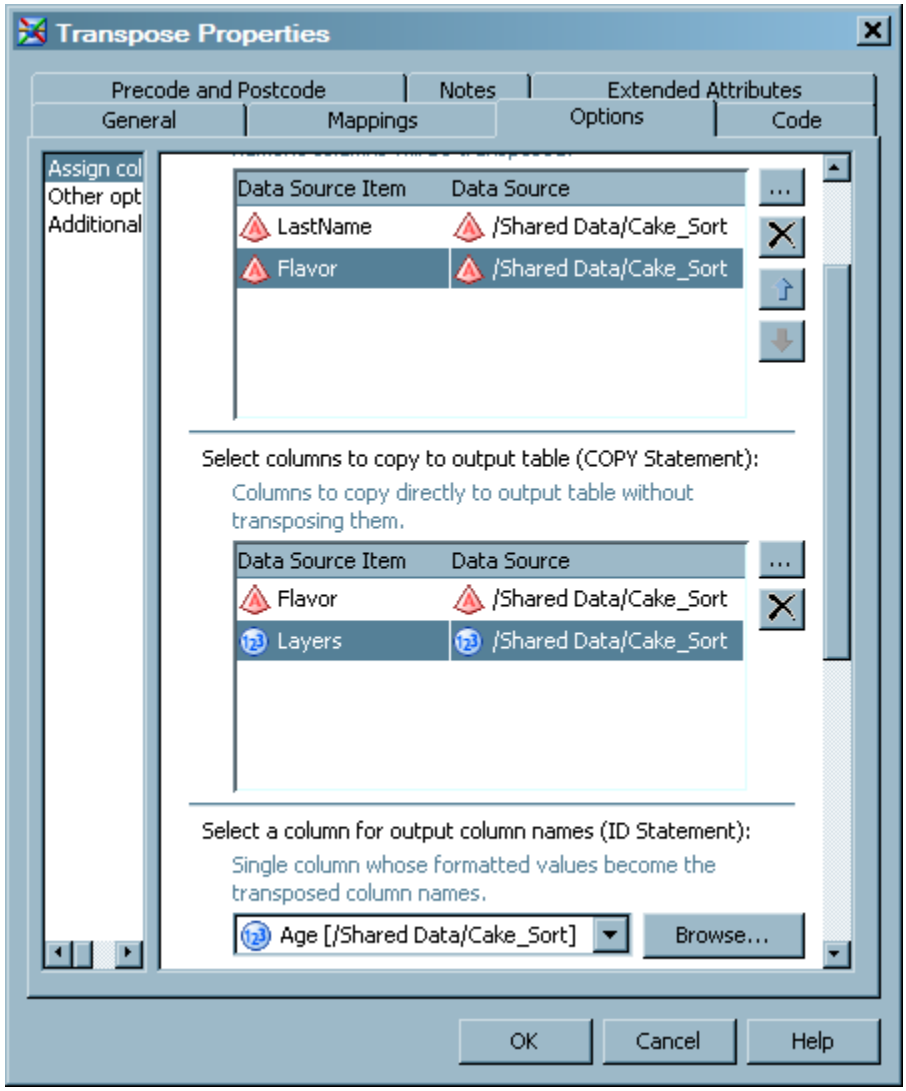
Display A3.68 *Sample Select Data Source Items Window*



All of the values in the rows of the columns that you select in this window become values in a single row after the transposition is completed. In the sample job, the VAR statement columns are LastName and Flavor.

3. Specify additional columns as needed. For example, the COPY statement in the sample job includes the Flavor and Layers columns, and the ID statement includes the Age column. The following display shows the columns that are specified for the sample job:

Display A3.69 *Sample Column Options*



4. Click **Additional Options** to access the Additional Options page. Make sure that the **Update the metadata for the target tables** option is set to **Yes**. This step ensures that the target tables contain the columns needed for the Transpose transformation to run successfully.

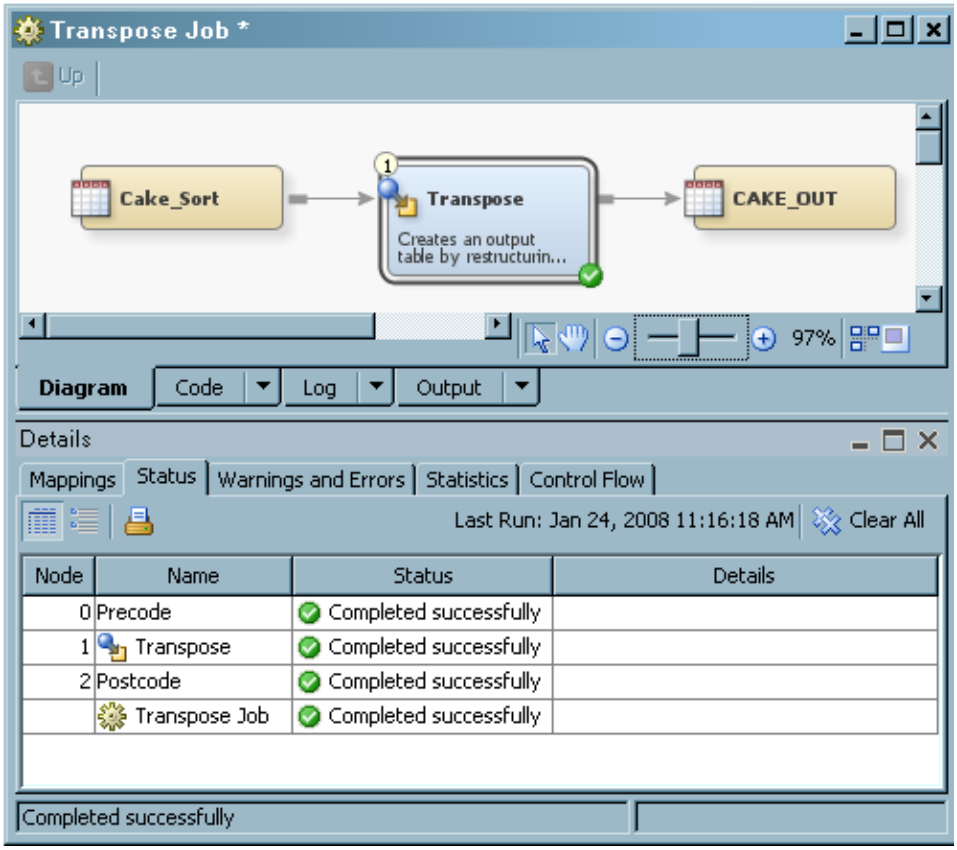
5. Set remaining options as needed. For example, the sample job specifies system options, TRANSPOSE procedure options, and metadata update options.

Run the Job and View the Output

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

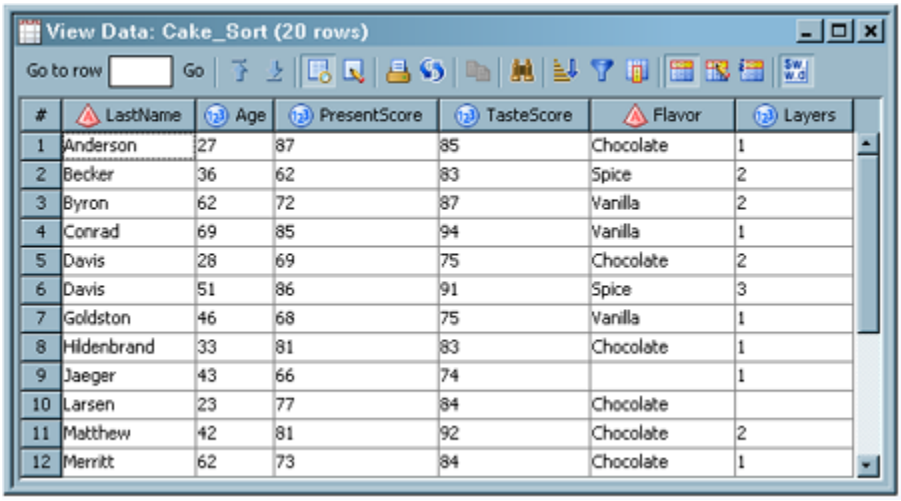
1. Right-click on an empty area of the job, and click **Run** in the pop-up menu. SAS Data Integration Studio generates code for the job and submits it to the SAS Application Server for execution. The following display shows a successful run of a sample job:

Display A3.70 *Sample Completed Job*



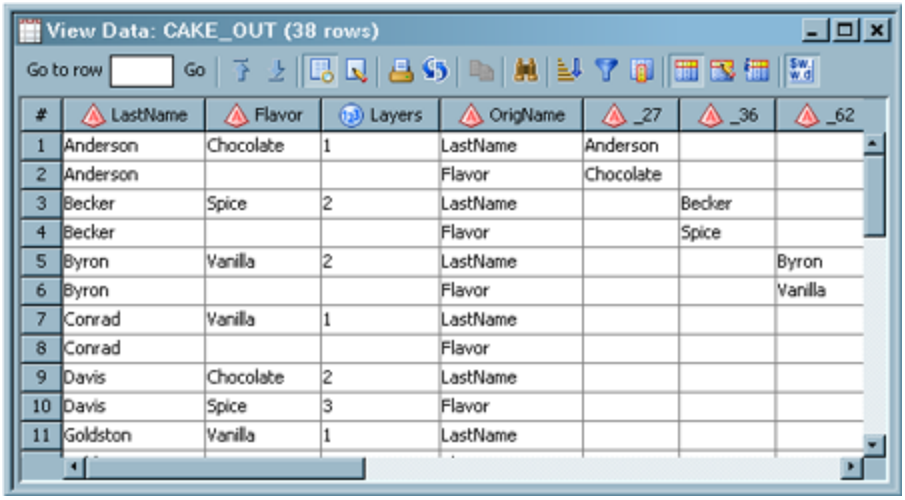
2. If error messages display, read and respond to the messages as needed.
3. You can compare the source table to the target table to see the results of the TRANSPOSE procedure. The source table for the sample job, Cake_Sort, is shown in the following display:

Display A3.71 *Source Table in the View Data Window*



#	LastName	Age	PresentScore	TasteScore	Flavor	Layers
1	Anderson	27	87	85	Chocolate	1
2	Becker	36	62	83	Spice	2
3	Byron	62	72	87	Vanilla	2
4	Conrad	69	85	94	Vanilla	1
5	Davis	28	69	75	Chocolate	2
6	Davis	51	86	91	Spice	3
7	Goldston	46	68	75	Vanilla	1
8	Hildenbrand	33	81	83	Chocolate	1
9	Jaeger	43	66	74		1
10	Larsen	23	77	84	Chocolate	
11	Matthew	42	81	92	Chocolate	2
12	Merritt	62	73	84	Chocolate	1

The target table for the sample job, CAKE_OUT, is shown in the following display:


View Data: CAKE_OUT (38 rows)

Go to row Go

#	LastName	Flavor	Layers	OrigName	_27	_36	_62
1	Anderson	Chocolate	1	LastName	Anderson		
2	Anderson			Flavor	Chocolate		
3	Becker	Spice	2	LastName		Becker	
4	Becker			Flavor		Spice	
5	Byron	Vanilla	2	LastName			Byron
6	Byron			Flavor			Vanilla
7	Conrad	Vanilla	1	LastName			
8	Conrad			Flavor			
9	Davis	Chocolate	2	LastName			
10	Davis	Spice	3	Flavor			
11	Goldston	Vanilla	1	LastName			

Note that the values in the rows in the Age column in the source table have been transposed into columns in the target table (such as _27, _36, and _62).

Converting a SAS or DBMS Table to an XML Table

Overview

You can use the XML Writer transformation to convert almost any data source to an XML file. For example, you can convert data in a SAS data set (SAS proprietary format) to a more generic XML format. You can also convert data from any source that SAS can access, such as a text file or a DBMS table. The XML file that is output by the writer can be generic or of a specific type (Oracle, MSAccess, and so on). The XML file is easy to share and can easily be read by any third-party software.

The target of an XML Writer must be an XML table in a SAS XML Library. Access to the library must not be set to READONLY.

Problem

You want to convert a SAS or DBMS table to an XML table in order to use the information in a third-party application.

Solution

You can create a job or update an existing job so that the SAS or DBMS table is the input to an XML Writer transformation, and an XML file is the output for the transformation. The process flow for the job would be similar to the flow in the next display.

Display A3.72 *Sample Process Flow for an XML Writer Job*



In the sample flow, EMPLOYEES... (EMPLOYEES_SORTED) is an input table in SAS or DBMS format. The XML Writer transformation reads the input table and writes its content to Xsort_emp, an XML table. Assume that the SAS or DBMS input table exists in physical storage, and that the XML target table does not yet exist. The target table is created when the job is executed.

To create an XML Writer job, perform the following tasks:

- "Register an XML Library for the XML Target Table" on page 742
- "Register the XML Target Table" on page 743
- "Create and Populate the XML Writer Job" on page 744
- "Run the Job and Verify the Results" on page 745

Tasks

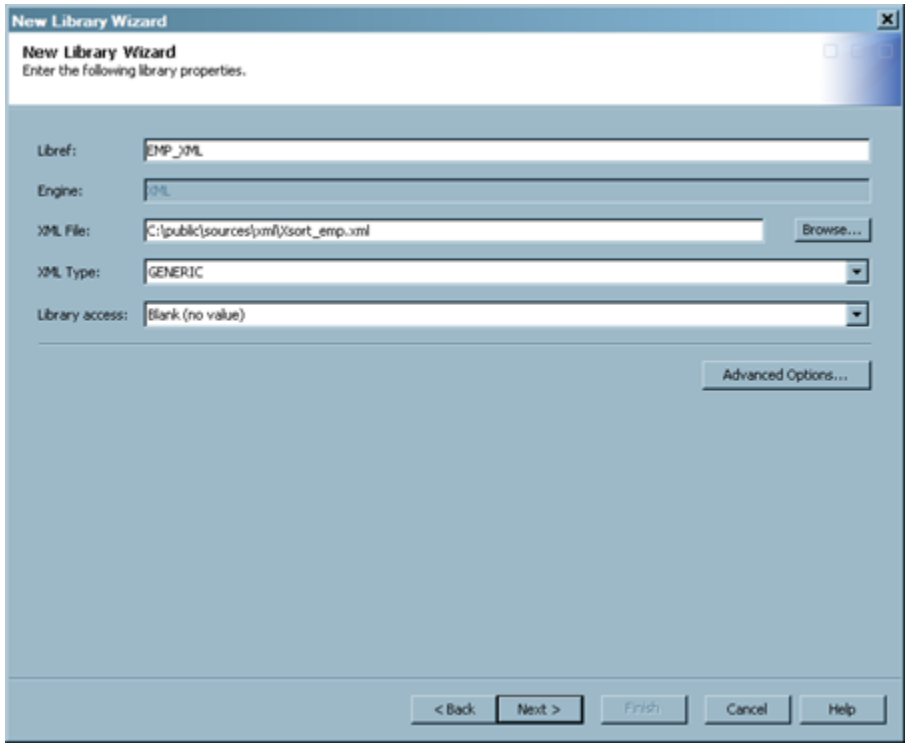
Register an XML Library for the XML Target Table

The XML Writer transformation uses a SAS XML library to access the file that contains the XML target table. Accordingly, to use the XML Writer transformation, you must have a SAS XML library that points to the file that contains the XML target table.

Perform the following steps to register an XML library that points to the file that contains an XML table:

1. Right-click a destination folder in the Folders tree. Then select **New** ⇨ **Library** from the pop-up menu to access the New Library Wizard window.
2. Select **SAS XML Library** from the list of library types. Then, click **Next** to access the general details page.
3. Enter a name and an optional description for the library. For example, the name for the library in the sample job could be EMP_XML. Then, click **Next** to access the Available servers page.
4. Select a server in the Available servers field and move it to the Selected servers field. (The server for the sample library is SASApp.) Then, click **Next** to access the library properties page.
5. Enter a libref for the new library in the **Libref** field and a path to an appropriate XML file in the **XML File** field. For the sample job, you would specify the path to the XML file (Xsort_emp.xml) that contains the target XML table. This file does not yet exist, but it is created when the job is executed. The following display shows the library properties for the sample job:

Display A3.73 *XML Library Properties*



For more information about SAS XML libraries, see the *SAS XML LIBNAME Engine: User's Guide*.

6. Click **Next** to access the summary page. Review the details and click **Finish** to save the library and close the New Library Wizard window.

Register the XML Target Table

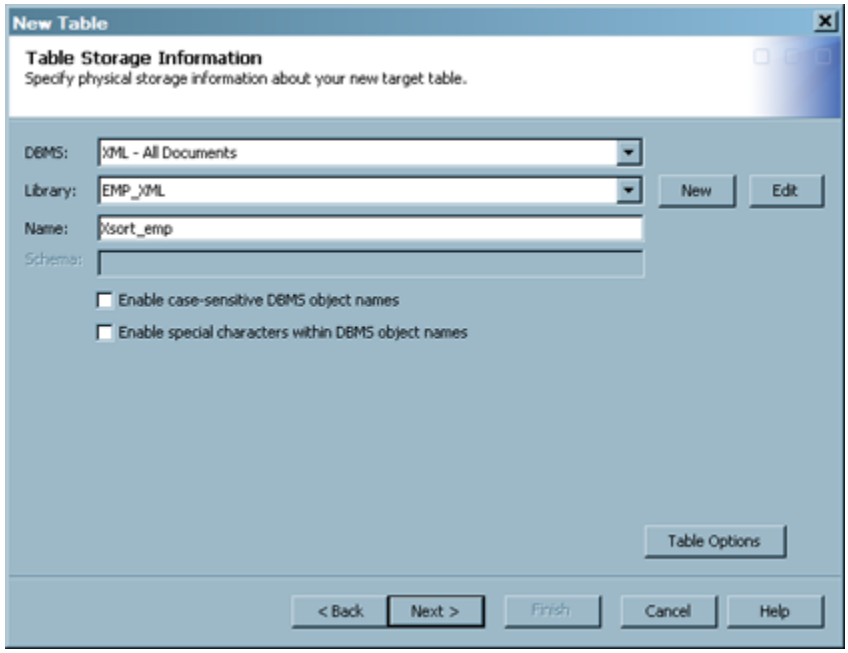
By registering the XML library, you registered the location of the XML file that contains the XML target table. You must also register the XML table itself, to specify its columns and other attributes.

The New Table wizard is used to register a 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. This wizard enables you to copy metadata from one or more registered tables into the metadata object for the new table. For the sample job, assume that we want all columns in the input table (EMPLOYEES...) to appear in the XML target table.

Perform the following steps to register the XML target table:

1. Right-click a destination folder in the Folders tree. Then select **New** ⇨ **Table** from the pop-up menu to access the General Information page of the New Table window.
2. Enter a name and an optional description for the target table. For example, the name for the table in the sample job is Xsort_emp. Then, click **Next** to access the Table Storage Information page.
3. Use the drop-down menus in the **DBMS** and **Library** fields to select the appropriate DBMS type and library name values. (The sample job values are **XML - All Documents** and **EMP_XML**.) The following display shows the table storage information for the sample job:

Display A3.74 *XML Table Storage Information*



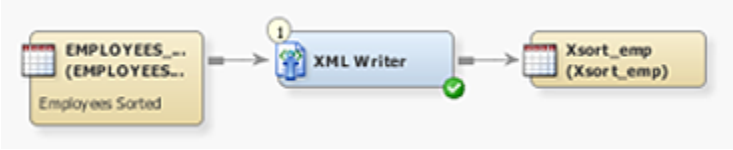
4. Click **Next** to access the Select Columns page.
5. Navigate in the **Available Tables** field until you find the table containing the columns that you want to use for the target table definition. Then, move the columns to the **Selected** field. For the sample job, all of the columns in the EMPLOYEES... table are used. Click **Next** to access the Change Columns/Indexes page.
6. Review the column data for the table and make any necessary changes. Click **Next** to access the summary page.
7. Review the details and click **Finish** to save the table and close the New Table wizard.

Create and Populate the XML Writer Job

Perform the following steps to create and populate a job:

1. Create an empty SAS Data Integration Studio job.
2. Select and drag an XML Writer 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.
3. Select and drag the source table from the Inventory tree. Then, drop it before the XML Writer transformation on the **Diagram** tab. The source table for the sample job is named EMPLOYEES... .
4. Drag the cursor from the source table to the input port of the XML Writer transformation. This action connects the source to the transformation.
5. Select and drag the XML target table from the tree view. For the sample job, the XML table is named Xsort_emp. Then drop the XML table after the XML Writer transformation on the **Diagram** tab.
6. Drag the cursor from the output port of the XML Writer transformation to the target table. This action connects the transformation to the target. The process flow looks similar to the following display:

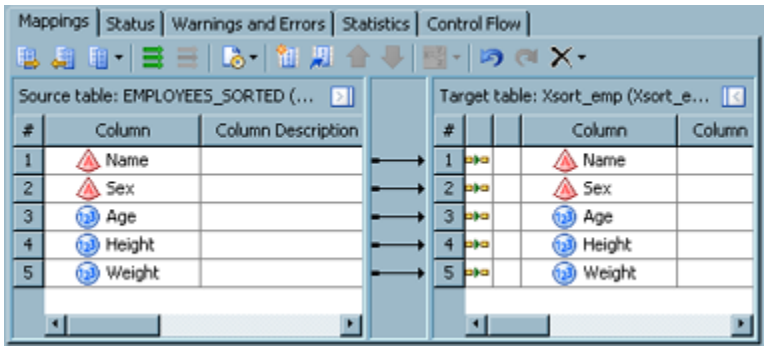
Display A3.75 *Sample Process Flow for an XML Writer Job*



7. Select the XML Writer transformation in the process flow on the **Diagram** tab. Then, click the **Mapping** tab on the Details pane.

8. Review the mappings between the **Source table** and **Target table** fields. Correct any improper mappings between the two tables. The following display shows the **Mapping** tab for the sample job:

Display A3.76 *XML Writer Mapping Tab*

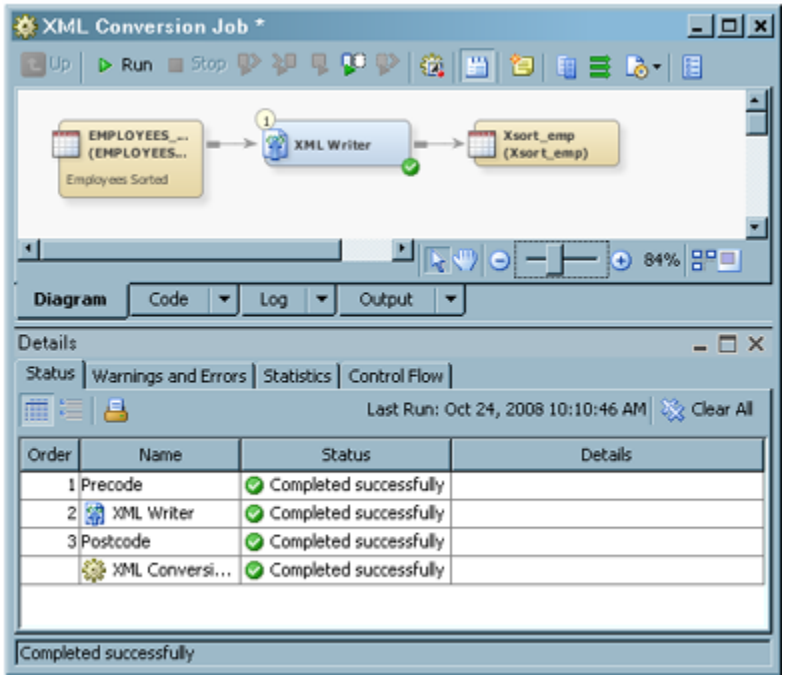


Run the Job and Verify the Results

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. The following display shows a successful run of the sample job:

Display A3.77 *Sample Completed Job*



2. If error messages display, read and respond to the messages as needed.

To review the data in the XML table, open the XML file in a Web browser or XML editor. The following display shows the table header and first data row in the XML file: Xsort_emp.xml.

Display A3.78 *XML Target Table*

```
<?xml version="1.0" encoding="windows-1252" ?>
<TABLE>
   <ALL_EMPX>
      <Name> Joyce </Name>
      <Sex> F </Sex>
      <AGE> 11 </AGE>
      <HEIGHT> 51 </HEIGHT>
      <WEIGHT> 50 </WEIGHT>
   </ALL_EMPX>
```

Using ODS to Specify Output from the XML Writer

Problem

You want to change how the output of the XML Writer transformation is displayed.

Solution

You can use an Output Delivery System (ODS) tagset to change the output. You cannot use an XML Map because the SAS XML LIBNAME statement does not support XML Maps for write operations. You must be familiar with ODS and the general steps for

editing jobs in SAS Data Integration Studio. For more information about ODS, see the *SAS Output Delivery System: User's Guide*.

Tasks

Specify an ODS Tagset for the XML Writer Transformation in a Job

Perform the following steps to edit the ODS tagset for the XML Writer Transformation in a job.

1. Open the properties window for the XML Writer transformation.
2. Click the **Precode and Postcode** tab.
3. Select **Precode**.
4. Edit the sample ODS tagset that is displayed in the **Precode** field.
5. Click **Save As** on the toolbar for the **Precode and Postcode** tab to access the Save As window.
6. Select **Metadata** to save the ODS tagset as metadata.
7. Click **OK** to save the tagset and return to the **Precode and Postcode** tab.
8. Click **OK** to close the properties window for the XML Writer transformation.

After you have specified an ODS tagset, you can run the job and verify the output.

Using SOAP to Access a Third-Party Web Service

Problem

You want a job to access a third-party web service that uses a SOAP interface.

Solution

Create a job that uses a SOAP transformation to access the web service. SOAP (Simple Object Access Protocol) is a simple XML-based protocol to let applications exchange information over HTTP. The SOAP transformation generates SAS SOAP procedure code to access a web service in the context of a job. The procedure invokes a SOAP web service through Java Native Interface (JNI). The procedure provides options for the request XML document (IN), the service endpoint (URL), the SOAP action or operation (SOAPACTION), and the map used to handle the response (OUT).

To use the SOAP transformation, perform the following tasks:

- “Verify Prerequisites” on page 747.
- “Create a Job That Uses a SOAP Transformation to Access a Web Service” on page 748.

Verify Prerequisites

It is assumed that you are familiar with how to use the SOAP interface to access web services. Gather the same information that you need to use the SAS SOAP procedure to

read from and write to the desired web service. Example options and values are shown in the next section. For detailed information about the SAS SOAP procedure, see the documentation for that procedure in the *Base SAS Procedures Guide*.

Create a Job That Uses a SOAP Transformation to Access a Web Service

Perform the following steps. It is assumed that you are familiar with creating SAS Data Integration Studio jobs, as described in "Creating Jobs" on page 146.

1. Create an empty SAS Data Integration Studio job.
2. In the **Transformations** tree of the process editor, open the **Access** folder.
3. Drag the **SOAP** transformation into the empty job.
4. Right-click the **SOAP** transformation and select **Properties**.
5. Specify the options that you need to read from and write to the desired web service. See the following table for some example values.
6. Save the job.
7. Run the job and verify your output.

The following table shows an example set of options that you can specify in the SOAP transformation. In this example, the web service is available from the National Weather Service in the United States. For the example transaction, a ZIP code is passed to the service in XML format (WeatherGovRequestXML.xml). The service returns the longitude and latitude for that ZIP code in XML format (SOAPSoapActionOutputFile.xml).

Table A3.1 *Example Options for the SOAP Transformation*

Tab Selection Path	Fields	Example Values
Options ⇨ Input File	Specify the SAS fileref associated with request XML file (IN):	soapacti
	File Options:	RECFM=V LRECL=5000 *Note:* RECFM=V specifies that the input file is in variable format. LRECL=5000 specifies the logical record length for the input file in bytes.
	Will the SOAP Request XML be contained in one file?	Yes
	Specify an existing file that contains the SOAP request XML:	C:\SOAPFiles \WeatherGovRequestXML.xml

Tab Selection Path	Fields	Example Values
Options ⇨ Output File	Specify the SAS fileref associated with response XML file (OUT):	soapacto
	Specify complete path and filename of response XML file	C:\SOAPFiles \SOAPSoapActionOutputFile.xml
	File Options:	RECFM=V LRECL=5000 *Note:* RECFM=V specifies that the output file is in variable format. LRECL=5000 specifies the logical record length for the output file in bytes.
Options ⇨ Additional Required Aurguments	Will the call be made to a SAS registered Web service?	No *Note:* One example of a SAS registered service is a SAS Data Integration Studio job that has been deployed as a web service.
	Specify the URL of the Web service endpoint to be called:	http://graphical.weather.gov/xml/ SOAP_server/ndfdXMLserver.php
Options ⇨ SOAP Optionssection	Specify an output log file used for debugging purposes (DEBUG):	C:\SOAPFiles \SOAPSoapActionDebug.txt *Note:* Includes debugging statements from the **mprint** option set.
	Specify a defined SOAP action (SOAPACTION):	http://graphical.weather.gov/xml/ DWMLgen/wsdl/ ndfdXML.wsdl#LatLonListZipCo de
	Specify the mustUnderstand attribute in the SOAP header (MUSTUNDERSTAND):	Yes
Options ⇨ Security	Will Web authentication be used?	No
	Will WS-Security Web services be used?	No
	Will an HTTP proxy server be used?	Yes
	Specify the HTTP Proxy Server Host Name (PROXYHOST):	my.proxy.com
	Specify the HTTP Proxy Server Port (PROXYPORT):	80

Tab Selection Path	Fields	Example Values
Options ⇨ Properties	Is a SOAP envelope to be included in the XML response?	Yes
Options ⇨ Additional Options	System options:	mprint *Note:* Specifies that SAS statements generated by macro execution are traced for debugging.

Using REST to Access a Third-Party Web Service

Problem

You want a job to access a third-party web service that uses a REST interface.

Solution

Create a job that uses a REST transformation to access the web service. Representational State Transfer (REST) is a set of architectural principles for designing web services that access a system's resources. A resource is accessed with a Uniform Resource Identifier (URI). The REST transformation generates SAS HTTP procedure code to read from and write to a web service in the context of a job.

To use the REST transformation, perform the following tasks:

- "Verify Prerequisites" on page 750.
- "Create a Job That Uses a REST Transformation to Access a Web Service" on page 750.

Verify Prerequisites

It is assumed that you are familiar with how to use the REST interface to access web services. Gather the same information that you need to use the SAS HTTP procedure to read from and write to the desired web service. Example options and values are shown in the next section. For detailed information about the SAS HTTP procedure, see the documentation for that procedure in the *Base SAS Procedures Guide*.

Create a Job That Uses a REST Transformation to Access a Web Service

Perform the following steps. It is assumed that you are familiar with creating SAS Data Integration Studio jobs, as described in "Creating Jobs" on page 146.

1. Create an empty SAS Data Integration Studio job.
2. In the **Transformations** tree of the process editor, open the **Access** folder.
3. Drag the **REST** transformation into the empty job.

4. Right-click the **REST** transformation and select **Properties**.
5. Specify the options that you need to read from and write to the desired web service. See the following table for some example values.
6. Save the job.
7. Run the job. Review status messages to verify that the job executed without error.

The following table shows an example set of options that you can specify in the REST transformation. In this example, when you submit a request to the web service, it returns contact information in the output variable *RESTGetText_Auth_UserPwd*.

Table A3.2 *Example Options for the REST Transformation*

Tab Selection Path	Fields	Example Values
Options ⇨ REST Properties	HTTP Method:	GET
	HTTP request endpoint:	http://myComputer.com/contactservice/contactservice.svc/contacts
	HTTP context-type:	application/x-www-form-urlencoded
	HTTP charset:	windows-1252
Options ⇨ Input and Output	Input type:	Text
	Input text:	None
	Output type:	Text
	Output Text Macro Variable Name:	RESTGetText_Auth_UserPwd
Options ⇨ Authentication	Web authentication type:	Basic
	Web authentication username:	myUserName
	Web authentication password:	myPassword

Generating Enterprise Decision Management Output

Overview

Use the Enterprise Decision Management transformation to generate Enterprise Decision Management output in the context of a SAS Data Integration Studio job. The

transformation uses an Enterprise Decision Management flow package that is created in SAS Decision Manager. The package is used to structure the data in an input table and generate columns in the output tables. The Enterprise Decision Management transformation maps physical data to decision flows. The output tables attached to the transformation produce decision-making results from the mapped input data. SAS Decision Manager supports collaborative maintenance, governance, and continuous enhancement of analytical models for operational decision making.

The required inputs, or columns, for an Enterprise Decision Management flow job are defined by Enterprise Decision Manager. Each Enterprise Decision Management flow job that you run in SAS Data Integration Studio requires a source table that contains the required inputs defined by the Smart Object.

When the source table columns are equivalent in name, the job uses the original source table to provide the required inputs. (For relational database management systems, the columns also must be identical in case. If the input table is a SAS table, only identical column names are required.)

However, when the source table columns are not equivalent to the required input columns, a view is generated as input to the job. Therefore, you should try to use a source table with column names that are identical to the defined required inputs to enhance performance.

The Enterprise Decision Management flow package that is created in SAS Decision Manager contains an automatically generated correlation_id column in its required inputs. However, it is also possible to manually define a correlation_id request variable or a correlation_id reply variable in the SAS Decision Manager application. Do not create one of these manually defined correlation_id variables if you plan to use the Enterprise Decision Management flow package in the Enterprise Decision Management transformation in a SAS Data Integration Studio. When you attempt to run the job under this circumstance, the following error is displayed:

```
ERROR: Compilation error.
ERROR: Duplicate declaration for correlation_id with incompatible types.
```

Mixed-case reply variables contained in Enterprise Decision Management flow packages can cause Enterprise Decision Management flow jobs to fail when you run them in SAS Data Integration Studio. To avoid this problem, select the **Enable case-sensitive DBMS object names** check box on the **Physical Storage** tab in the properties window for the **EDM Results** tab.

Problem

You want to generate Enterprise Decision Management output from a source table in a SAS Data Integration Studio job.

Solution

You can use the Enterprise Decision Management transformation in a SAS Data Integration Studio job to generate Enterprise Decision Management output. For example, you can create a job that contains a source table, the Enterprise Decision Management transformation, and two output tables named EDM Results and EDM Stats. This sample job includes the following tasks:

- "Create and Populate the Job" on page 753
- "Configure the Enterprise Decision Management Transformation" on page 753
- "Run the Job and View the Output" on page 756

Tasks

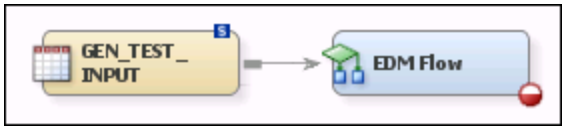
Create and Populate the Job

Perform the following steps to create and populate the job:

1. Create an empty SAS Data Integration Studio job with an appropriate name, such as EDM Output Job.
2. Select and drag an Enterprise Decision Management 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 the source table from the Inventory tree. Then, drop it before the Enterprise Decision Management transformation on the **Diagram** tab. The source table for the sample job is named GEN_TEST_INPUT.
4. Drag the cursor from the source table to the input port of the Enterprise Decision Management transformation. This action connects the sources to the transformation.

   The sample job is shown in the following display:

   Display A3.79 *Enterprise Decision Management Job Flow*

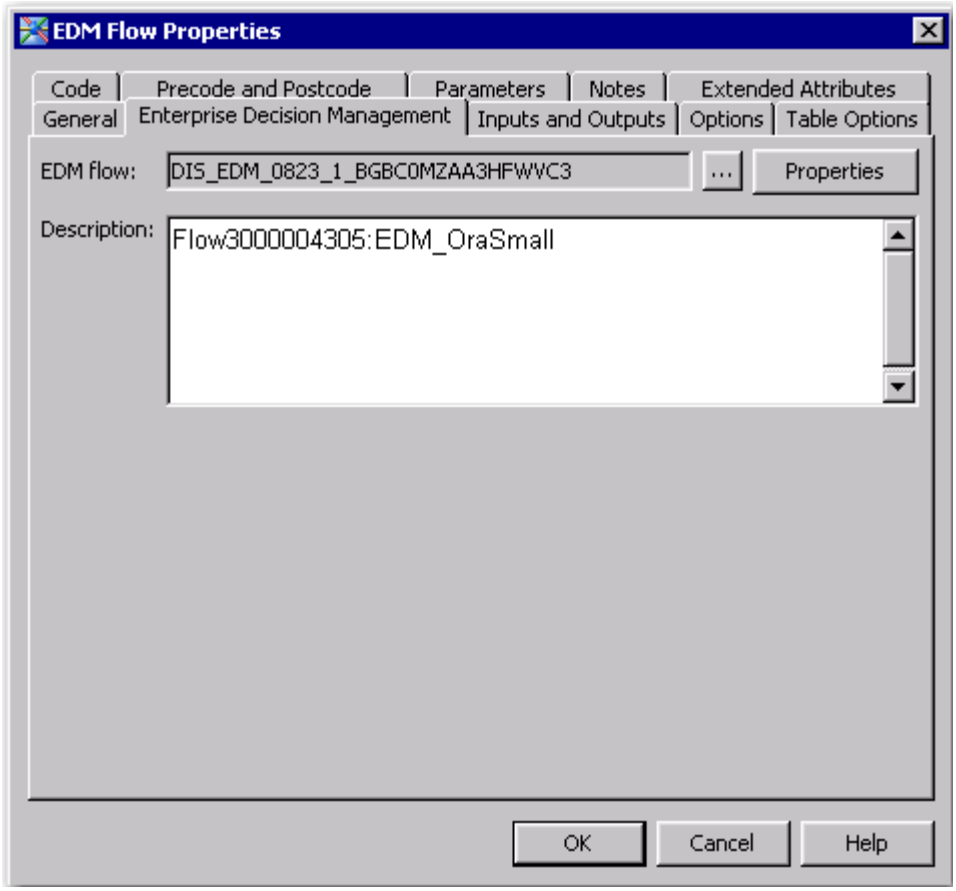
   Note that the Enterprise Decision Management transformation is named EDM Flow in the context of a SAS Data Integration job flow.

Configure the Enterprise Decision Management Transformation

To configure the Enterprise Decision Management transformation, select an Enterprise Decision Management flow in the **Enterprise Decision Management** tab. Then, map all required inputs with source table columns in the **Inputs and Outputs** tab.

The settings for the **Enterprise Decision Management** tab are shown in the following display:

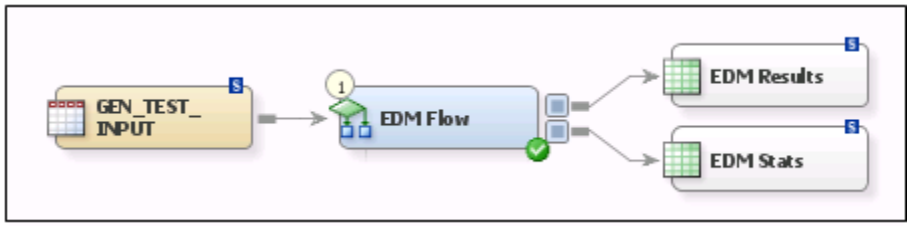
Display A3.80 *Enterprise Decision Management Tab*



When you select an Enterprise Decision Management flow, the output tables associated with the Enterprise Decision Management flow are added to the job flow.

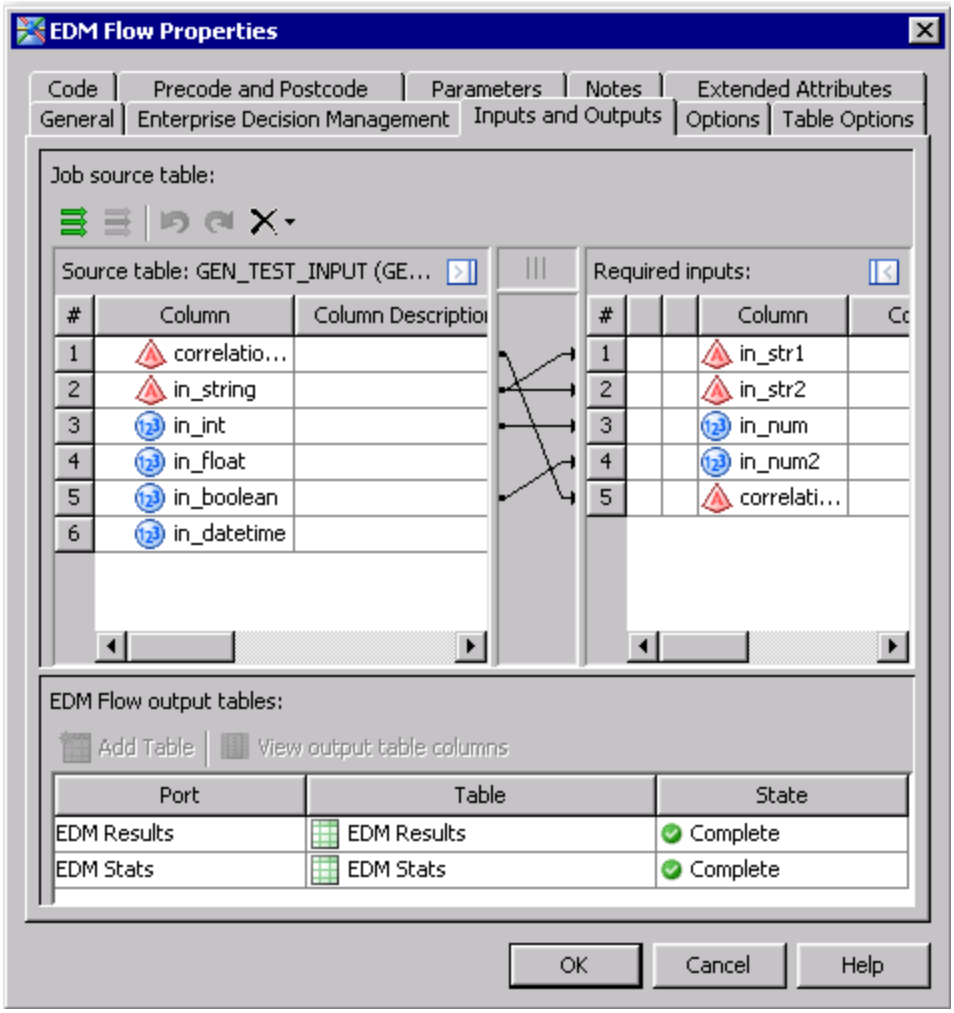
The completed job flow is shown in the following display:

Display A3.81 *Completed EDM Job Flow*



The settings for the **Inputs and Outputs** tab are shown in the following display:

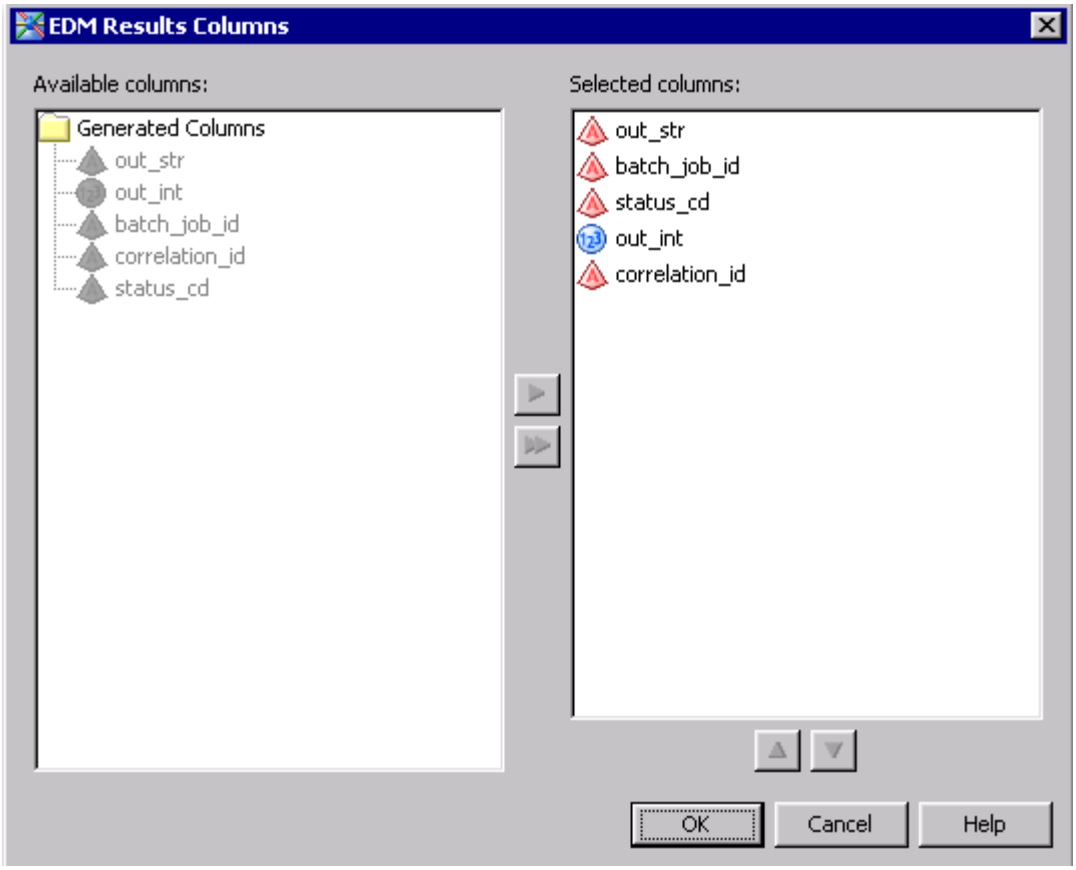
Display A3.82 *Inputs and Outputs Tab*



Note that you can select one of the output tables listed in the table and click **View output table columns** to review its columns.

The following display shows the output columns for the EDM Results table:

Display A3.83 *EDM Results Columns*

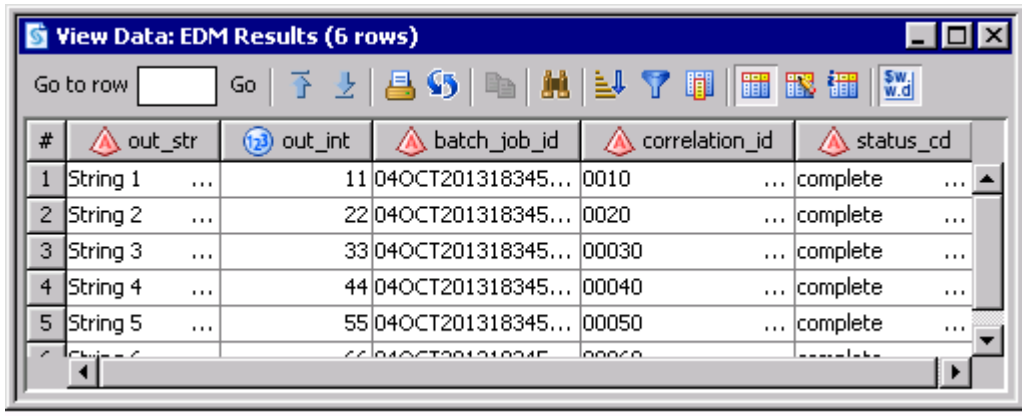


Run the Job and View the Output

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 the job completes successfully, you can right-click an output table and click **Open** to review its output table.

The following display shows a view of the EDM Results output table.

Display A3.84 *EDM Results Output*



#	out_str	out_int	batch_job_id	correlation_id	status_cd
1	String 1 ...	11	04OCT201318345...	0010 ...	complete ...
2	String 2 ...	22	04OCT201318345...	0020 ...	complete ...
3	String 3 ...	33	04OCT201318345...	00030 ...	complete ...
4	String 4 ...	44	04OCT201318345...	00040 ...	complete ...
5	String 5 ...	55	04OCT201318345...	00050 ...	complete ...

Usage Notes

The temporary work tables for the Enterprise Decision Management transformation should be left in SAS data set format. Otherwise, the transformation might fail.

Transformations in a job typically create temporary work tables as they execute. The default work tables are SAS data sets in the SAS WORK library. You can redirect these temporary tables to an alternative format and location, as described in "Redirecting Temporary Output Tables" on page 196. However, the temporary work tables for the Enterprise Decision Management transformation should be left in SAS data set format.

The Enterprise Decision Management transformation requires an Enterprise Decision Management flow. SAS Decision Manager 2.1 can produce this flow. SAS Decision Manager 2.2 cannot produce this flow. In order to use the Enterprise Decision Management transformation, you can use SAS Decision Manager 2.1 to export the flow to a location where the Enterprise Decision Management transformation can access it. You can also use a version of SAS decision management software later than SAS Decision Manager 2.2 to produce this flow.

The global option "Use run time lookup for credentials for statements requiring credentials" should not be enabled when you run a job with an Enterprise Decision Management transformation, and the input to that transformation is an Oracle table. If you disable this option, any SAS Data Integration Studio job with SAS Connect sign-on statements will include user names and passwords in the generated code.

Running Conditional Processes

Overview

The Conditional Start and Conditional End transformations support conditional flow processing within a SAS Data Integration Studio job. The Conditional Start transformation uses SAS macro %if logic. You can define distinct portions of a job to run conditionally, based on whether the associated defined condition is determined to be true or false. Job flow is sequential and multiple conditions can be defined within the same job that might or might not be logically related to one another. Each conditional definition block that you create starts and ends a task or series of tasks that run if the defined condition is true.

If you need to include a Return Code Check transformation in a job that contains the Conditional Start and Conditional End transformations, place it in one of the following locations:

- after the transformations that are being processed between the Conditional Start and Conditional End transformations
- after the Conditional End transformation

.

Problem

You want to run one or more sections of a SAS Data Integration Studio job as a conditional process.

Solution

You can use the Conditional Start and Conditional End transformations to define sections of the job that processed conditionally. For example, you can create a job that sorts an input table for country whenever the job is run after the first 15 days of the current month. The input table and Sort transformation in this job are placed between Conditional Start and Conditional End transformations to define a conditional process. Then, a condition is set in the Conditional Start transformation to specify the days in the month in which the process can be run. This sample job includes the following tasks:

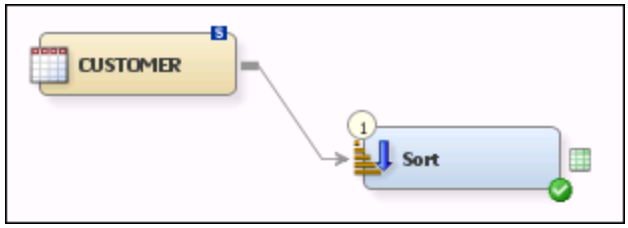
- "Create and Configure the Job" on page 758
- "Run the Job and View the Output" on page 761

Tasks

Create and Configure the Job

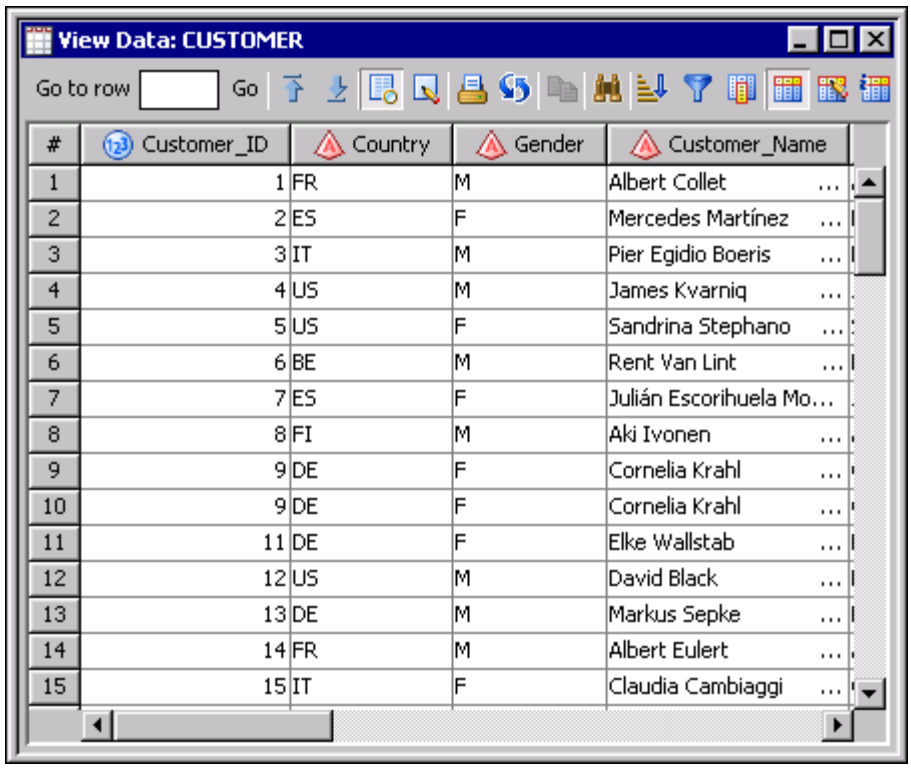
The following display shows a simple job that sorts a table of customer data by country:

Display A3.85 *Sort Job*



You can review the initial, unsorted data for the Customer source table, as shown in the following display:

Display A3.86 *Customer Source Table*

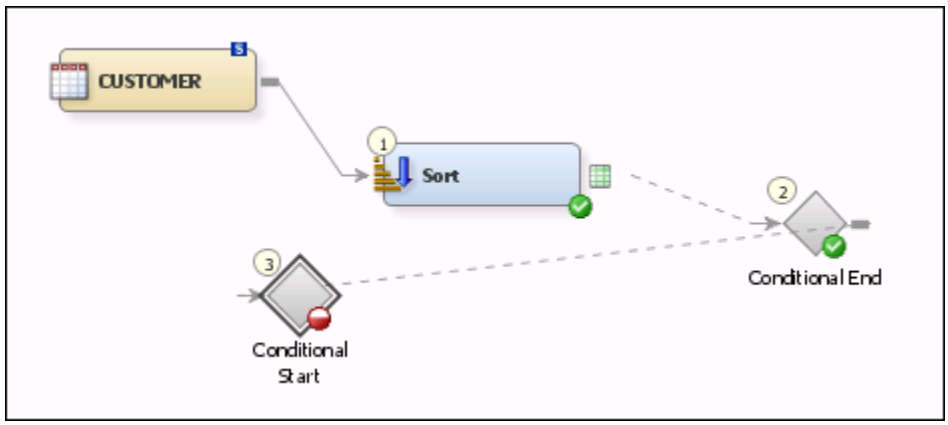


#	Customer_ID	Country	Gender	Customer_Name
1	1	FR	M	Albert Collet
2	2	ES	F	Mercedes Martínez
3	3	IT	M	Pier Egidio Boeris
4	4	US	M	James Kvarniq
5	5	US	F	Sandrina Stephano
6	6	BE	M	Rent Van Lint
7	7	ES	F	Julián Escorihuela Mo...
8	8	FI	M	Aki Ivonen
9	9	DE	F	Cornelia Krahl
10	9	DE	F	Cornelia Krahl
11	11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

You can add the Conditional Start and Conditional End transformations to a SAS Data Integration Studio job to create a conditional definition block.

Note that the Conditional Start and Conditional End transformations do not yet enclose the Sort transformation and the Customer table in the following display:

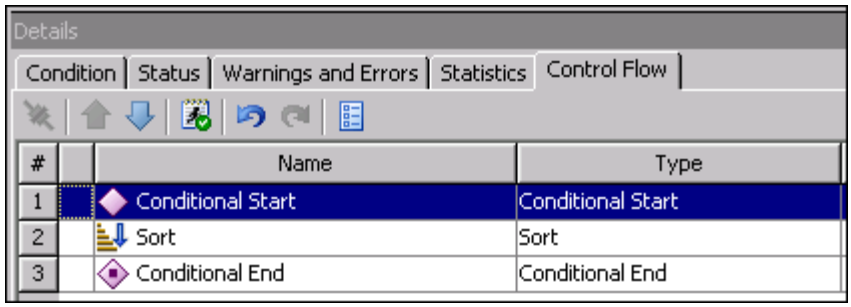
Display A3.87 *Conditional Transformations*



You must reorder the contents of the **Diagram** tab to define the conditional definition block.

Use the **Control Flow** tab reorder these contents, as shown in the following display:

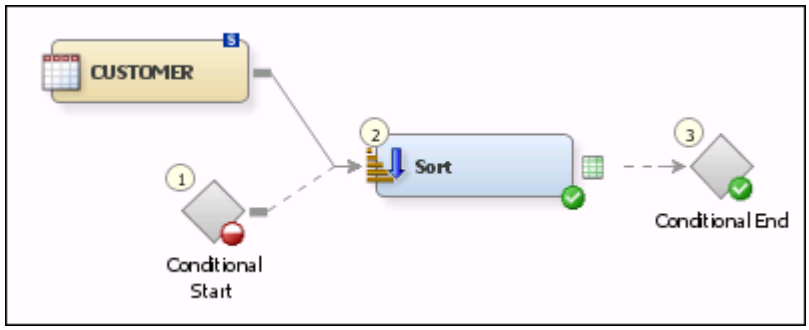
Display A3.88 *Control Flow Tab*



Note that the tab shows the transformations in the following order: Conditional Start, Sort, and Conditional End.

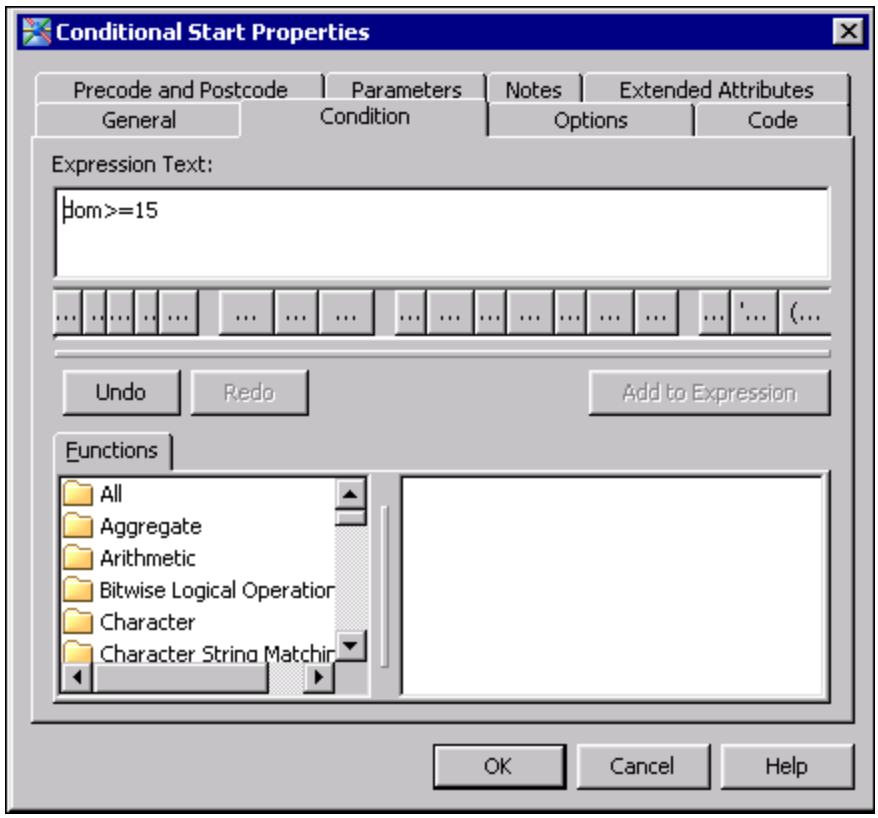
The following display shows the reordered flow on the **Diagram** tab:

Display A3.89 *Reordered Conditional Transformations*



Now that the job in is enclosed in the conditional definition block, the condition can be defined in the **Condition** tab, as shown in the following display:

Display A3.90 *Condition Tab*



The condition input in the **Condition** tab in the is processed by the %eval (*Condition*) function in the generated code of the transformation. You can click on the function in the **Functions** pane to see a description.

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. You can check the **Status** tab to verify that the job completes successfully and that the condition flow is executed.

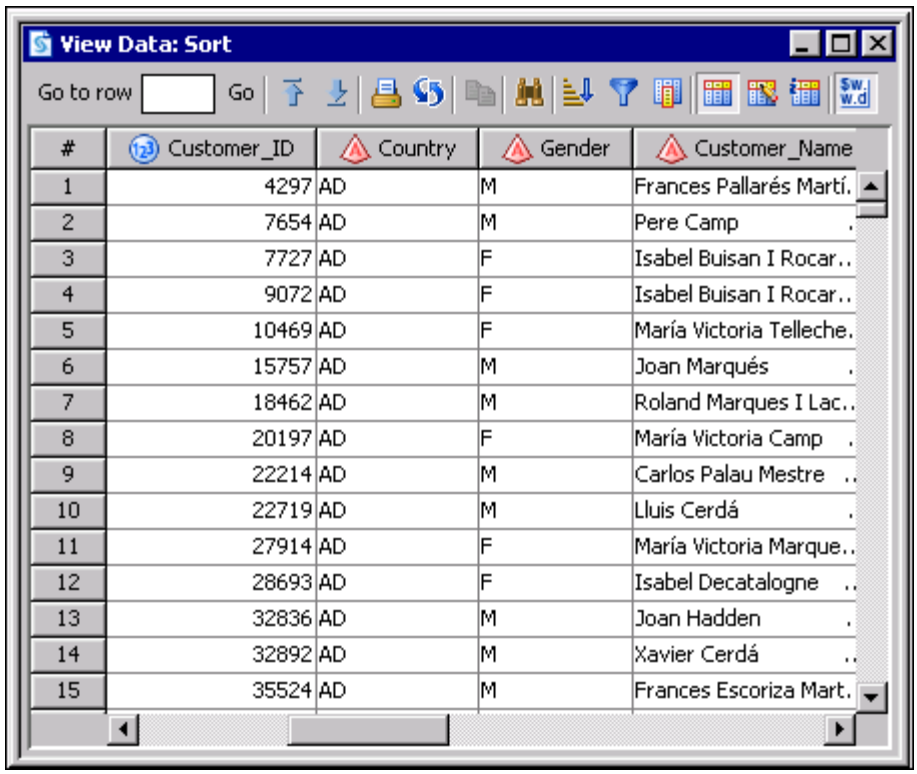
The **Status** tab is shown in the following display:

Display A3.91 *Job Status*

Order	Name	Status	Details
1	Precode	Comple...	
2	Conditio...	Comple...	ETLS_DIAG: Condition flow executed, conditio...
3	Conditio...	Comple...	
4	Postcode	Comple...	
	If Then ...	Comple...	

You can also verify that output changed in accordance with the condition, as shown in the following display:

Display A3.92 *Conditional Output*



#	Customer_ID	Country	Gender	Customer_Name
1	4297	AD	M	Frances Pallarés Martí.
2	7654	AD	M	Pere Camp
3	7727	AD	F	Isabel Buisan I Rocar..
4	9072	AD	F	Isabel Buisan I Rocar..
5	10469	AD	F	María Victoria Telleche.
6	15757	AD	M	Joan Marqués
7	18462	AD	M	Roland Marques I Lac..
8	20197	AD	F	María Victoria Camp
9	22214	AD	M	Carlos Palau Mestre
10	22719	AD	M	Lluis Cerdá
11	27914	AD	F	María Victoria Marque..
12	28693	AD	F	Isabel Decatalogne
13	32836	AD	M	Joan Hadden
14	32892	AD	M	Xavier Cerdá
15	35524	AD	M	Frances Escoriza Mart.

In this case, the output is sorted by country This result is expected because the expression in the **Condition** tab was executed.

Appendix 4

Java Code and Methods for Report Plug-ins

Example Java Code for a Report Plug-in 763
Reporting Interface Methods 769

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 the 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 763.

The following table contains information about the methods you can use to create your own report.

Table A4.1 *Reporting Interface Methods*

Name	Description	Default Abstract Report Implementation
getName();	This method returns the report name and is displayed in the Name column of the Reports window.	Not implemented.
getDescription();	This method returns the report description and is displayed in the Description column of the Reports window.	Not implemented.
String getCategory();	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.
StringBuffer getSourceCode();	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.
void onSelected();	This method executes when the user runs the selected report before submitting any SAS code to the application server that is returned by the getSourceCode() 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 **Browse** 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 **Additional report options** 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 **Format** 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 **Style** 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 **Additional options** 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.

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
a property or set of properties that is drawn from source data, and is used to uniquely identify a record. For example, if customer records include a unique customer ID, the customer ID might be selected for use as a business key.

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
See change management facility

change management facility
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. 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.

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.

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 whether 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.

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 serves as a boundary that separates the elements of a text string.

delivery transport
in the Publishing Framework, the method of delivering a package to the consumer. Supported transports include e-mail and WebDAV. Although not a true transport, a channel also functions as a delivery mechanism.

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 data element that categorizes values in a data set into non-overlapping categories that can be used to group, filter, and label the data in meaningful ways. Hierarchies within a dimension typically represent different groupings of information that pertains to a single concept. For example, a Time dimension might consist of two hierarchies: (1) Year, Month, and Date, and (2) Year, Week, and Day.

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.

enrichment
the addition of value to data in a data table by appending information from another source or by applying analytics.

fact table
the central table in a star schema or snowflake schema. The fact table contains the individual facts that are being stored in the database as well as the keys that connect each fact to the appropriate value in each dimension.

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 metadata repository that is used to specify metadata for global resources that can be shared by other repositories. For example, a foundation repository is used to store metadata that defines users and groups on the metadata server.

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.

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 files and objects that use a particular data source, in order to assess the impact of any changes to that data source.

Integrated Object Model server
See 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
a SAS object server that is launched in order to fulfill client requests for IOM services. Short form: IOM 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.

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 can create output.

locale
a setting that reflects the language, local conventions, and culture for a geographic region. Local conventions can include specific formatting rules for paper sizes, dates, times, and numbers, and a currency symbol for the country or region. Some examples of locale values are French_Canada, Portuguese_Brazil, and Chinese_Singapore.

lookup standardization
a process that applies a scheme to a data set for the purpose of data analysis or data cleansing.

match code
an encoded representation of an input character variable that is used to cluster and compare data.

message queue
in application messaging, a place where one program can send messages that are 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.

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.

metadata server
a server that provides metadata management services to one or more client applications.

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.

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.

parameterized job
a job that specifies its inputs and outputs as parameters.

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.

primary key
a column or combination of columns that uniquely identifies a row in a table.

project repository
a metadata repository that serves as an individual work area or playpen. In general, each user who participates in change management has an individual project repository. (Project repositories are available for SAS Data Integration Studio only.)

publish
to deliver electronic information 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.

queue
See message queue

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.

reverse impact analysis
See data lineage

SAS Application Server
a logical entity that represents the SAS server tier, which in turn comprises servers that execute code for particular tasks and metadata objects.

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.

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/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 reusable collection of match codes and standardization values that is applied to input character values for the purposes of transformation or analysis.

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 user who has the necessary credentials to administer one or more SAS servers. Server administrators can perform actions such as stopping, pausing, and resuming servers, and administering server logs.

server component
in SAS Management Console, a metadata object that specifies information about how to connect to a particular type 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.

source
See source data

source data
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.

target
a container for output data that has been extracted from a source. A target can be a table, view, or file.

transport
See delivery transport

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
a set of extensions to the HTTP protocol that enables users to collaboratively edit and manage files on remote web servers. Short form: WebDAV.

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 32
accessibility features 10
actions
  based on job status 216
  based on transformation status 217
  default 210
  prerequisites for 215
administrative tasks 9
aggregate columns 299
Analysis folder 34
Analysis window 614
application servers
  modifying configuration files or SAS start commands for 269
Archived folder 35
assistive technologies 10
Authorization tab 19
automatic column mappings 186
automatic joins 447
automatic propagation 190

B

Basic Properties pane 631
browsing table data 111
buffering options 487
bulk load tables 486
bulk loading 433

C

case
  in table and column names 88
CASE expressions 461
change data capture (CDC) 557
  changed data tables 559
  control tables 559
  from Oracle 560
  prerequisites for 558
Change Data Capture folder 35
Change Data Capture transformations 557
change detection
  dimension tables 525
change management 47
  adding metadata 49
  checking in metadata 50
  checking out metadata 50
  clearing metadata from projects 51
  creating connection profiles for administrators 49
  creating connection profiles for users 49
  deleting metadata 51
  undoing checkouts 51
  usage notes for 52
change tracking
  datetime 554, 556
  dimension tables 524
checkouts
  undoing 51
Checkouts tree 615
clauses
  adding subqueries to 473
  reviewing and modifying 445
cleansing data 3, 297
clearing metadata 51
cluster tables 577
COBOL copybooks 136
COBOL data files 136
code
  common code generated for jobs 160
  credentials in generated code 163
  displaying SAS code for jobs 160
  for transformations 195
  generated 263
  jobs with generated source code 146
  jobs with user-supplied source code 147
  overriding generated code 132
  user-written 271

user-written SQL code 488
Code Editor 615
code testing 128
column mappings 185
automatic 186
deleting 188
derived 187
from source table to work table 276
from work table to target table 276
one-to-one 187
options for 189
column metadata
adding 92
additional operations 95
maintaining 92
modifying 93
notes and documents for 95
column names
case and special characters in 88
columns
adding to query target table 455
aggregate columns 299
avoiding unneeded columns 298
deleting from indexes 110
dropping unneeded columns 298
key columns 104
managing for performance 297
matching variable size to data length 299
rearranging in indexes 111
scope of column changes in jobs 189
updating columns in keys 109
Comparison Results window 616
conditional action sets
default 210
conditions
default 210
configuration files
modifying for application servers 269
Connection Profile window 617
connection profiles
creating 23
creating for administrators, with change management 49
creating for users, with change management 49
opening 23
updating 24
connections for objects 193
connectivity 3
constraints
removing non-essential constraints during a load 432
Control Flow tab 174
Control folder 36
control tables 512
registering 512
credentials
in generated code 163, 266
cross-reference tables 526
cube wizards 640
cubes
registering 29

D

data cleansing 3, 297
data enrichment 3
data federation 4
Data folder 36
data integration 3
advantages of 4
data integration environment 4
libraries 8
overview diagram 4
SAS Data Integration Studio 5
SAS Management Console 5
servers 6
data optimization 480
Data Quality folder 38
data surveyor wizards 641
Data Transfer transformation 158
data validation 297
datetime change tracking 554
closing out rows in 556
DB2 Bulk Table Loader transformation 422, 426
DBMS names 89
setting options in Register Tables wizard 91
DBMS servers 7
DBMS tables
preserving foreign keys in 654
debugging 301
adding code to process flows 302
checking job status 301
limiting input to transformations 302
setting and checking status codes 303
setting SAS invocation options on jobs 303
SQL queries 453
verifying output from transformations 302
default actions 211
default conditional action sets 213
default conditions 210
default SAS Application Server 27
Delimited External File wizard 121
delimited external files
registering 120
deploying jobs 226
as stored processes 237, 238

as Web services 243
creating Web service jobs 244
for execution on remote host 236
for scheduling 227
prerequisites for deploying as stored process 237
prerequisites for scheduling 227
prerequisites for Web service jobs 243
redeploying jobs for scheduling 233
redeploying to stored processes 241
requirements for Web service jobs 244
scheduling for complex process flows 235
stored process as Web service 251
viewing or updating stored process metadata 242
Web service jobs as stored process 249
derived column mappings 187
Designer tab (SQL)
adding CASE expression 462
adding GROUP BY clause 466
adding HAVING clause 467
adding joins to queries 455
adding ORDER BY clause 469
submitting queries from 475
Designer window 443
desktop 617
Details pane 619
Diagram tab 175
dimension table lookup 545
dimension tables 524
change detection and loading for 525
change tracking 524
cross-reference tables and 526
generated keys and 525
loading with Type 1 and 2 updates 535
Type 1 updates 527
document objects
saving reports as 328
documentation 9
for process flow diagrams 155
documents
adding to registered objects 55
for columns 95
viewing contents of 57

E

empty jobs 147
encoding options 133
environment 4
ERM systems 8
error log location 21
ETL (extraction, transformation, and loading) 3
executing jobs
on remote host 236
explicit pass-through processing 484
exporting metadata 60
documenting 62
SAS Metadata Bridges 66, 75
SAS Package metadata 60, 61, 62
selected metadata 62
Expression Builder window 620
external files 120
accessing with FTP or HTTP server 133
browse and edit options for 118
common tasks 120
in job process flows 138
NLS support for 133
overriding code generated by wizards 132
registering COBOL data files 136
registering delimited files 120
registering files with user-written code 128
registering fixed-width files 124
updating metadata 131
viewing data in 135
viewing metadata 131
extraction, transformation, and loading (ETL) 3

F

fact tables 527
loading with dimension table lookup 545
structure and loading of 527
fixed-width external file wizard 124, 142
fixed-width external files
registering 124
Folders tree 24, 625
adding folders 25
adding metadata objects to folders 25
changing folder paths 26
icon overlays for metadata objects 628
renaming folders 26
foreign keys
adding 108
applying changes to tables 74
key columns 104
preserving in DBMS tables 654
restoring metadata for 75
format libraries
specifying in preprocess to job 29
formats
user-defined 28
FTP servers
accessing external files 133

G

generated code 263
  credentials in 163, 266
  displaying for jobs 267
  displaying for transformations 268
  editing for jobs 286
  for jobs 160
  jobs with generated source code 146
  LIBNAME statements and 264
  macro variables and 265
  macro variables for status handling in 219
  modifying configuration files for application servers 269
  modifying SAS start commands 269
  options for jobs 268
  options for transformations 269
  overriding, when created by External File wizards 132
  remote connection statements and 265
  replacing for jobs 287
  SYSLAST macro statement and 264
generated keys 525
generated transformations 247, 277
  impact analysis on 284, 316
  maintaining 284
  updating 285
global options 46
  for tables 87
global Options window 47
grids
  submitting jobs to 169
GROUP BY clause
  adding to queries 465

H

hash joins 482
HAVING clause
  adding to queries 465
hiding logs 305
HTTP servers
  accessing external files 133

I

I/O processing
  minimizing 480
impact analysis 312
  on generated transformations 284
  performing 313
  performing on generated transformations 316
  reverse 312, 317
implicit pass-through processing 484
implicit property for joins 482
Import and Export Metadata wizards 641
importing COBOL copybooks 136
importing metadata 60
  comparing metadata to repository 71
  invalid change analysis result 75
  SAS Metadata Bridges 66
  SAS Package metadata 60, 61, 63
importing SAS metadata
  comparing metadata to repository 73
index joins 481
indexes 109
  creating 110
  deleting 110
  deleting columns from 110
  rearranging columns in 111
  removing non-essential indexes during a load 432
INFILE statement
  replacing generated statements 132
input tables
  adding subqueries to 470
intermediate files
  deleting for performance 295
invalid change analysis results 75
Inventory tree 625
  icon overlays for metadata objects 628
invocation options
  setting on jobs 303
iterative jobs 505
  adding input and transformation directly 507
  creating and running 506
  creating control tables 512
  creating parameterized jobs 509

J

Java code
  for report plug-ins 763
Java options 20
Job Documentation Report
  customizing 325
Job Editor window 629
  making connections in 193
  submitting jobs from 167
  viewing code in 267
job management 166
job metadata
  viewing or updating 158
job options 155, 268
  global 268
  local 269
job properties
  viewing or updating 159
job properties window 631
job statistics

prerequisites for collecting 169
job status 162, 301
actions based on 216
jobs 146
accessing local and remote data 156
adding User Written Code transformation to 274
column mappings 185
common code generated for 160
creating empty jobs 147
creating jobs containing jobs 150
creating process flows for 148
credentials in generated code 163
data access in context of 156
Data Transfer transformation 158
default temporary output tables 150
deploying 226
displaying generated code for 267
displaying SAS code for 160
editing generated code for 286
external files in process flows 138
generated code for 263
interactive data access 157
iterative 505
managing submitted jobs 147
parameterized 477, 505, 509
replacing generated code for 287
reviewing successful jobs 170
running 147
SAS invocation options on 303
scope of column changes 189
status handling for 209
submitting for immediate execution 166
submitting one step at a time 169
submitting queries from 475
submitting segments of 169
submitting selected transformations 167
submitting to a grid 169
troubleshooting 175
viewing or updating metadata 158
with generated source code 146
with user-supplied source code 147
join algorithms 481
Join transformations 443
join types
changing 450
results by 451
joins 443
adding to queries on Designer tab 455
automatic 447
hash joins 482
implicit property for 482
index joins 481
joining a table to itself 476
parameters with 477
reviewing and modifying 445
selecting join type 450
sort-merge 481
SPD Server star joins 478

K

key columns 104
Key Effective Date transformation 524
keys
adding primary or unique keys 107
deleting 109
generated 525
maintaining 104
renaming 109
surrogate 300
surrogate primary 551
updating columns in 109
viewing 105

L

level_value option 21
LIBNAME statement
generated code and 264
generated job code and 161
libraries 8
registering 28
List Cluster Contents transformation 580
List Data transformation
adding to process flows 308
load techniques 429
adding rows 430
matching and updating rows 431
removing all rows 429
loader transformations 421
bulk loading 433
DB2 Bulk Table Loader transformation 426
Oracle Bulk Table Loader transformation 425
removing non-essential indexes and constraints during a load 432
selecting a load technique 429
SPD Server Table Loader transformation 422
Table Loader options 427
Table Loader transformation 424
loading output tables 421
local data
accessing 156
local options
for tables 87
Log tab 178
logs
capturing additional options 305

error log location 21
evaluating 304
for process flow optimization 304
hiding 305
message logging 21
redirecting large logs to a file 306
viewing 305
Lookup transformation 523
lookups
transformations for 300

M

macro variables
for status handling 162, 219
for status handling in generated code 219
for status handling in user-written code 224
generated code and 265
manual propagation 191
mappings 185
master data management 4
memory allocation 22
message logging 21
message queues 567
Microsoft queues 574
polling WebSphere queues 572
prerequisites 568
supported data transfer types 567
transformations for 569
WebSphere queues 570
metadata
adding 49
checking in 50
checking out 50
clearing from projects 51
connectivity and 3
deleting 51
import and export wizards 641
importing and exporting 60
maintaining column metadata 92
updating external file metadata 131
updating table metadata 84
viewing external file metadata 131
viewing or updating job metadata 158
viewing or updating stored process metadata 242
viewing table metadata 84
metadata objects
adding notes or documents to registered objects 55
adding to folders 25
copying and pasting 65
copying to folders 26
dragging to folders 26
icon overlays for 628
moving to folders 26
Microsoft message queues 574
migration 4

N

names
DBMS names 89
SAS names 88
New Job wizard 147
new object wizards 639
New Table wizard 82
registering tables with 82
New User-Written External File wizard 128
NLS
encoding options 133
support for external files 133
notes
adding to registered objects 55
for columns 95
viewing contents of 57

O

objects
connections for 193
one-to-one column mappings 187
Options window 634
Oracle
capturing changed data from 560
Oracle Bulk Table Loader transformation 422, 425
ORDER BY clause
adding to queries 468
other SQL transformations
pass-through processing 502
Output folder 41
output tables
loading in process flows 421
temporary 150

P

parallel processing 515
prerequisites for 515
setting options for 516
parameterized jobs 477, 505
creating 509
pass-through processing 484, 502
performance
data optimization 480
process flow optimization 293
sort performance 435
SQL processing 479, 486

physical tables
  managing for performance 294
  updating table metadata 85
plug-in location 21
pre-sorting data 480
Precode and Postcode tab
  adding user-written code to 272
primary keys
  adding 107
  key columns 104
  surrogate 551
process data management 294
process flow diagrams
  documenting 155
  viewing or updating 159
process flow optimization 293
  additional information 308
  column management 297
  debugging techniques 301
  logs for 304
  process data management 294
  reviewing temporary output tables 306
  streamlining process flow components 299
process flows
  adding debugging code to 302
  adding List Data transformation to 308
  adding User Written Code transformation to 308
  creating 19
  creating for jobs 148
  external files in 138
  loading output tables in 421
  scheduling for complex process flows 235
  streamlining components 299
projects
  clearing metadata from 51
propagation
  automatic 190
  manual 191
  path options 192
properties windows 631
Publish folder 41

Q

queries 443
  adding a GROUP BY clause 465
  adding a HAVING clause 465
  adding an ORDER BY clause 468
  adding CASE expressions 461
  adding columns to target table 455
  adding joins to, on Designer tab 455
  adding user-written SQL code 452
  configuring a SELECT clause 459
  creating or configuring a WHERE clause 463
  creating simple queries 457
  debugging 453
  Designer window 443
  reviewing and modifying clauses, joins, and tables 445
  submitting 475
  submitting as part of a job 475
  submitting from Designer tab 475
  subqueries 470
  validating 475

R

redeploying jobs
  for scheduling 233
  to stored processes 241
redirecting
  logs 306
  temporary output tables 307
Register Tables wizard 81, 640
  setting name options in 91
registered objects
  adding notes or documents to 55
registering
  COBOL data files 136
  control tables 512
  cubes 29
  delimited external files 120
  external files with user-written code 128
  fixed-width external files 124
  libraries 28
  tables 29, 81, 82
remote connection statements 265
  generated code and 162
remote data
  accessing 156
  minimizing access 297
remote host
  deploying jobs for execution on 236
Remove Cluster transformation 580
report plug-ins
  example Java code for 763
  reporting interface methods 769
reporting interface methods 769
reports 322
  creating 331
  customizing the Job Documentation Report 325
  customizing the Tables Report 324
  opening 329
  running and saving 326
  saving as document object 328
  viewing 329

Reports window 322, 634
  selecting a perspective 323
reverse impact analysis 312, 317
rows
  adding 430
  closing out in datetime change tracking 556
  matching and updating 431
  removing all 429
running jobs 147

S

SAS Application Servers 6, 27
SAS code
  displaying for jobs 160
SAS Data Integration Studio
  documentation 9
  environment and 5
  starting 20
SAS data servers 7
SAS Grid Server 6
SAS Intelligence Platform
  documentation 9
SAS invocation options
  setting on jobs 303
SAS Management Console
  environment and 5
SAS Metadata Bridges 66
  exporting metadata 75
  importing new metadata 68
  importing updated metadata 70
  preparing to import or export 67
  usage notes for 66
SAS Metadata Server 6
  connecting to 22
  reconnecting to 24
SAS names 88
SAS OLAP Server 6
SAS Package
  importing and exporting metadata 60, 62, 63
  preparing to import or export metadata 61
SAS Package wizards 641
SAS start commands
  modifying for application servers 269
SAS Workspace Server 6
SAS/CONNECT Server 6
SAS/SHARE Server 7
SCD
  See slowly changing dimensions (SCD)
SCD Type 2 Loader transformation 422, 523
scheduling
  deploying jobs for 227
  for complex process flows 235
  prerequisites for deploying jobs 227
  redeploying jobs for 233
  to execute jobs on remote host 236
Section 508 10
security 19
SELECT clause
  configuring 459
SELECT statement
  optimizing 487
servers 6
slowly changing dimensions (SCD) 522
  closing out rows in datetime change tracking 556
  dimension tables 524
  fact tables 527
  loading dimension tables with Type 1 and 2 updates 535
  loading fact tables with dimension table lookup 545
  loading tables and adding surrogate primary key 551
  project stages 524
  star schema loading process 524
  star schemas and 522
  tracking changes in source datetime values 554
  transformations supporting 523
  types of 523
Sort transformation 435
  creating tables containing sorted contents of a source 438
sort transformations 435
  optimizing sort performance 435
sort-merge joins 481
sorting
  performance and 435, 438
  pre-sorting data 480
source code
  jobs with generated source code 146
  jobs with user-supplied source code 147
source tables
  mapping columns to work table 276
sources
  table containing sorted contents of 438
SPD Server 7
SPD Server cluster tables 577
  creating 578
  maintaining 579
SPD Server Dynamic Cluster folder 41
SPD Server star joins 478
SPD Server Table Loader transformation 422
special characters
  in table and column names 88

SQL Join transformations
adding a GROUP BY clause 465
adding a HAVING clause 465
adding an ORDER BY clause 468
adding CASE expressions 461
adding columns to target table 455
adding joins to queries on Designer tab 455
adding subqueries 470
adding user-written SQL code 452
automatic joins 447
configuring a SELECT clause 459
creating or configuring a WHERE clause 463
creating simple queries 457
data optimization 480
debugging queries 453
Designer window 443
implicit property for a join 482
join algorithms 481
joining a table to itself 476
optimizing processing performance 479, 486
parameters with joins 477
pass-through processing 484
selecting join type 450
SPD Server star joins 478
submitting queries 475
validating queries 475
SQL Properties window
options for optimizing performance 486
star joins 478
star schemas
loading process 524
slowly changing dimensions and 522
transformations for 300
starting SAS Data Integration Studio 20
Statistics tab 171
status codes
setting and checking 303
status handling
actions based on 216
actions based on transformation status 217
default conditions, actions, and conditional action sets 210
for jobs and transformations 209
in generated code 219
in user-written code 224
macro variables for 162, 219
prerequisites for actions 215
Status tab 170, 176
stored process server 6
stored processes
deploying as Web service 251
deploying jobs as 237, 238
deploying Web service jobs as 249
prerequisites for 238
prerequisites for deploying jobs as 237
redeploying jobs to 241
viewing or updating metadata 242
subqueries 470
adding to clauses 473
adding to input tables 470
Surrogate Key Generator transformation 524
surrogate keys 300
surrogate primary keys 551
synchronization 4
SYSLAST macro statement
generated code and 264
job code and 161

T

Table Loader transformation 422, 424
setting options 427
table metadata
updating with physical table 85
viewing or updating 84
table names
case and special characters in 88
default name options 91
name options for registered tables 92
table properties window 632
tables 80
browse and edit options for 118
browsing data 111
bulk load 486
CDC control tables 559
changed data tables 559
common tasks 80
containing sorted contents of a source 438
control tables 512
creating with View Data window 117
cross-reference tables 526
default temporary output tables 150
dimension tables 524, 535
editing data 114
fact tables 527, 545
global options for 87
indexes 109
joining a table to itself 476
key columns 104
loading and adding surrogate primary key 551
local options for 87
managing physical tables for performance 294
registering 29
registering with New Table wizard 82

registering with Register Tables wizards 81
reviewing and modifying, in queries 445
setting default name options 91
source tables 276
SPD Server cluster tables 577
specifying options for 86
target tables 276
temporary output tables 306
updating metadata with physical table 85
work tables 276
Tables Report
customizing 324
target tables
adding columns to 455
mapping columns from work table 276
temporary output tables 150
preserving 307
redirecting 307
reviewing 306
viewing 307
testing code 128
threaded reads 487
Tools-Options window 634
tracking
datetime change tracking 554, 556
transformation options 269
transformation properties window 632
transformations 32
actions based on status of 217
adding directly to iterative jobs 507
Change Data Capture 557
Data Transfer 158
DB2 Bulk Table Loader 422, 426
displaying generated code for 268
editing generated code for 286
for lookups 300
for message queues 569
for star schemas 300
generated 277, 284, 316
Join 443
Key Effective Date 524
limiting input 302
List Cluster Contents 580
List Data 308
loader transformations 421
Lookup 523
Oracle Bulk Table Loader 422, 425
Remove Cluster 580
replacing generated code for 287
SCD Type 2 Loader 422, 523
Sort transformations 435
SPD Server Table Loader 422
status handling for 209
submitting selected transformations 167
supporting slowly changing dimensions 523
Surrogate Key Generator 524
Table Loader 422, 424
User Written Code 274, 308
verifying output 302
viewing code for 195
Websphere Queue Reader 571, 572
Websphere Queue Writer 570, 571
Transformations tree 32
tree view 635
troubleshooting unsuccessful jobs 175
Type 1 updates 527, 535
Type 2 updates 535

U

undoing checkouts 51
Ungrouped folder 43
unique keys
adding 107
key columns 104
update table metadata feature 85
updates
Type 1 527, 535
Type 2 535
User Written Code transformation
adding to process flows 308
user-defined formats 28
user-supplied source code 147
user-written code 271
adding to Precode and Postcode tab 272
editing generated code for jobs or transformations 286
generated transformations and 277
macro variables for status handling in 224
maintaining generated transformations 284
replacing generated code for jobs or transformations 287
SQL 452, 488
User-Written Code transformation
adding to jobs 274
user-written external files
registering 128

V

validating data 297
View Data window 637
creating tables 117
View File window 638
views
managing for performance 294

W

Warnings and Errors tab 177
Web services
  creating jobs 244
  deploying jobs as 243
  deploying jobs as stored process 249
  deploying stored processes as 251
  prerequisites for deploying jobs 243
  requirements for deploying jobs 244
WebSphere message queues 570
  polling 572
Websphere Queue Reader transformation
  configuring and running jobs 571
  creating jobs 571
  verifying jobs 572
Websphere Queue Writer transformation
  configuring and running jobs 571
  creating jobs 570
  verifying jobs 571
WHERE clause
  creating or configuring 463
wizards 639
work tables
  mapping columns from source table 276
  mapping columns to target table 276