sas THE POWER TO KNOW.®

SAS® 9.2 Intelligence Platform Data Administration Guide

SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2009. *SAS® 9.2 Intelligence Platform: Data Administration Guide*. Cary, NC: SAS Institute Inc.

SAS® 9.2 Intelligence Platform: Data Administration Guide

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

ISBN-13: 978-1-59994-313-8

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

1st electronic book, February 2009

1st printing, March 2009

Contents

What’s New

Overview

The *SAS Intelligence Platform: Data Administration Guide* focuses on the SAS Intelligence Platform and third-party products that you need to install and the metadata objects that you need to create in order to establish connectivity to your data sources (and data targets). It also contains information about setting up shared access to SAS data and explains how using different data-access engines affects security.

New Data Surveyors

In previous releases, SAS integrated data from ERP and CRM systems by accessing the underlying database with SAS/ACCESS software. In this release, SAS has partnered with Composite Software to provide data integration with ERP and CRM systems from PeopleSoft, Oracle Applications, Siebel, and Salesforce.com. These new Data Surveyors access data by using the vendor API and certified interfaces, and they comply with the security of the application. A detailed example of connecting to Salesforce.com is provided.

Documentation Enhancements

The following enhancements were made for this release:

- added more information about SPD Server and dynamic clusters.
- added information about Change Data Capture.
- added information about application response monitoring logging.
- added more information about establishing connectivity to Oracle.
- removed information about registering database schemas because the SAS Management Console no longer has a database schema wizard. Schemas are associated with a data library by typing the schema name in a text field.
- removed the task for making column labels available for drill-through tables on an OLAP server. This task is not necessary in this release.

- added information about setting libraries to be read-only for reporting applications.

CHAPTER 1 Overview of Common Data Sources

Overview

This chapter describes the features of the most common data sources that you encounter as you perform administrative tasks. In addition, a simple diagram is provided for each data source that shows how the data flows as connections are established between source storage, SAS engines and servers, and SAS applications.

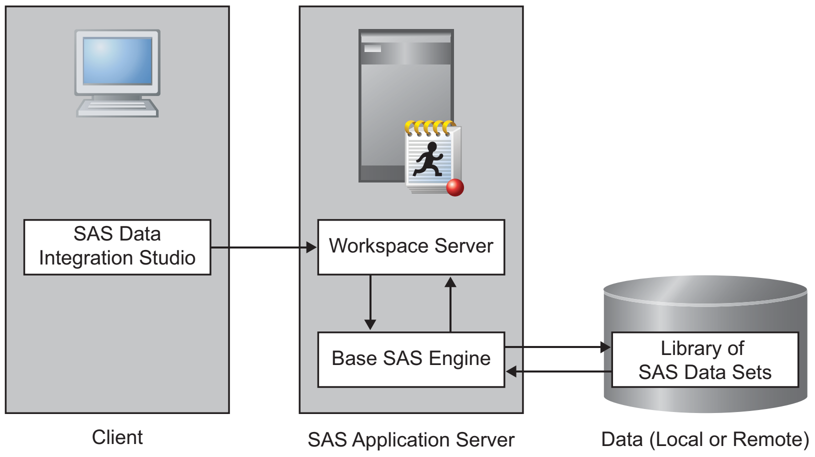
Accessibility Features in the SAS Intelligence Platform Products

For information about accessibility for any of the products mentioned in this book, see the documentation for that product. If you have questions or concerns about the accessibility of SAS products, send e-mail to **accessibility@sas.com**.

SAS Data Sets

SAS data sets (tables) are the default SAS storage format. You can use them to store data of any granularity. A SAS table is a SAS file stored in a SAS library that SAS creates and processes. A SAS table contains data values that are organized as a table of observations (rows) and variables (columns) that can be processed by SAS software. A SAS table also contains descriptor information such as the data types and lengths of the columns, as well as which engine was used to create the data. For more information about using default SAS storage, see *SAS Language Reference: Concepts* and *SAS Language Reference: Dictionary*. The following figure shows how connectivity to SAS data sets is configured.

Figure 1.1 Establishing Connectivity to SAS Data Sets



For a detailed example of a SAS data set connection, see "Establishing Connectivity to a Library of SAS Data Sets" on page 19.

Shared Access to SAS Data Sets

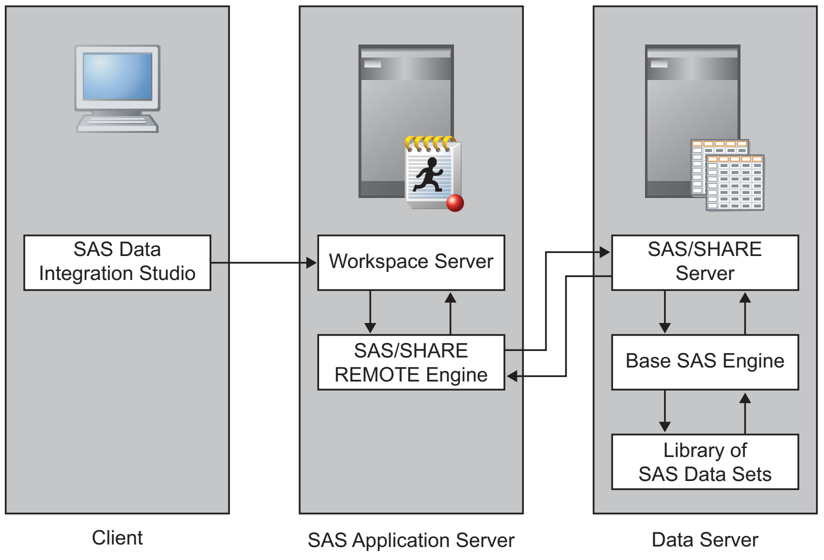
SAS/SHARE software provides concurrent update access to SAS files for multiple users. SAS/SHARE is often required for transaction-oriented applications where multiple users need to update the same SAS data sets at the same time. Data entry applications where multiple users are entering data to the same data set are a good example of this type of usage. SAS/SHARE software provides both member-level locking and record-level locking. Therefore, two or more users can update different observations within the same data set, and other users can print reports from the same data set.

SAS/SHARE supports multi-user read and write access to both SAS data files and SAS catalogs. Multi-user access to SAS catalogs simplifies the maintenance of applications by allowing users and developers to share the same program libraries. Users can execute applications at the same time that developers update the source programs.

SAS/SHARE software also acts as a data server that delivers data to users for their processing needs. This capability provides data administrators both a centralized point of control for their data and a secure environment to control who accesses the data. SAS/SHARE is also designed to be a reliable data server that functions as long as the system that the server is running on is operational.

Finally, SAS/SHARE enables you use SAS software to define views of your data. This allows administrators to restrict certain users to subsets of data for security or efficiency purposes. Access to rows and columns in SAS tables can be defined using this technique. The following figure shows shared access to SAS data sets. Note that the data server in the figure can be a different operating system and architecture from the SAS Application Server, if the site is licensed for that configuration.

Figure 1.2 Establishing Shared Access to SAS Data Sets

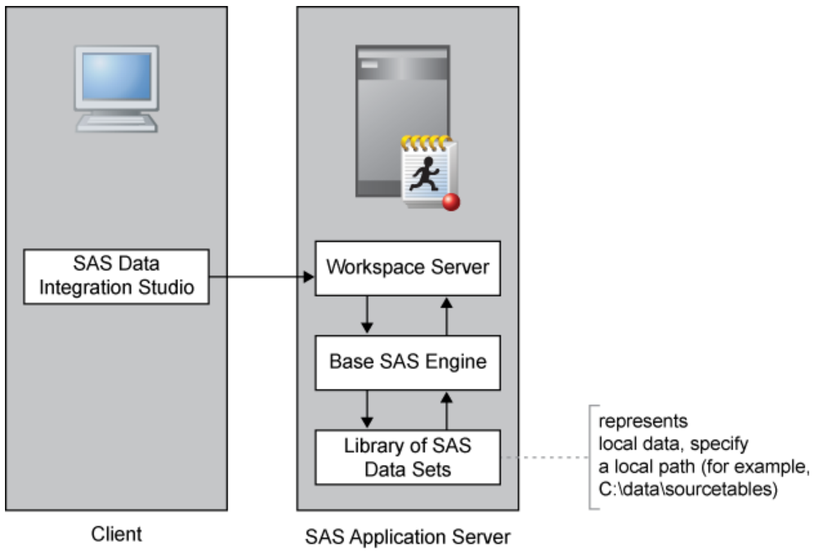


For a detailed example of a shared SAS data set connection, see "Establishing Shared Access to SAS Data Sets" on page 22.

Local and Remote Access to Data

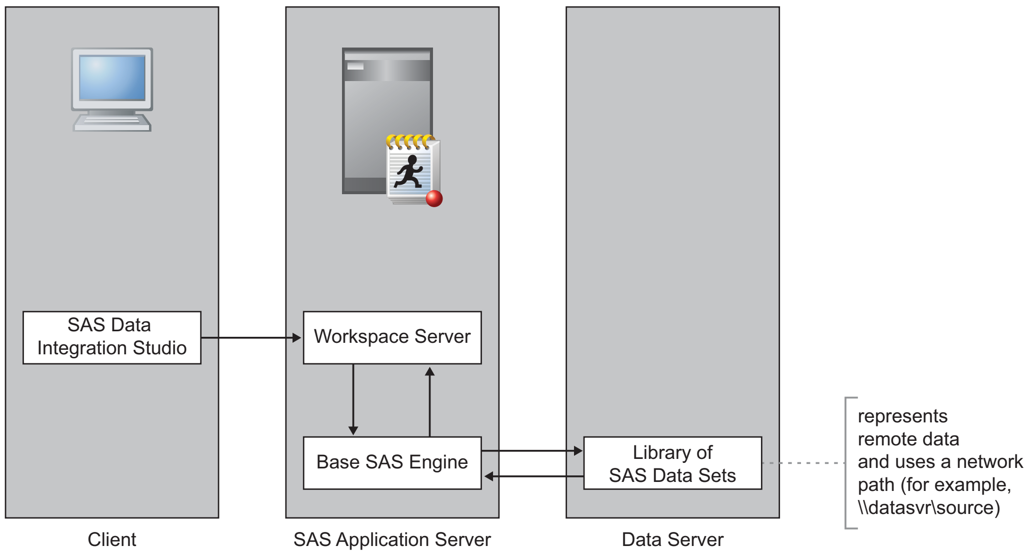
To access data you must register the data as a library in SAS Management Console. The procedures for accessing data and registering data are explained later in this document. However, one of the important details for file-based data, such as SAS data sets, is that you need to specify the file system path to the data. This path is needed so a SAS Application Server can access it. As shown in the following figure, SAS data sets that are local to the SAS Application Server have a fully qualified path such as **`C:\data\sourcetables`**:

Figure 1.3 SAS Workspace Server Accessing Local Data Sets


SAS Data Integration Studio
Workspace Server
Base SAS Engine
Library of SAS Data Sets
represents local data, specify a local path (for example, C:\data\sourcetables)
Client
SAS Application Server

Often, file-based data is stored on a host that is remote from the SAS Application Server. When the hosts have a network path for shared directories such as a Windows UNC path or UNIX NFS, then that path is used. The following figure shows an example of a SAS Workspace Server accessing a UNC path, `\\dataserver\sourcetables`, on a data server.

Figure 1.4 SAS Workspace Server Accessing Remote Data Sets


SAS Data Integration Studio
Workspace Server
Base SAS Engine
Library of SAS Data Sets
represents remote data and uses a network path (for example, \\datasvr\source)
Client
SAS Application Server
Data Server

Note: This figure shows a SAS Workspace Server accessing data over a shared file system. To access data over network connection (without the file system), use SAS/SHARE as described in this document. △

External Files

An external file is a file that is maintained by the machine operating environment or by a software product other than SAS. A flat file with comma-separated values is one example. SAS Data Integration Studio provides three source designer wizards that enable you to create metadata objects for external files:

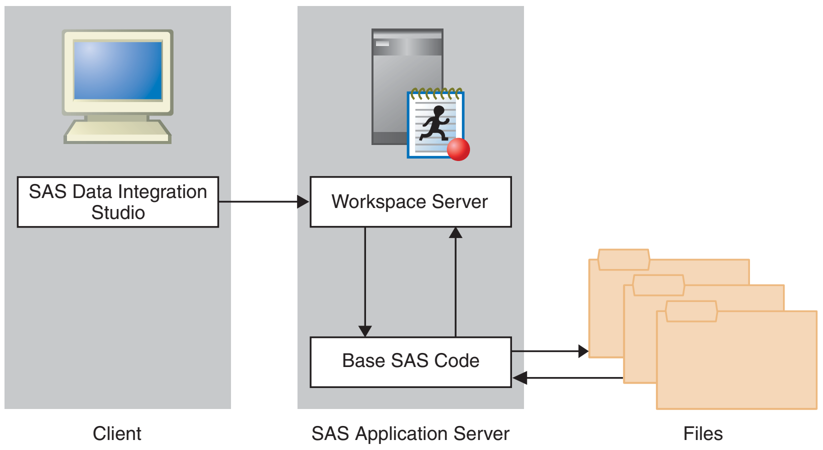
- the delimited external file wizard for external files in which data values are separated with a delimiter character. This wizard enables you to specify multiple delimiters, nonstandard delimiters, missing values, and multi-line records.
- the fixed-width external file wizard for external files in which data values appear in columns that are a specified number of characters wide. This wizard enables you to specify non-contiguous data.
- the user-written external file wizard for complex external files that require user-written SAS code to access their data.

The external file source designer wizards enable you to do the following:

- display a raw view of the data in the external file
- display a formatted view of the data in the external file, as specified in the SAS metadata for that file
- display the SAS DATA step and SAS INFILE statement that the wizard generates for the selected file
- display the SAS log for the code that is generated by the wizard
- specify options for the SAS INFILE statement that is generated by the wizard, such as National Language Support (NLS) encoding
- override the generated SAS INFILE statement with a user-written statement
- supply a user-written SAS DATA step to access an external file

The following figure shows establishing connectivity to external files:

Figure 1.5 Establishing Connectivity to External Files



For a detailed example of an external file connection, see "Establishing Connectivity to a Flat File" on page 32.

XML Data

The XML LIBNAME engine works in a way similar to other SAS engines. A LIBNAME statement is executed so that a libref is assigned and an engine is specified. That libref is then used throughout the SAS session.

Instead of the libref being associated with the physical location of a SAS library, the libref for the XML engine is associated with a physical location of an XML document. When you use the libref that is associated with an XML document, SAS either translates the data in a SAS data set into XML markup or translates the XML markup into SAS format.

The XML LIBNAME engine can read input streams from a Web service input and write an output stream to a Web service output. The XML LIBNAME engine supports reading XML files in complex structures using XMLMaps. An XMLMap is a user-defined file that contains XML tags that tell the XML LIBNAME engine how to interpret an XML document. XMLMaps are defined using the SAS XML Mapper product. For additional information, see the *SAS XML LIBNAME Engine User's Guide*.

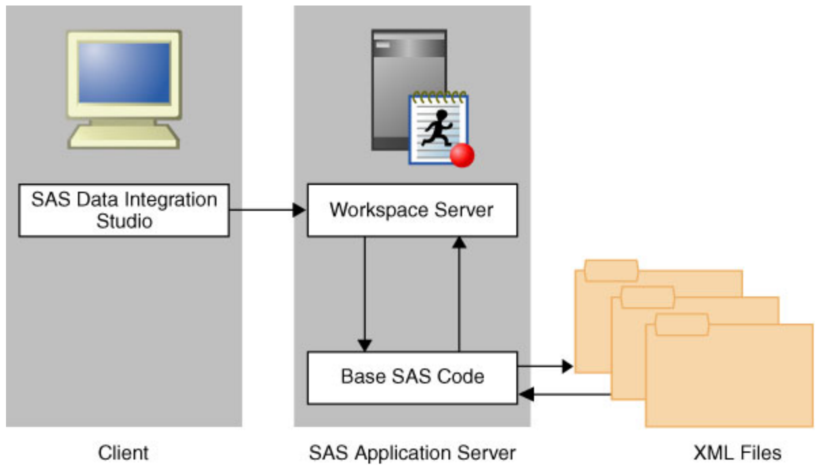
XML files are written by the XML Writer transformation provided by SAS Data Integration Studio. The XML LIBNAME engine supports Output Delivery System (ODS) tag sets; XMLMaps are not supported for writing. The XML Writer transformation in SAS Data Integration Studio ships with a sample ODS tag set, if needed. An output XML document can either be:

- □ used by a product that processes XML documents
- □ moved to another host for the XML LIBNAME engine to process by translating the XML markup back to a SAS data set

Because the XML LIBNAME engine is designed to handle tabular data, all the data sent to or from a Web service must be in table form.

The following figure shows connectivity to XML files:

Figure 1.6 Establishing Connectivity to XML Files



Message Queues

Message queues are collections of data objects that enable asynchronous communication between processes. These processes are typically applications that run on different computers, and might be configured in a heterogenous network. Queue management software ensures that messages are transmitted without error. SAS Data Integration Studio can perform messaging jobs to read and write messages to Microsoft MSMQ as well as IBM WebSphere MQ. For more information about administering message queues, see *SAS Intelligence Platform: Desktop Application Administration Guide*. For more information about creating messaging jobs, see *SAS Data Integration Studio: User's Guide*.

Relational Database Sources

SAS/ACCESS

Data also can be stored in third-party hierarchical and relational databases such as DB2, Oracle, SQL Server, and Teradata. SAS/ACCESS interfaces provide fast, efficient reading and writing of data to these facilities.

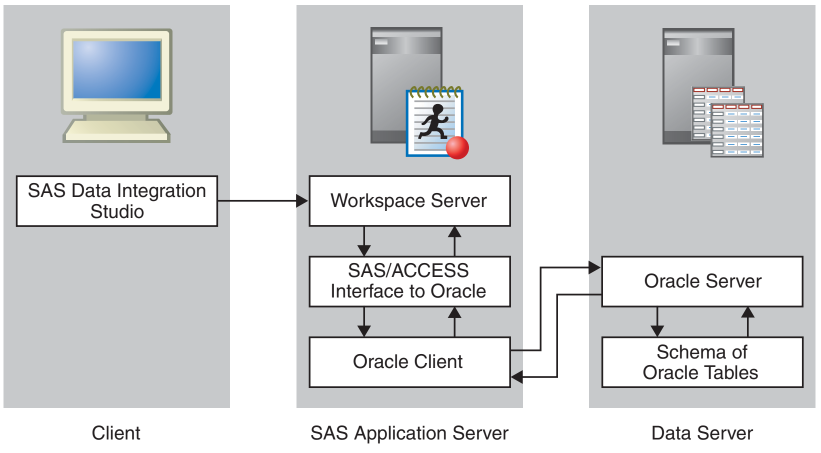
Several of the SAS/ACCESS engines support threaded reads. This enables you to read entire blocks of data on multiple threads instead of reading data just one record at a time. This feature can reduce I/O bottlenecks and enables thread-enabled procedures to read data quickly. These engines and DB2 on z/OS also have the ability to access database management system (DBMS) data in parallel by using multiple threads to the parallel DBMS server.

The following SAS/ACCESS engines support this functionality:

- Oracle
- Sybase
- DB2 (UNIX and PC)
- SQL Server
- Teradata

For more information about using the SAS/ACCESS interfaces, see *SAS/ACCESS for Relational Databases: Reference*. The following figure shows how connectivity to Oracle databases is configured:

Figure 1.7 Establishing Connectivity to Oracle Databases



For a detailed example of an Oracle connection, see "Establishing Connectivity to an Oracle Database" on page 37.

ODBC Sources

Open database connectivity (ODBC) standards provide a common interface to a variety of databases such as DB2, Microsoft Access, Oracle, and Microsoft SQL Server databases. Specifically, ODBC standards define application programming interfaces (APIs) that enable an application to access a database if the ODBC driver complies with the specification.

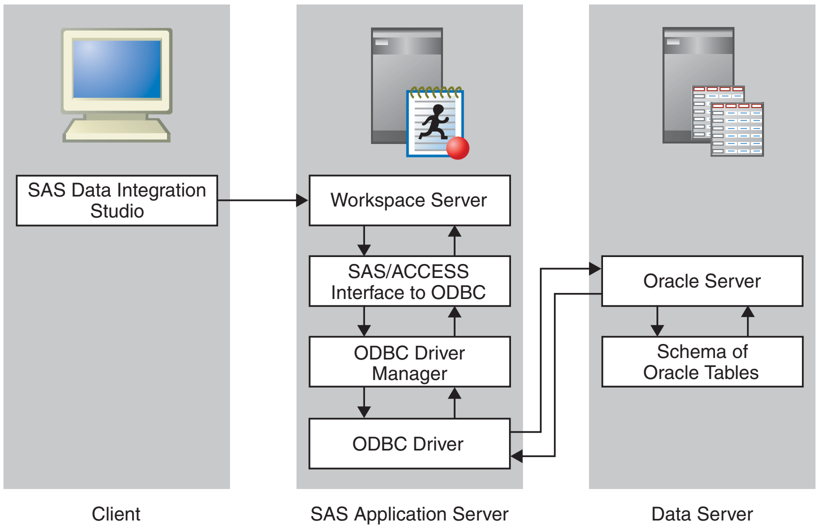
Note: If a SAS/ACCESS engine is available for a database, then performance is better with the SAS/ACCESS engine rather than with the ODBC interface. △

The basic components and features of ODBC include the following:

- ODBC functionality is provided by three components: the client interface, the ODBC driver manager, and the ODBC driver. SAS provides the SAS/ACCESS interface to ODBC, which is the client interface. For PC platforms, Microsoft developed the ODBC Administrator, which is used from the Windows Control Panel to perform software administration and maintenance activities. The ODBC driver manager also manages the interaction between the client interface and the ODBC driver. On UNIX platforms, a default ODBC driver manager does not exist and SAS does not provide a driver manager with SAS/ACCESS to ODBC. For UNIX platforms, you should obtain an ODBC driver manager from your ODBC driver vendor.
- The ODBC administrator defines a data source as the data that is used in an application and the operating system and network that are used to access the data. You create a data source by using the ODBC Administrator in the Windows Control Panel and then selecting an ODBC driver. You then provide the information (for example, data source name, user ID, password, description, and server name) that is required by the driver to make a connection to the desired data. The driver displays dialog boxes in which you enter this information. During operation, a client application usually requests a connection to a named data source, not just to a specific ODBC driver.
- An ODBC Administrator tool is not available in a UNIX environment such as HP-UX, AIX, or Solaris. During an install, the driver creates a generic **`.odbc.ini`** file that can be edited to define your own data sources.

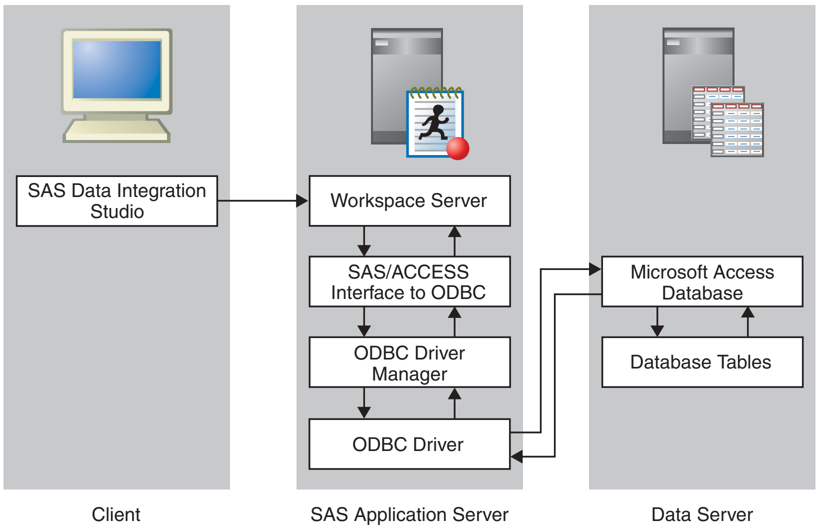
The following figure shows how ODBC is used to establish connectivity to Oracle databases:

Figure 1.8 Establishing Connectivity to Oracle Databases by Using ODBC



For a detailed example of an ODBC-based Oracle connection, see "Establishing Connectivity to an Oracle Database by Using ODBC" on page 41. The following figure shows how ODBC is used to establish connectivity to Access databases:

Figure 1.9 Establishing Connectivity to Access Databases by Using ODBC



For a detailed example of an ODBC-based Access connection, see "Establishing Connectivity to a Microsoft Access Database by Using ODBC" on page 45.

Scalable Performance Data Server and Scalable Performance Data Engine

Overview of Scalable Performance Data Server and Scalable Performance Data Engine

Both the SAS Scalable Performance Data Engine (SPD Engine) and the SAS Scalable Performance Data Server (SPD Server) are designed for high-performance data delivery. They enable rapid access to SAS data for intensive processing by the application. The SAS SPD Engine and SAS SPD Server deliver data to applications rapidly by organizing the data into a streamlined file format that takes advantage of multiple CPUs and I/O channels to perform parallel input and output functions.

The SAS SPD Engine is included with Base SAS software. It is a single-user data storage solution that shares the high-performance parallel processing and parallel I/O capabilities of SAS SPD Server, but it lacks the additional complexity of a full-blown server. The SAS SPD Server is available as a separate product or as part of the SAS Intelligence Storage bundle. It is a multi-user parallel-processing data server with a comprehensive security infrastructure, backup and restore utilities, and sophisticated administrative and tuning options. SAS SPD Server libraries can be defined using SAS Management Console.

SAS SPD Engine and SAS SPD Server use multiple threads to read blocks of data very rapidly and in parallel. The software tasks are performed in conjunction with an operating system that enables threads to execute on any of the machine's available CPUs.

Although threaded I/O is an important part of both product offerings' functionality, their real power comes from the way that the software structures SAS data. They can read and write partitioned files and, in addition, use a specialized file format. This data structure permits threads, running in parallel, to perform I/O tasks efficiently.

Although not intended to replace the default Base SAS engine for most tables that do not span volumes, SAS SPD Engine and SAS SPD Server are high-speed alternatives for processing very large tables. They read and write tables that contain billions of observations.

The SAS SPD Engine and SAS SPD Server performance are boosted in these ways:

- support for terabytes of data
- scalability on symmetric multiprocessing (SMP) machines
- parallel WHERE selections
- parallel loads
- parallel index creation
- partitioned tables
- parallel I/O data delivery to applications
- implicit sorting on BY statements

The SAS SPD Engine runs on UNIX, Windows, z/OS (on HFS and zFS file systems only), and OpenVMS for Integrity Servers (on ODS-5 file systems only) platforms. The SAS SPD Server runs on Tru64 UNIX, Windows Server, HP-UX, and Sun Solaris platforms.

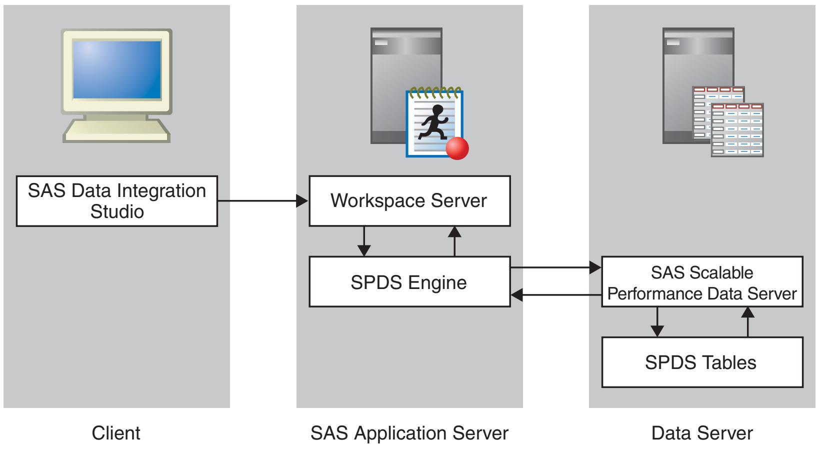
Symmetric Multiprocessing

The SAS SPD Server exploits a hardware and software architecture known as symmetric multiprocessing (SMP). An SMP machine has multiple CPUs and an operating system that supports threads. An SMP machine is usually configured with multiple disk I/O controllers and multiple disk drives per controller. When the SAS SPD Server reads a data file, it launches one or more threads for each CPU; these threads then read data in parallel. By using these threads, a SAS SPD Server that is running on an SMP machine provides the quick data access capability that is used by SAS in an application.

For more information about using the SAS SPD Server, see *SAS Scalable Performance Data Server: Administrator's Guide* and **support.sas.com/rnd/scalability/spds**.

The following figure shows how connectivity to SPD Servers is established:

Figure 1.10 Establishing Connectivity to a SAS SPD Server



For a detailed example of a SAS SPD Server connection, see "Establishing Connectivity to a Scalable Performance Data Server" on page 48.

Dynamic Clustering

The SAS SPD Server provides a virtual table structure called a clustered data table. A cluster contains a number of slots, each of which contains a SAS SPD Server table. The clustered data table uses a layer of metadata to manage the slots.

This virtual table structure provides the SAS SPD Server with the architecture to offer flexible storage to allow a user to organize tables based on values contained in numeric columns, including SAS date, time, or datetime values. This new type of organization is called a dynamic cluster table. Dynamic cluster tables enable parallel loading and selective removal of data from very large tables, making management of large warehouses easier. These unique capabilities provide organizational features and performance benefits that traditional SAS SPD Server tables cannot provide.

Dynamic cluster tables can load and process data in parallel. Dynamic cluster tables provide the flexibility to add new data or to remove historical data from the table by accessing only the slots affected by the change, without having to access the other slots, thus reducing the time needed for the job to complete. Additionally, a complete refresh of a dynamic cluster table requires a fraction of the disk space that would otherwise be needed, and can be divided into parallel jobs to complete more quickly. All of these benefits can be realized using simple SPDO procedure commands to create and alter a cluster.

The two most basic commands are CLUSTER CREATE and CLUSTER UNDO. Two additional commands are ADD and LIST. You execute each of these commands within PROC SPDO.

The CLUSTER CREATE command requires three options:

□ the name of the cluster table (cluster-table-name) that will be created

□ a list of SAS Scalable Performance Data Server tables that will be included in the cluster (using the MEM= option)

□ the number of slots (using the MAXSLOT= option), for member tables, that the cluster will have

The following example shows the syntax for PROC SPDO with a CLUSTER CREATE command:

```
PROC SPDO LIBRARY=domain-name;
SET ACLUSER user-name;
CLUSTER CREATE cluster-table-name
MEM = SPD-Server-table1
MEM = SPD-Server-table2
MEM = SPD-Server-table3
MEM = SPD-Server-table4
MEM = SPD-Server-table5
MEM = SPD-Server-table6
MEM = SPD-Server-table7
MEM = SPD-Server-table8
MEM = SPD-Server-table9
MEM = SPD-Server-table10
MEM = SPD-Server-table11
MEM = SPD-Server-table12
MAXSLOT=24;
QUIT;
```

Here is the syntax for the UNDO command:

```
PROC SPDO LIBRARY=domain-name;
SET ACLUSER user-name;
CLUSTER UNDO sales_hist;
QUIT;
```

This example shows the syntax for the ADD command:

```
PROC SPDO LIBRARY=domain-name;
SET ACLUSER user-name;
CLUSTER ADD sales_hist
MEM = 2005sales_table1
MEM = 2005sales_table2
MEM = 2005sales_table3
MEM = 2005sales_table4
MEM = 2005sales_table5
MEM = 2005sales_table6;
QUIT;
```

Finally, here is the syntax for the LIST command:

```
PROC SPDO LIBRARY=domain-name;
SET ACLUSER user-name;
CLUSTER LIST sales_hist;
QUIT;
```

These operations run quickly. These features reduce the downtime of the table for maintenance and improve the availability of the warehouse.

ERP and CRM Systems

Overview of ERP and CRP Systems

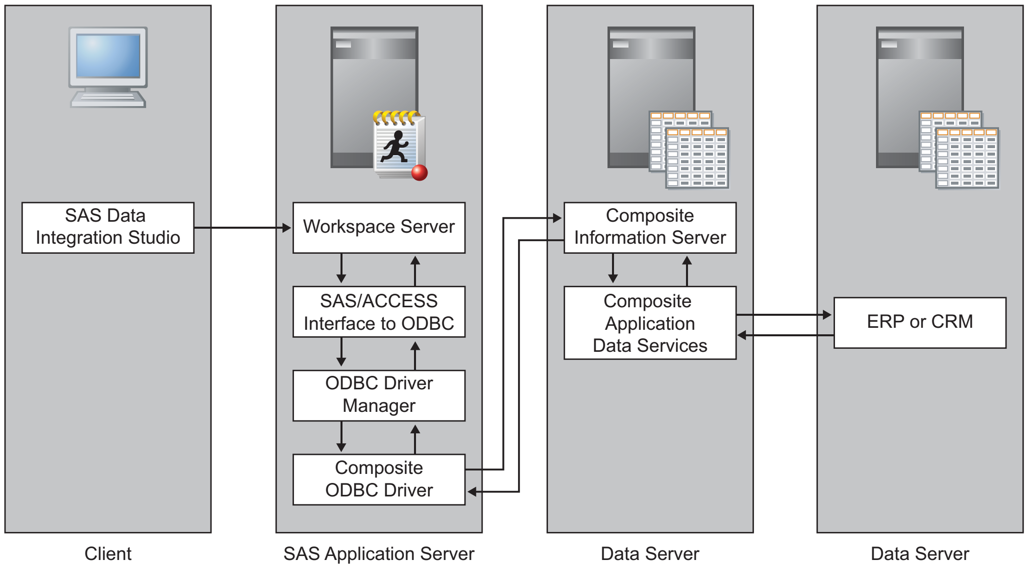
Enterprise Resource Planning (ERP) and Customer Relationship Management (CRM) systems contain a wealth of data in tables, columns, variables, and fields, but they lack several key features:

- □ the ability to provide integration with other data sources
- □ the ability to do backward-looking drill-down analysis into what caused the effect (Business Intelligence)
- □ the ability to do forward-looking cause and effect analysis (Business Analytics)

New Data Surveyors

Previously, SAS provided data surveyors that relied on accessing the underlying database—Oracle, DB2, and SQL Server—and not the application APIs. SAS now provides, through software from Composite Software, both Service Oriented Architecture (SOA) and SQL data services that unlock the data in PeopleSoft, Oracle Applications, Siebel, as well as the recently offered Salesforce.com. The following figure shows how SAS interacts with Composite Software:

Figure 1.11 Establishing Connectivity Using Composite Software


SAS Data Integration Studio
Workspace Server
SAS/ACCESS Interface to ODBC
ODBC Driver Manager
Composite ODBC Driver
Composite Information Server
Composite Application Data Services
ERP or CRM
Client
SAS Application Server
Data Server
Data Server

The Composite Information Server uses a Data Service to access a data source through the data source’s API. The Composite Information Server then offers the data through an ODBC interface. You configure an ODBC data source name on the SAS Application Server with the Composite ODBC driver. Then you use SAS Management Console to register an ODBC server and an ODBC library. For a detailed example of a Composite Information Server connection to Salesforce.com, See "Establishing Connectivity to a Composite Information Server" on page 24.

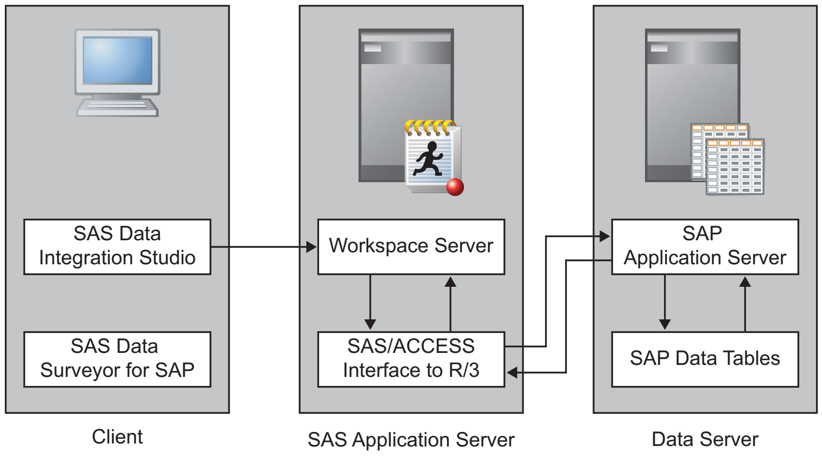
Data Surveyor for SAP

The Data Surveyor for SAP remains as in previous versions. It contains Java plug-ins to SAS Data Integration Studio and SAS Management Console, plus the required SAS/ACCESS engine necessary to get the information out of the DBMS system. Understanding the metadata of these business applications is at the heart of the data surveyor. The SAP Data Surveyor has knowledge about the structure of the tables deployed in SAP. This knowledge contains information about the ERP metadata that allows you to do the following:

- □ understand complex data structures
- □ navigate the large amounts of tables (SAP has over 20,000)

The following figure shows how connectivity to SAP servers is established:

Figure 1.12 Establishing Connectivity to an SAP Server



For a detailed example of an SAP server connection, see "Establishing Connectivity to an SAP Server" on page 51.

Change Data Capture

Data extraction is an integral part of all data warehousing projects. Data is often extracted on a nightly or regularly scheduled basis from transactional systems in bulk and transported to the data warehouse. Typically, all the data in the data warehouse is refreshed with data extracted from the source system. However, an entire refresh involves the extraction and transportation of huge volumes of data and is very expensive in both resources and time. With data volumes now doubling yearly in some organizations a new mechanism known as change data capture (CDC) is increasingly becoming the only viable solution for delivering timely information into the warehouse to make it available to the decision makers. CDC is the process of capturing changes made at the data source and applying them throughout the enterprise. CDC minimizes the resources required for ETL processes because it deals only with data changes. The goal of CDC is to ensure data synchronicity. SAS offers a number of CDC options.

- Some database vendors (Oracle 10g) provide tables of just changed records. These tables can be registered in SAS Data Integration Studio and used in jobs to capture changes.
- SAS Data Integration Studio allows the user to determine changes and take appropriate action.
- SAS has partnered with Attunity, a company that specializes in CDC. Their Attunity Stream software provides agents that non-intrusively monitor and capture changes to mainframe and enterprise data sources such as VSAM, IMS, Adabas, DB2, and Oracle. SAS Data Integration Studio provides a dedicated transformation for Attunity.

The Attunity-based solution does the following:

- moves only CHANGES to the data
- requires no window of operation
- provides higher frequency and reduced latency transfers. It is possible for multiple updates each day, providing near-real-time continuous change flow.
- reduces the performance impact of the following activities:
  - rebuilding of target table indexes
  - recovering from a process failure that happens mid-stream

DataFlux Integration Server and SAS Data Quality Server

Certain enterprise software bundles for the SAS Intelligence Platform include data quality software from SAS and from DataFlux (a SAS company). The data quality software enables you to analyze, standardize, and transform your data to increase the accuracy and value of the knowledge that you extract from your data.

The data quality product from SAS is SAS Data Quality Server, which consists of SAS language elements and a Quality Knowledge Base from DataFlux. The language elements analyze and cleanse data by referencing data definitions in the Quality Knowledge Base. SAS Data Quality Server also provides a SAS language interface to the DataFlux Integration Server.

The data quality software from DataFlux consists of the DataFlux Integration Server, a second Quality Knowledge Base, and the dfPower Studio software. The DataFlux Integration Server runs jobs and real-time services that are created in dfPower Studio. The jobs and real-time services can be executed by SAS programs that contain the procedures and functions in SAS Data Quality Server. Among its many capabilities, the dfPower Studio software enables you to create jobs and real-time services and customize the data definitions in Quality Knowledge Bases.

SAS Data Integration Studio provides enabling software for data quality applications. Four data quality transformations enable you to analyze data, cleanse data, or trigger the execution of DataFlux jobs or real-time services on DataFlux Integration Servers.

The data quality software from SAS and DataFlux requires setup and configuration after installation. For administrative information, see “Administering SAS Data Integration Studio” in the *SAS Intelligence Platform: Desktop Application Administration Guide*.

CHAPTER

2

Connecting to Common Data Sources

Overview of Connecting to Common Data Sources

This chapter consists of detailed examples for establishing a connection to each of the common data sources introduced in Chapter 1, "Overview of Common Data Sources," on page 1. Some of the connection processes covered in this chapter have common elements that might be applied to similar data sources. For example, the description of the process of using SAS/ACCESS to connect to an Oracle database might be useful when you connect to other relational databases such as DB2, Sybase, and Informix. Also, the descriptions of ODBC connections to Oracle and Microsoft Access databases and the account of the connection to an SAP source can be helpful when you connect to similar data sources.

In order to perform the procedures for registering libraries, you must have ReadMetadata and WriteMetadata permission for the repository and the SAS Application Servers that the data library is assigned to.

This chapter also explains the process that registers tables as metadata from the data sources. Registering a table in metadata enables you to view the data in a SAS application. For more information about managing table metadata, see Chapter 4, "Managing Table Metadata," on page 71.

Overview of SAS/ACCESS Connections to RDBMS

This section provides generic instructions for using SAS Management Console to configure access to a database. SAS/ACCESS must be licensed and configured before using SAS Management Console to register the library that holds the tables. The generic procedure has two stages:

1 Register the DBMS server.
2 Register the DBMS library.

Register the DBMS Server

To register a DBMS server, perform the following steps:

1 Right-click **Server Manager** and select the **New Server** option to access the New Server wizard.

2 Select the database server type from the **Database Servers** list. Then, click **Next**.

3 Enter an appropriate server name in the **Name** field. Click **Next**.

4 Accept the defaults for the server properties. Click **Next**.

5 Specify the database vendor-specific values on the connection properties page. If the user credentials for the database are different from the credentials used to log in to SAS, then you must create an Authentication domain to store valid database credentials. For more information, see "How to Store Passwords for a Third-Party Server" in the *SAS Intelligence Platform: Security Administration Guide*.

   Click **Next**.

6 Examine the final page of the wizard to ensure that the proper values have been entered. Click **Finish** to save the wizard settings.

Register the DBMS Library

To register a DBMS library, perform the following steps:

1 In SAS Management Console, expand **Data Library Manager**. Right-click **Libraries** and select the **New Library** option to access the New Library wizard.

2 Select the database type from the **Database Data** list. Click **Next**.

3 Enter an appropriate library name in the **Name** field. Click **Next**.

4 Select an application server from the list, and use the right arrow to assign the application server. This step makes the library available to the server and makes the library visible to users of the server. Click **Next**.

5 Specify a libref on the library properties page. You can also click **Advanced Options** to perform tasks such as pre-assignment. Pre-assigning a library is valuable if your clients include SAS Enterprise Guide or SAS Add-In for Microsoft Office. For more information, see Chapter 3, "Assigning Libraries," on page 59. Click **Next** to access the next page of the wizard.

6 On the server and connection page, select the database server from the previous stage. Contact your database administrator if you are unsure of the correct value for the schema field. Click **Next**.

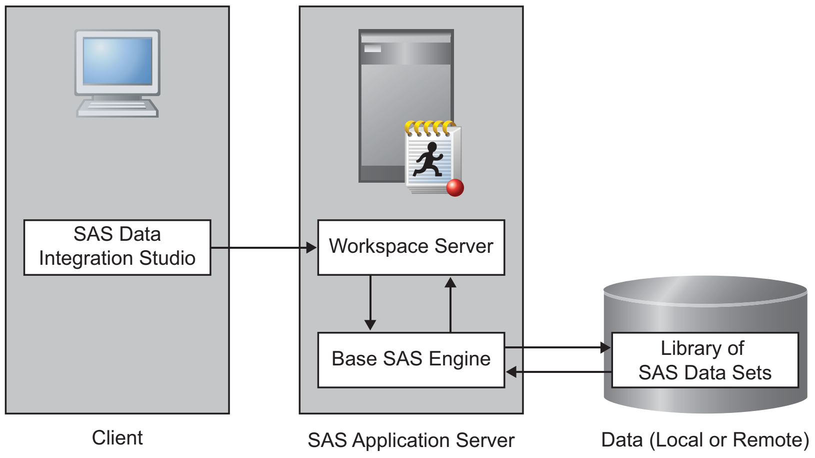
7 Examine the final page of the wizard to ensure that the proper values have been entered. Click **Finish** to save the library settings. At this point, you can register tables, as explained in "Registering and Verifying Tables" on page 54.

Establishing Connectivity to a Library of SAS Data Sets

Register the Base SAS Library

The following figure provides a logical view of accessing a library of SAS data sets.

Figure 2.1 Establishing Shared Access to SAS Data Sets



After you have installed the required SAS software, you need to set up a connection from a SAS server to a SAS data set. This connection requires that you register the Base SAS library with the SAS Metadata Server. In addition, you must import any user-defined formats that have been created for the data set in order to view or operate on the data. Assume that the SAS software has already been loaded by using the standard installation wizard and that the data set is stored in a location that can be accessed.

Register the library by using SAS Management Console. This metadata enables your SAS applications to access the data sets that you need to work with. For this example, the data set contains information about customers of the Orion Gold enterprise.

To register a Base SAS library, perform the following steps:

1 In SAS Management Console, expand **`Data Library Manager`**. Right-click **`Libraries`**. Then, select the **`New Library`** option to access the first page of the New Library wizard.

2 Select **`SAS BASE Library`** from the **`SAS Data`** list. Click **`Next`**.

3 Enter an appropriate library name in the **`Name`** field (for example, **`Orion Gold Customers`**). Note that you can supply an optional description if you want. Click **`Next`**.

4 Enter the following library properties:

Table 2.1 Library Properties

Field	**Sample Value**
`Libref`	**`ORGOLD`**
`Engine`	**`BASE`**
`Path Specification`	**`C:\SAS\Config\Lev1\SASApp\Data`** (Enter the fully qualified path to the library. This path is specified differently in different operating systems. Make sure that the appropriate path is displayed in the **`Selected items`** field.)

You can also click **`Advanced Options`** to perform tasks such as pre-assignment and setting host-specific and LIBNAME options. Click **`Next`** to access the next page of the wizard.

5 Select one or more SAS servers. The library is available to the server or servers that you select from this list and visible to users of the server. Click **`Next`**.

6 Examine the final page of the wizard to ensure that the proper values have been entered. Click **`Finish`** to save the settings.

At this point, you can register tables, as explained in "Registering and Verifying Tables" on page 54. If registering tables fails, check that the user account has host access permission to the path.

Working with User-Defined Formats

If you have existing SAS data sets, you might also have a catalog of user-defined formats and informats. You have two options for making these formats available to applications such as SAS Data Integration Studio and SAS Information Map Studio:

- □ Give the format catalog a standard name and place it in an expected location. This is the preferred method.
- □ Create a user-defined formats configuration file, and use the FMTSEARCH system option to point to the format catalog.

Use a Standard Name and Location for the Format Catalog

To make the format catalog available, in the preferred method, perform the following steps:

1 Name the format catalog formats.sas7bcat.

2 Place the catalog in the directory *SAS-config-dir*`\Lev1\SASApp\SASEnvironment\SASFormats`.

Create a User-Defined Formats Configuration File

Alternatively, you can create a user-defined formats configuration file in which you point to the location of the formats catalog.

To make a format catalog available using the alternative method on Windows and UNIX systems, perform the following steps:

1 To the SAS configuration file *SAS-config-dir*`\Lev1\SASApp\sasv9_usermods.cfg`, add the CONFIG system option, and use it to point to the user-defined formats configuration file.

```
-config "SAS-config-dir\Lev1\SASApp\userfmt.cfg"
```

2 Then, use the FMTSEARCH system option in the same configuration file to point to the format catalog:

```
-set fmtlib1 "SAS-config-dir\Lev1\Data\orformat"
-insert fmtsearch (fmtlib1.orionfmt)
```

In this example, *SAS-config-dir*`\Lev1\Data\orformat` is the location of the format catalog, and orionfmt (filename orionfmt.sas7bcat) is the name of the format catalog. If you have more than one catalog to list, leave a space between each catalog name.

Note: On UNIX systems, you must enter the variable name in uppercase. For example, you enter **`FMTLIB1`** instead of **`fmtlib1`**. △

To make a format catalog available using the alternative method on z/OS systems, perform the following steps:

1 Add the AUTOEXEC system option to the SAS launch command as shown in the following example.

```
SAS-config-dir/Lev1/SASApp/startsas.sh
   o("autoexec="./WorkspaceServer/userfmt.sas"")
```

In this example, **startsas.sh** is your SAS launch command script, and **userfmt.sas** is the name of the SAS autoexec file. When you enter the command, you must enter it all on one line.

2 In the autoexec file, use the LIBNAME statement to assign the format library and the OPTIONS statement to set the FMTSEARCH system option. For example, you might specify the following statements:

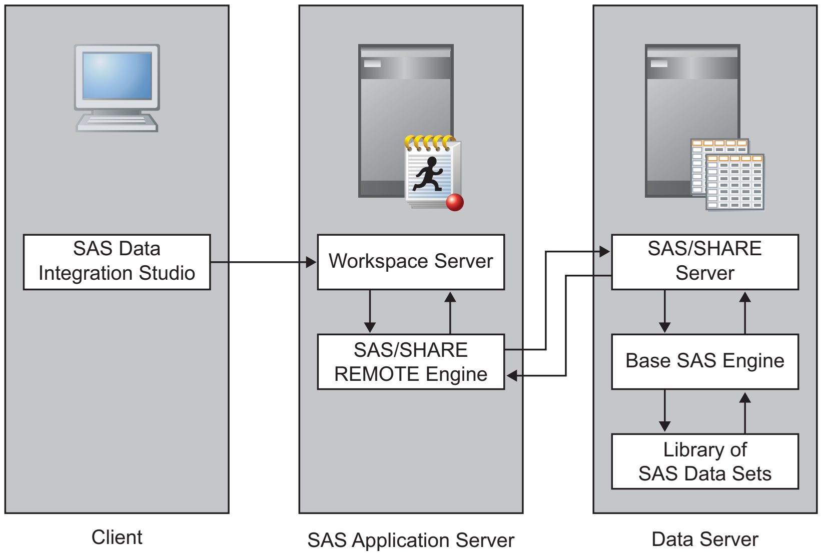
```
LIBNAME fmtlib1 'SAS-config-dir/Lev1/Data/orformat' repname=Foundation;
options fmtsearch=(fmtlib1.orionfmt);
```

Establishing Shared Access to SAS Data Sets

Overview of Establishing Shared Access

The following figure provides a logical view of accessing SAS data sets through a SAS/SHARE server.

Figure 2.2 Establishing Shared Access to SAS Data Sets



Base SAS libraries allow the following access:

□ Any number of users can read data.

□ A single user can write or update data.

This access can be extended through the use of the SAS/SHARE server. A SAS/SHARE server permits multiple users to update the same items in a SAS library.

You can share access to a library of existing SAS data sets by using a SAS/SHARE server to manage access to the data. Assume that the SAS/SHARE software has already been loaded by using the standard installation wizard, and that you have a SAS/SHARE server registered in metadata (for example, SHAREServer) that was created by the wizard. Configuring shared access is a two-stage process:

1. Create a SAS/SHARE REMOTE Engine Library. This library is assigned to a SAS application server, as shown in the previous figure.
2. While creating the SAS/SHARE REMOTE Engine Library, choose the option to register a new library to the SAS/SHARE server. This is shown in the previous figure as a Base SAS library. It is very important to pre-assign this library and to assign it to the SAS/SHARE server.

Create a SAS/SHARE Remote Engine Library

To create a SAS/SHARE Remote Engine library, perform the following steps:

1. In SAS Management Console, expand **`Data Library Manager`**. Right-click **`Libraries`**. Then, select the **`New Library`** option to access the New Library wizard.
2. Select **`SAS/SHARE REMOTE Engine Library`** from the **`SAS Data`** list. Click **`Next`**.
3. Enter an appropriate library name in the **`Name`** field (for example, **`SharedAccessToOrionGold`**). You can supply an optional description. Click **`Next`**.
4. Select one or more SAS servers (not a SAS/SHARE server at this point) and click the right arrow. The library is available to the servers included in this list and visible to users of the server. Click **`Next`**.
5. Enter a value for **`Libref`** and click **`Next`**.
6. Enter the following library properties:

   Table 2.2 Server and Connection Information

Field	Sample Value
`SAS/SHARE Server`	**`SHAREServer`**
`SAS/SHARE Server Library`	Click **`New`** to register a new library such as a Base SAS library. Assign the new library to the SAS/SHARE Server and set the library as pre-assigned.
`Default Login`	**`(None)`** (This default login is used to resolve conflicts between multiple logins to an authentication domain. In such cases, the default login is used.)

   Click **`Next`**.
7. Examine the final page of the wizard (for the SAS/SHARE REMOTE Engine Library) to ensure that the proper values have been entered. Click **`Finish`** to save the settings.
8. Restart the SAS/SHARE server.

At this point, you can register tables, as explained in “Registering and Verifying Tables” on page 54.

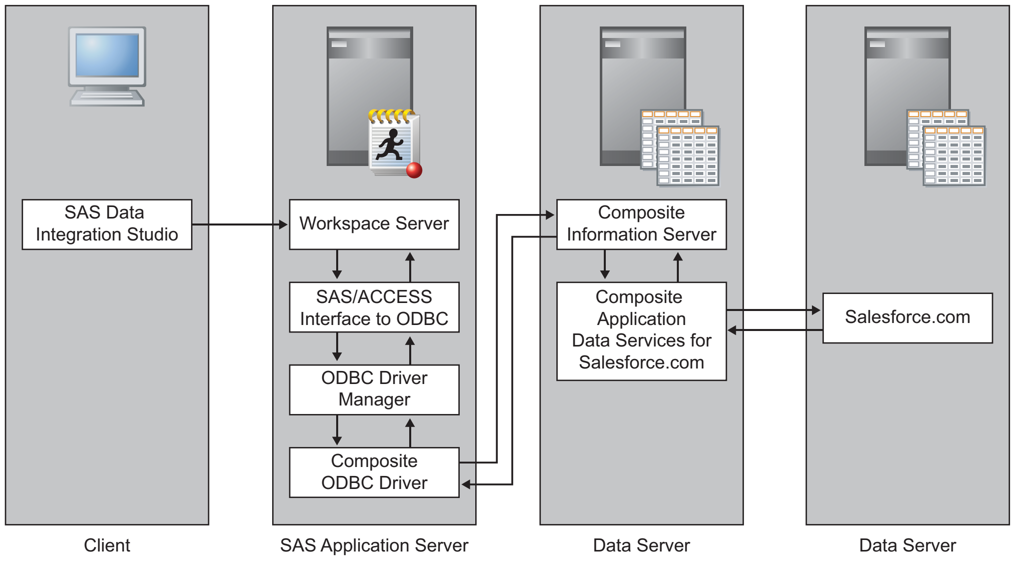
Establishing Connectivity to a Composite Information Server

Overview of Establishing Connectivity to a Composite Information Server

SAS has partnered with Composite Software to provide customers with access to enterprise resource planning (ERP) and customer relationship management (CRM) data. Composite Software provides the Composite Information Server and Application Data Services that tailor the Composite Information Server's access to the ERP or CRM data source. For this detailed example, the Composite Application Data Service to Salesforce.com is used. Detailed information about the data services is available from Composite Software.

The following figure provides a logical view of how SAS accesses CRM data from Salesforce.com through a SAS/ACCESS ODBC interface to a Composite Information Server.

Figure 2.3 Establishing Connectivity to a Composite Information Server


SAS Data Integration Studio
Workspace Server
SAS/ACCESS Interface to ODBC
ODBC Driver Manager
Composite ODBC Driver
Composite Information Server
Composite Application Data Services for Salesforce.com
Salesforce.com
Client
SAS Application Server
Data Server
Data Server

SAS supports setting up a connection to a Composite Information Server with ODBC. Configuring this connection is a four-stage process:

1 Configure the Composite Information Server to communicate with the data source. (This is described in the Composite Software documentation, but the high-level steps are covered here to show the relationships between data sources and user accounts.)

2 Define an ODBC data source.

3 Register the database server.

4 Register the database library.

Prerequisites

This example assumes that the following configuration tasks are complete before beginning the configuration of SAS software:

1 configuration of a user account and password for the data source, Salesforce.com, that will be used by the Composite Application Data Service for communicating with the data source.

2 installation of a Composite Information Server and the Composite Application Data Services for Salesforce.com.

3 installation of SAS/ACCESS Interface to ODBC. For requirements information, go to the Install Center at **`http://support.sas.com/documentation/installcenter/92/documents/index.html`** and use the operating system and SAS version to locate the appropriate SAS Foundation Configuration Guide.

4 (UNIX only) configuration of SAS/ACCESS environmental variables. For more information, see “Setting UNIX Environment Variables for SAS/ACCESS” on page 56.

Stage 1: Configuring the Composite Information Server

The following tasks are documented in detail in the *Composite Information Server Administration Guide*. The high-level steps are presented here to provide a complete walk through. At the completion of this stage the Composite Information Server can access the data at Salesforce.com.

To add Salesforce.com as a data source to Composite, perform the following steps:

1 Use Composite Studio to add a new data source to the **`Shared`** folder. Choose Salesforce.com as the data source driver. Set option **`Pass-Through Login`** to Enabled. The values in the user name and password fields are used to test and confirm connectivity. Deselect the **`Save Password`** check box. This step adds the physical data source.

Table 2.3 Data Source Wizard Properties

Field	**Sample Value**
`Data Source Driver`	**`Salesforce.com`**
`Datasource Name`	**`Salesforce.com`**
`Username`	Salesforce.com user name
`Password`	account password. If the connection fails, follow the instruction to reset the security token at Salesforce.com and append the security token value to the account password.
`Save Password`	Deselected
`Pass-Through Login`	Enabled

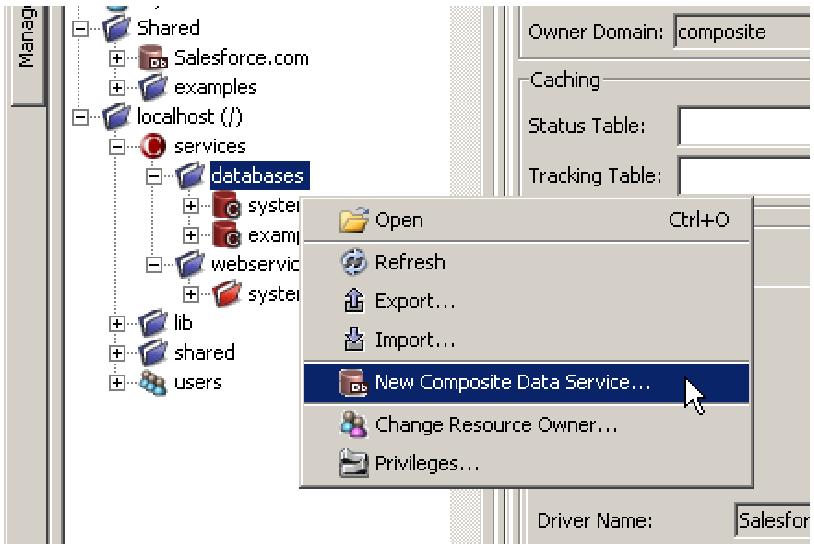
2 If the security token was used, then restart the Composite Information Server before continuing.

3 Right-click the Salesforce.com data source and select **`Open`**.

4 Click the **`Re-Introspection`** tab at the bottom of the right side pane. Either schedule Re-introspection on this pane, or periodically navigate to this pane and

click the **Re-Introspect Now** button. Re-introspection is necessary when tables or columns are added, removed, or altered.

5 Use Composite Studio to add a new data service that uses the data source. Right-click *host name*/**services/databases** and select **New Composite Data Service**.

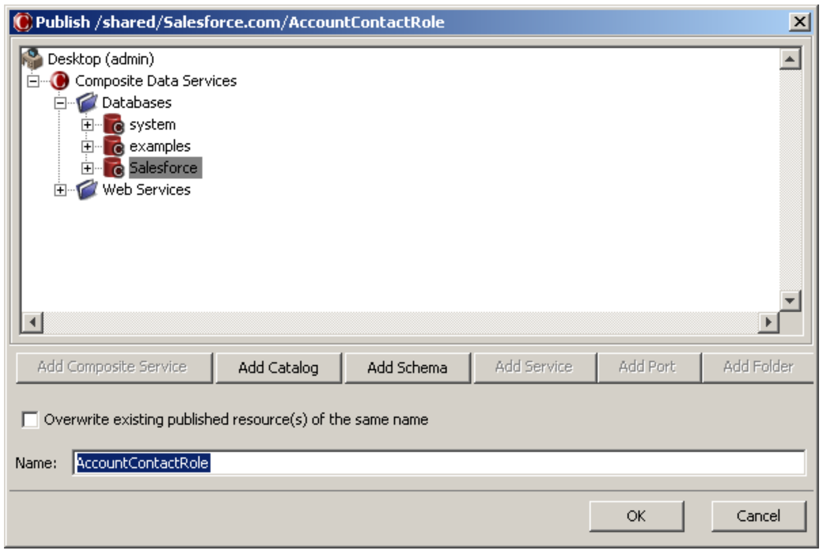


Enter the following configuration settings on the Add Composite Data Service dialog box:

Table 2.4 Add Composite Data Service Wizard Properties

Field	Sample Value
Data Service Name	**Salesforce**
Data Service Type	**Composite Database**

6 Right-click each table and procedure from the Salesforce.com data source and select **Publish**. On the Publish window, be sure to select the Salesforce data service and to remove spaces from the table name. For example, change “Account Contact Role” to “AccountContactRole.”

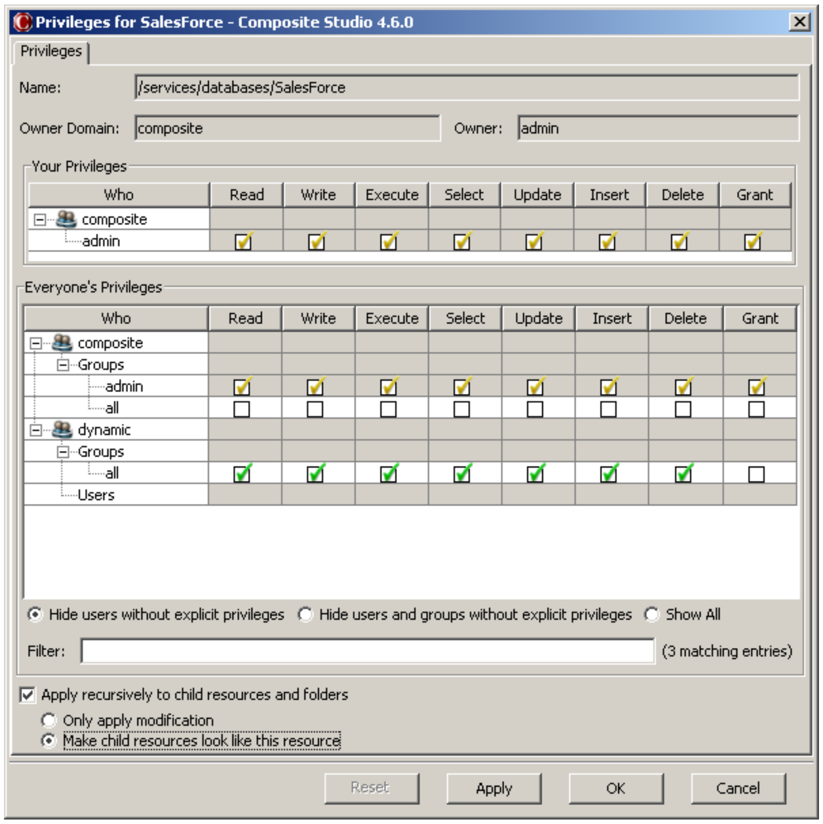


☐ Overwrite existing published resource(s) of the same name

Name: AccountContactRole

7 Enable the dynamic domain. For more information, see “Enabling the Dynamic Domain” in the *Composite Information Server Administration Guide*.

8 Use Composite Studio to set privileges on the data service (shown in the following figure) and the data source (not shown, but similar). Right-click *host name*/ **`services/databases/Salesforce`** and set permissions for **dynamic** ▶ **Groups** ▶ **all**. You must set permissions for Read and Select to make the data available to users. You might choose to set additional permissions for your site. Afterward, right-click **`Shared/Salesforce.com`** and set the same permissions.



Name: /services/databases/SalesForce

Owner Domain: composite Owner: admin

Your Privileges

Who	Read	Write	Execute	Select	Update	Insert	Delete	Grant
composite								
admin	✓	✓	✓	✓	✓	✓	✓	✓

Everyone's Privileges

Who	Read	Write	Execute	Select	Update	Insert	Delete	Grant
composite								
Groups								
admin	✓	✓	✓	✓	✓	✓	✓	✓
all	☐	☐	☐	☐	☐	☐	☐	☐
dynamic								
Groups								
all	✓	✓	✓	✓	✓	✓	✓	☐
Users								

[✓ Hide users without explicit privileges] Hide users and groups without explicit privileges Show All

Filter: (3 matching entries)

☑ Apply recursively to child resources and folders

Only apply modification [✓ Make child resources look like this resource]

Note: Ensure the **Apply changes recursively to child resources and folders** check box is selected. △

Stage 2: Configuring the Composite ODBC Driver

After the Composite Information Server is configured to transfer data with the data source, Salesforce.com, then the Composite ODBC Driver must be configured. This driver is configured on the SAS server host machine that is used to transfer data with the Composite Information Server. The driver is used by SAS to open a connection to the Composite Information Server, pass credentials for the data source to Composite, and transfer data.

To configure the Composite ODBC driver, perform the following steps:

1 On the SAS server host machine, start an installation of Composite Information Server. When you choose the Composite software components to install, select only the **ODBC** check box.
2 Open the Windows Control Panel. Then, double-click **Administrative Tools**. Then, double-click **Data Sources (ODBC)** to access the ODBC Data Source Administrator dialog box.
3 Click the **System DSN** tab, and then click **Add** to access the Create New Data Source dialog box.
4 Select **Composite** from the list, and click **Finish** to access the Composite Software ODBC Driver Configuration dialog box.
5 Enter the following configuration settings:

Table 2.5 Configuration Settings

Field	Sample Value
DSN Name	**SalesforceDSN**
Composite Host	Enter the host name for the Composite Information Server
Port	Use the default value of 9401
User Name	**demo** (This is a dummy value. Setting the domain to **dynamic** enables passing each users' credentials to the data source, Salesforce.com, instead of using **demo**.)
Password	Leave this field blank
Domain	**dynamic**
Datasource	**Salesforce** (Enter the name of the Data Service that was published during the first stage.)
Catalog	Leave this field blank

6 Click **OK** to save the configuration settings and return to the ODBC Data Source Administrator dialog box. Then, click **OK** to save the data source.

Stage 3: Register the ODBC Database Server

To register an ODBC database server, perform the following steps:

1 Open the SAS Management Console application.

2 Right-click **`Server Manager`** and select the **`New Server`** option to access the New Server wizard.

3 Select **`ODBC Server`** from the **`Database Servers`** list. Click **`Next`**.

4 Enter an appropriate server name in the **`Name`** field (for example, **`Composite Server`**). You can supply an optional description. One server is required for each DSN. Click **`Next`**.

5 Enter the following server properties:

Table 2.6 Server Properties

Field	Sample Value
`Major Version Number`	**`3`**
`Minor Version Number`	**`7`**
`Data Source Type`	**`ODBC - Other Database`**
`Software Version`	**`3.70`**
`Vendor`	**`Data Direct`**
`Associated Machine`	Select the Composite Information Server host machine value from the drop-down list. If the value that you need is not available, click **`New`** to access the New Machine dialog box. Then enter the appropriate value in the **`Host Name`** field.

Click **`Next`**.

6 Enter the following connection properties:

Table 2.7 Connection Properties

Field	Sample Value
`Datasrc`	**`SalesforceDSN`** (Use the value entered in the **`DSN Name`** field in the Composite Software ODBC Driver Configuration dialog box.)
`Authentication type`	**`User/Password`**
`Authentication Domain`	**`CompositeAuth`** (You might need to create a new authentication domain. For more information, see “How to Store Passwords for a Third-Party Server” in the *SAS Intelligence Platform: Security Administration Guide*.) Click **`New`** to access the New Authentication Domain dialog box. Then enter the appropriate value in the **`Name`** field and click **`OK`** to save the setting.

Click **`Next`**.

7 Examine the final page of the wizard to ensure that the proper values have been entered. Click **`Finish`** to save the wizard settings.

Stage 4: Register the ODBC Database Library

Important: Before tables can be registered, you must use the User Manager plug-in to SAS Management Console and edit each user that will access Salesforce.com. On the

Accounts tab, add a new account. For the account, set the User ID to be the Salesforce.com user name, set password to be the Salesforce.com security token, and set authentication domain to CompositeAuth.

After you have registered the database server, you register the database library. To register the database library, perform the following steps:

1 In SAS Management Console, expand the **Data Library Manager** node. Right-click **Libraries** and select the **New Library** option to access the New Library wizard.
2 Select **ODBC Library** from the **Database Data** list. Click **Next**.
3 Enter an appropriate library name in the **Name** field. For example, **Salesforce**. You can supply an optional description. Click **Next**.
4 Select the SAS server from the list that was configured with the Composite Software ODBC Driver and use the right arrow to assign the library to the SAS server. Click **Next**.
5 Enter the following library properties:

Table 2.8 Library Properties

Field	Sample Value
Libref	**sfref**
Engine	**ODBC**

6 *Important:* Click **Advanced Options**. On the Advanced Options dialog box, click the **Input/Output** tab and set **Preserve DBMS table names** to YES. Click **OK**.
7 Enter the following settings:

Table 2.9 Server and Connection Information

Field	Sample Value
Database Server	**Composite Server** (Use the database server that you selected in the New Server wizard.)
Database Schema Name	This field is not used.
Connection	Use the default value of Connection: <server_name>.
Default Login	Use the default value of (None).

Click **Next**.

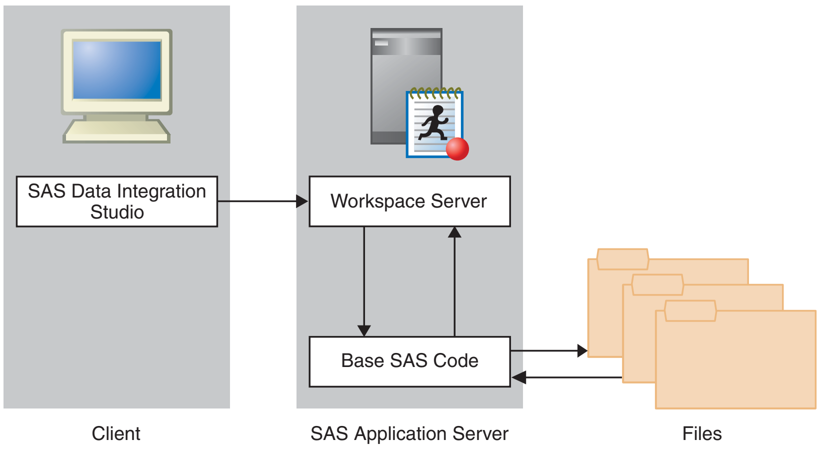
8 Examine the final page of the wizard to ensure that the proper values have been entered. Click **Finish** to save the library settings. At this point, you can register tables, as explained in “Registering and Verifying Tables” on page 54.

Establishing Connectivity to an Excel File

Overview of Establishing Connectivity to an Excel File

The following figure provides a logical view of using an Excel file as a data source.

Figure 2.4 Establishing Connectivity to an Excel File



The Excel file must be stored in a location that can be accessed by a Windows 32-bit machine. This example focuses on a file that is local to a SAS server, but a Windows UNC path such as `\\datasrv\sales\firstquarter.xls` is also acceptable.

To establish connectivity to an Excel file, perform the following steps:

1. In SAS Management Console, expand **`Data Library Manager`**. Right-click **`Libraries`**. Then, select the **`New Library`** option to access the New Library wizard.
2. Select **`Microsoft Excel Library`** from the Database Data list. Click **`Next`**.
3. Enter a value for **`Libref`** and click **`Next`**.
4. Enter an appropriate library name in the **`Name`** field (for example, FirstQuarterSales). Specify a metadata folder location for the library in the **`Library`** field. You can supply an optional description. Click **`Next`**.
5. Select one or more SAS servers and click the right arrow. The library is available to the servers included in this list and visible to users of the server. Click **`Next`**.
6. On the server and connection page, click **`New`** in the **`Server Details`** group box to specify a new database server.

   The New Server wizard displays.
7. Enter a name such as FirstQuarterSalesFolder in the **`Name`** field. Click **`Next`**.
8. Click **`Next`** on the Server Properties page.
9. On the Connection Properties page, enter the path to the Excel file. Enclose the value in quotation marks. For example, "**`C:\sales\firstquarter.xls`**." Click **`Next`**.
10. Examine the final page of the New Server wizard to ensure that the proper values have been entered. Click **`Finish`** to save the server settings.

    The New Server wizard closes and returns you to the final page of the New Library wizard.
11. Examine the final page of the New Library wizard to ensure that the proper values have been entered. Click **`Finish`** to save the library settings.

    At this point, you can register tables, as explained in “Registering and Verifying Tables” on page 54.

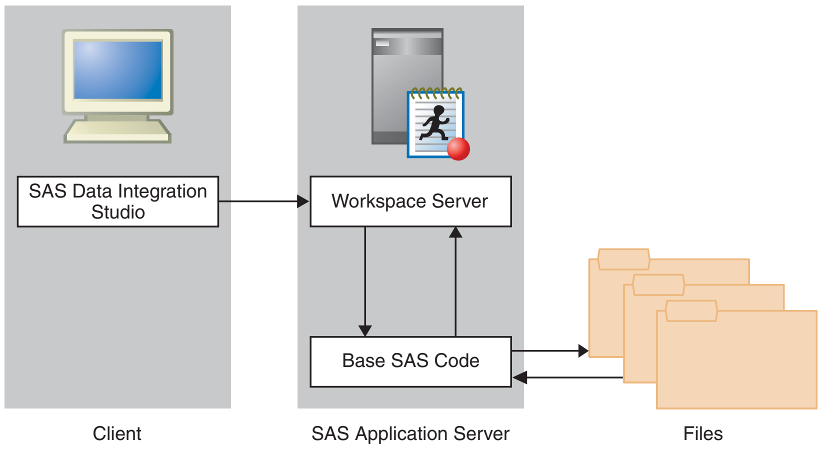
The Microsoft Excel Library wizard in SAS Management Console generates the metadata to construct a LIBNAME statement for the EXCEL LIBNAME engine. For more information about the EXCEL LIBNAME engine and supported options, see the *SAS/ACCESS Interface to PC Files: Reference*.

Establishing Connectivity to a Flat File

Overview of Establishing Connectivity to a Flat File

The following figure provides a logical view of using an external file as a data source.

Figure 2.5 Establishing Connectivity to External Files



You can connect to a flat file using the External File Source Designer in SAS Data Integration Studio.

Assume that the SAS software has already been loaded by using the standard installation wizard, and that the flat file is stored in a location that can be accessed. This example focuses on a comma-delimited flat file. A similar process is used for other types of flat files, but some steps are different.

To establish a connection to a flat file, perform the following steps:

1 Open SAS Data Integration Studio. Then, select **File** ▶ **New** ▶ **External File** ▶ **Delimited** to access the New Delimited External File wizard.

2 Enter a name for the external file and click **`Next`**.

3 Enter the fully qualified path to the file in the **`File name`** field (for example, *SAS-config-dir*`\sources\customer_data.dat`). Click **`Next`**.

4 On the Delimiters and Parameters page of the wizard, deselect the **`Blank`** option in the **`Delimiters`** group box. Then, select the **`Comma`** option. Click **`Next`** to access the Column Definitions page of the wizard.

5 To define the columns, perform the following steps:

   a Click **`Refresh`** to view the data from the flat file in the **`File`** tab in the view pane at the bottom of the page.

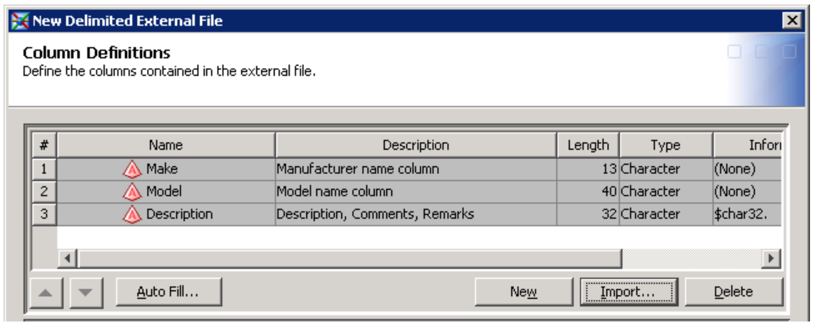
   b Click **`Auto Fill`** to access the Auto Fill Columns dialog box. Change the value entered in the **`Start record`** field in the **`Guessing records`** group box to **`2`**. This setting is based on the assumption that the first data record of the flat file contains header information and that the record is unique because it holds the column names for the file. Therefore, excluding the first data record from the guessing process yields more accurate preliminary data because it is excluded when the guessing algorithm is run.

6 Click **Import** to access the Import Column Definitions dialog box. The following four methods are provided for importing column definitions:
   - ☐ Get the column definitions from other existing tables or external files.
   - ☐ Get the column definitions from a format file.
   - ☐ Get column definitions from a COBOL format file.
   - ☐ Get the column names from column headings in the file.

   In most cases, you will either get the column definitions from an external format file or get the column names from the column headings in the external file. Here is an example of a format file:

```
# Header follows
Name,SASColumnType,SASColumnName,SASColumnLength,
SASInformat,SASFormat,Desc,ReadFlag
# Column definition records records follow
Make,C,Make,13,,$char13.,Manufacturer name column,y
Model,C,Model,40,,$char40.,Model name column,y
# Comma within quotation marks below is not a delimiter
Description,C,Description,32,$char32.,,'Description, Comments, Remarks',y
```

   A sample of the output is shown in the following figure:


New Delimited External File
Column Definitions
Define the columns contained in the external file.

#	Name	Description	Length	Type	Infor
1	Make	Manufacturer name column	13	Character	(None)
2	Model	Model name column	40	Character	(None)
3	Description	Description, Comments, Remarks	32	Character	$char32.

Auto Fill...
New
Import...
Delete

   For this example, select the **Get the column names from column headings in the file** radio button. Keep the default settings for the fields underneath it.

   Note: If you select **Get the column names from column headings in the file**, the value in the **Starting** record field in the **Data** tab of the view pane in the Column Definitions dialog box 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 dialog box. △

7 Click **OK** to return to the Column Definitions page.

8 The preliminary data for the external file object is displayed in the columns table at the top of the page. The **Informat** and **Format** columns for the rows in the table are based on the values that are included in the sample data that is processed by the guessing function. The results are accurate for this particular set of records, but you should still examine them to make sure that they are representative of the data in the rest of the flat file. Edit the values by clicking directly on the cells in the column table and making the necessary changes.

9 Click the **Data** tab at the bottom of the Column Definitions page. Then, click **Refresh**. The data should be properly formatted. If not, edit the cells in the column table and check the results by refreshing the **Data** tab. You can repeat this process until you are satisfied. You can review the SAS log for more details.

   Note: To view the code that will be generated for the external file, click the **Source** tab. To view the SAS log for the generated code, click the **Log** tab. The code that is displayed in the **Source** tab is the code that will be generated for the current external file. △

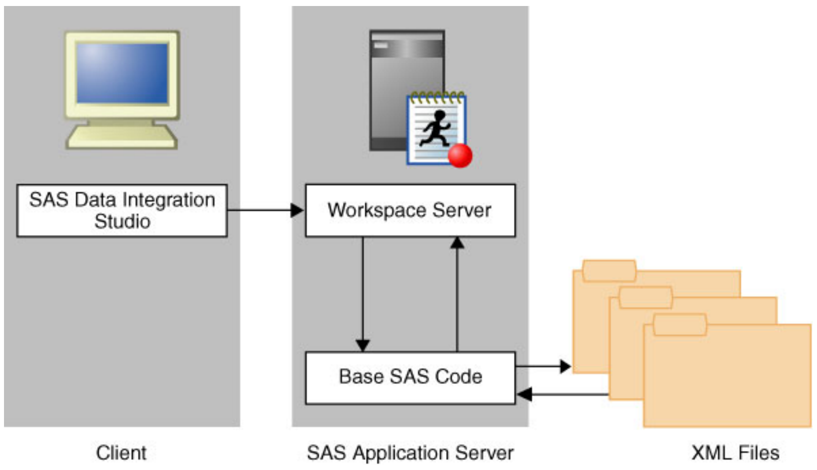
10 Click **Next**.

11 Examine the final page of the wizard to ensure that the proper values have been entered. Click **Finish** to save the library settings. The file is ready for use.

Establishing Connectivity to XML Data

The following figure provides a logical view of using XML files as a data source.

Figure 2.6 Establishing Connectivity to XML Files



The following steps describe how to specify a SAS XML library in SAS Management Console. Assume that the XML library will point to an XML file that contains climate information (**climate.xml**). The XML file is in generic format, as defined for the SAS XML LIBNAME engine. For more information, see the *SAS XML LIBNAME Engine: User's Guide*.

To register an XML library, perform the following steps:

1 In SAS Management Console, expand **Data Library Manager**. Right-click **Libraries**. Then, select the **New Library** option to access the New Library wizard.

2 Select **SAS XML library** from the **SAS Data** list. Click **Next**.

3 Enter an appropriate library name in the **Name** field (for example, **XML Lib**). Click **Next**.

4 Enter information about the library, such as the following:

Table 2.10 Library Properties

Field	Sample Value
Name	**XML Lib**
Libref	**xmllib**
Engine	**XML**

Field	Sample Value
`XML File`	`C:\sources\xml\climate.xml`
`XML Type`	`GENERIC`
`Library Access`	`READONLY`

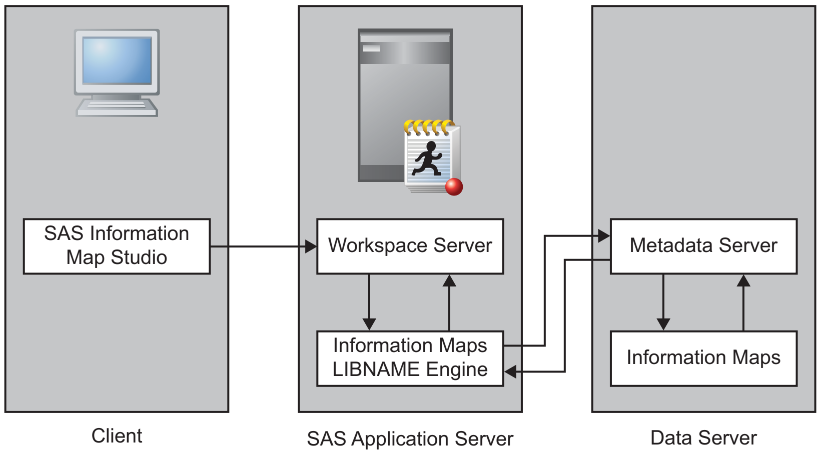
5 Click **`Finish`** to save the wizard settings.

Establishing Connectivity to a SAS Information Map

Overview of Establishing Connectivity to a SAS Information Map

A SAS Information Map is a business metadata layer on top of another data source. When an information map is registered as a table in a SAS Information Map library, it can also be used as the data source for other information maps. For more information about creating information maps, see either the *SAS Information Map Studio Help* or *Base SAS Guide to Information Maps*. Information map tables are expected to be used primarily by SAS Information Map Studio and SAS Enterprise Guide.

Figure 2.7 Establishing Connectivity to an Information Map



To register a SAS Information Map library, perform the following steps:

1 In SAS Management Console, expand **`Data Library Manager`**. Right-click **`Libraries`**. Then, select the **`New Library`** option to access the New Library wizard.

2 Select **`SAS Information Map Library`** from the **`SAS Data`** list. Click **`Next`**.

3 Enter an appropriate library name in the **`Name`** field (for example, **`InfoMapLib`**). Click **`Next`**.

4 Select a SAS server from the list and use the right arrow to assign the SAS server. This step makes the library available to the server and makes the library visible to users of the server. Click **`Next`**.

5 Enter information about the library, such as the following:

Table 2.11 Library Properties

Field	Sample Value
`Libref`	**`maplib`**
`Engine`	**`SASIOIME`**
`Metadata server`	select a metadata server from the list
`Default login`	**`(None)`**
`Information map location`	**`/Shared Data/SASInfoMaps`**

Click **`Next`**.

6 Examine the final page of the wizard to ensure that the proper values have been entered. Click **`Finish`** to save the library settings. At this point, you can register tables, as explained in “Registering and Verifying Tables” on page 54.

Special Considerations for Information Map Tables

- When registering the tables with SAS Management Console, ensure that the check box for the option **`Enable special characters within DBMS object names`** option is selected. The New Library wizard uses data item IDs from the source information map as column names for the table. These data item IDs sometimes contain special characters.
- The data item IDs of a source information map are used as column names when the information map is registered as an information map table. If you change the data item ID or table column name after the table is registered, then you will not be able to run queries on the table.
- By default, the values retrieved from an information map table are the detail values from the source information map. If you want to retrieve aggregated values, then you must set an aggregation option. You can set the AGGREGATE= options for tables and libraries in Data Library Manager in SAS Management Console. After you add a table as a data source for an information map, you can also set the aggregation option for the data source in the Table Properties dialog box in SAS Information Map Studio. Selecting the **`Use the detail values from the data source`** radio button is equivalent to setting the AGGREGATE= option to NO, and selecting the **`Use the aggregated values from the data source`** radio button is equivalent to setting the AGGREGATE= option to YES. This setting overrides the aggregation settings on the library or the table.
- When referenced by an information map table, measure data items that reference other measure data items or that use aggregate functions in their expressions can produce aggregated values only. If a source information map contains one of these data items, then set the AGGREGATE= option to YES on the library. Otherwise, these data items will not be registered as columns when you register the information map as a table.
- If the source information map for an information map table has measure data items that reference other measure data items or that use aggregate functions in their expressions, then the combination of setting the AGGREGATE= option to YES on the library and setting the AGGREGATE= option to NO on the table or on the data source (within SAS Information Map Studio) causes the table data to be inaccessible.

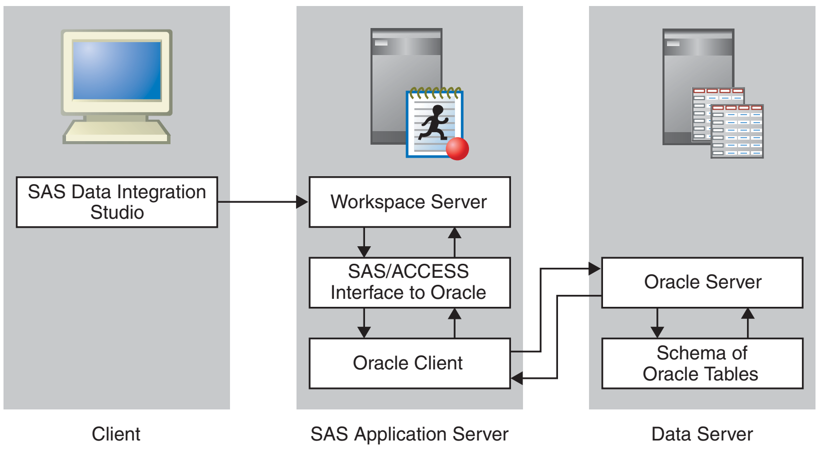
- □ Normally, when an information map table is registered, its columns get their formats from the associated source data items. But when a source data item is a measure data item and has a data type of character, then if the AGGREGATE= option is set to NO on the library, the format is not set to the format of the source data item. The format is set to the format of the column that the source data item is based on.
- □ Because an information map acts as a reference to underlying data, ReadMetadata permission must be granted to a user for the information map table, the source information map, and the table used by the source information map. Read permission is also needed on the source information map.
- □ Stored processes and prefilters associated with a source information map are applied to the information map table. (Filters that are not used as prefilters are not applied.)
- □ If a stored process is associated with a source information map and the stored process uses prompts with default values, the stored process is applied to the information map table. If the prompts do not have default values, the stored process does not affect the information map table.
- □ Information maps created from OLAP cubes cannot be registered as information map tables.
- □ The source information map name must be 32 bytes or less.

Establishing Connectivity to an Oracle Database

Overview of Establishing Connectivity to an Oracle Database

The following figure provides a logical view of using Oracle with SAS/ACCESS as a data source.

Figure 2.8 Establishing Connectivity to Oracle Databases



Setting up a connection from SAS to a database management system is a two-stage process:

1 Register the database server.

2 Register the database library.

This example shows the process for establishing a SAS connection to an Oracle database. It assumes that the software for the database has already been loaded by using the standard installation wizard for the database client. The following prerequisites have been satisfied:

- installation of SAS/ACCESS Interface to Oracle. For requirements information, go to the Install Center at **`http://support.sas.com/documentation/installcenter/92/documents/index.html`** and use the operating system and SAS version to locate the appropriate SAS Foundation Configuration Guide.
- installation of a supported Oracle Database Client.
- validation that the Oracle client can communicate with the Oracle server.
- (UNIX only) configuration of SAS/ACCESS environmental variables. For more information, see "Setting UNIX Environment Variables for SAS/ACCESS" on page 56.

Stage 1: Register the Database Server

To register the Oracle database server, perform the following steps:

1. Open the SAS Management Console application.
2. Right-click **`Server Manager`** and select the **`New Server`** option to access the New Server wizard.
3. Select **`Oracle Server`** from the **`Database Servers`** list. Then, click **`Next`**.
4. Enter an appropriate server name in the **`Name`** field (for example, **`Oracle Server`**). Note that you can supply an optional description if you want. Click **`Next`**.
5. Enter the following server properties:

Table 2.12 Server Properties

Field	Sample Value
`Major Version Number`	**`10`**
`Minor Version Number`	**`2`**
`Software Version`	**`10.2.0`**
`Vendor`	**`Oracle Corporation`**

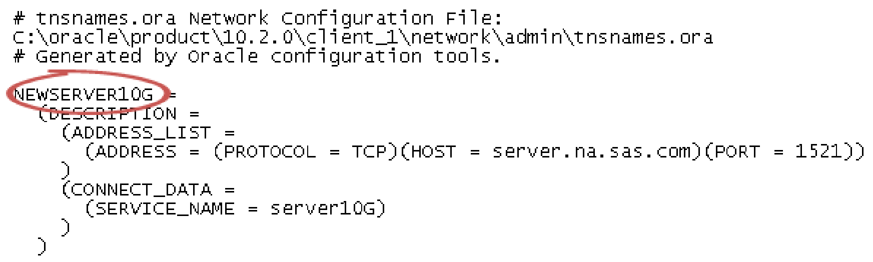
Click **`Next`**.

6. Enter the following connection properties:

Table 2.13 Connection Properties

Field	**Sample Value**
`Path`	**`NEWSERVER10G`** (This value is contained in the **`tnsnames.ora`** file generated during the Oracle installation. The file is stored in an Oracle installation directory such as **`/opt/oracle/app/oracle/product/10.2.0/db_1/network/admin/tnsnames.ora`**. The alias for the connection information is contained in this file. See the following figure.)
`Authentication type`	**`User/Password`**
`Authentication domain`	**`OracleAuth`** (You might need to create a new authentication domain. For more information, see “How to Store Passwords for a Third-Party Server” in the *SAS Intelligence Platform: Security Administration Guide*.) Click **`New`** to access the New Authentication Domain dialog box. Then enter the appropriate value in the **`Name`** field and click **`OK`** to save the setting.

The following figure shows a sample **`tnsnames.ora`** file:



```
# tnsnames.ora Network Configuration File:
C:\oracle\product\10.2.0\client_1\network\admin\tnsnames.ora
# Generated by Oracle configuration tools.

NEWSERVER10G =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = server.na.sas.com)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = server10G)
    )
  )
```

Note that the correct **`Path`** value is circled. Click **`Next`**.

7 Examine the final page of the wizard to ensure that the proper values have been entered. Click **`Finish`** to save the wizard settings.

Stage 2: Register the Database Library

After you have registered the database server, you can register the database library. To register the Oracle database library, perform the following steps:

1 In SAS Management Console, expand **`Data Library Manager`**. Right-click **`Libraries`**. Then, select the **`New Library`** option to access the New Library wizard.

2 Select **`Oracle Library`** from the **`Database Data`** list. Click **`Next`**.

3 Enter an appropriate library name in the **`Name`** field (for example, **`Oracle Library`**). You can supply an optional description. Click **`Next`**.

4 Select a SAS server from the list and use the right arrow to assign the SAS server. This step makes the library available to the server and makes the library visible to users of the server. Click **`Next`**.

5 Enter the following library properties:

Table 2.14 Library Properties

Field	Sample Value
`Libref`	**`ORAREF`**
`Engine`	**`ORACLE`**

You can also click **`Advanced Options`** to perform tasks such as pre-assignment and optimization. Click **`Next`** to access the next page of the wizard.

6 Enter the following settings:

Table 2.15 Server and Connection Information

Field	Sample Value
`Database Server`	**`OracleServer`** (Use the database server that you created in the New Server wizard.)
`Database Schema Name`	See your Database Administrator for the correct value.
`Default Login`	If an authentication domain is used, leave this set to **`None`**.

Click **`Next`**.

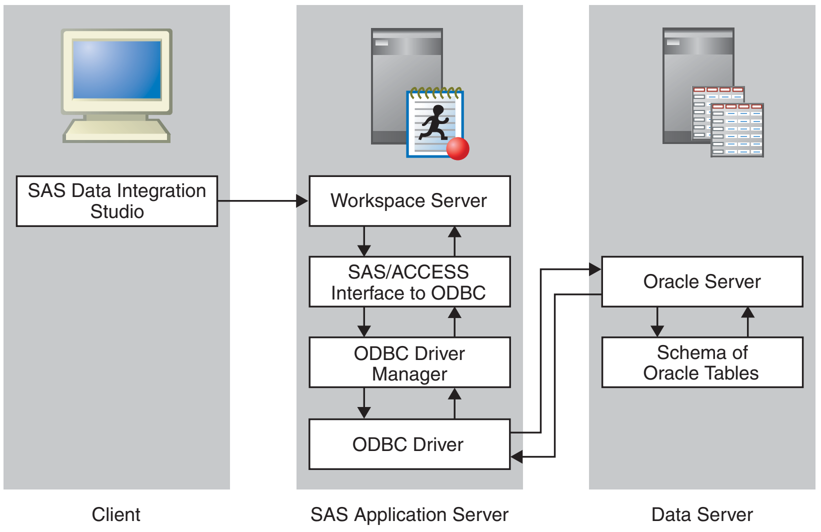
7 Examine the final page of the wizard to ensure that the proper values have been entered. Click **`Finish`** to save the library settings. At this point, you can register tables, as explained in “Registering and Verifying Tables” on page 54.

Establishing Connectivity to an Oracle Database by Using ODBC

Overview of Establishing Connectivity to an Oracle Database by Using ODBC

The following figure provides a logical view of using Oracle as a data source and connecting to the database with a SAS/ACCESS ODBC interface.

Figure 2.9 Establishing Connectivity to Oracle Databases By Using ODBC



Setting up a connection from SAS to an Oracle database management system by using ODBC is a three-stage process:

1 Define an ODBC data source.

2 Register the database server.

3 Register the database library.

This example shows the process for establishing a SAS connection to an Oracle database. It assumes that the software for the database has already been loaded with the standard installation wizard for the database client. Before you begin, satisfy the following prerequisites:

- installation of SAS/ACCESS Interface to ODBC. For requirements information, go to the Install Center at **`http://support.sas.com/documentation/installcenter/92/documents/index.html`** and use the operating system and SAS version to locate the appropriate SAS Foundation Configuration Guide.
- installation of a supported Oracle Database Client if your ODBC driver requires a client. Refer to the ODBC driver vendor's documentation to determine whether an Oracle client is required.
- validation that the Oracle client can communicate with the Oracle server.
- (UNIX only) configuration of SAS/ACCESS environmental variables. For information about setting environmental variables when you use SAS/ACCESS to

connect to data on UNIX systems, see “Setting UNIX Environment Variables for SAS/ACCESS” on page 56.

Stage 1: Define the ODBC Data Source

First, you must define the ODBC data source. To define the ODBC data source on Window systems, perform the following steps:

1 Open the Windows Control Panel. Then, double-click **`Administrative Tools`**. Double-click **`Data Sources (ODBC)`** to access the ODBC Data Source Administrator dialog box.

2 Click **`Add`** to access the Create New Data Source dialog box. Click the Oracle driver listed in the window (for example, **`Oracle in OraClient10g_home1`**). Click **`Finish`** to access the Oracle ODBC Driver Configuration dialog box.

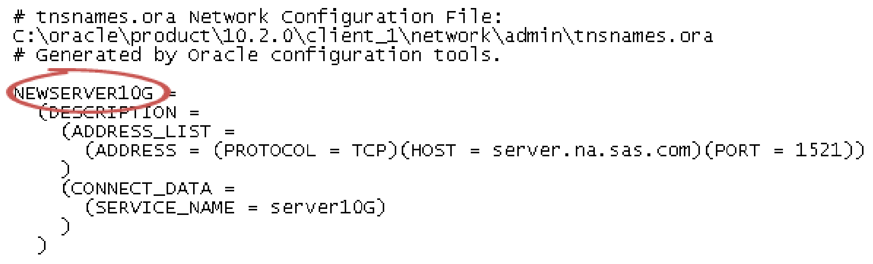
   Note: System data sources and user data sources store information about how to connect to the indicated data provider. A system data source is visible to all users with access to the system, including Windows services. A user data source is visible only to a particular user, and it can be used on the current machine only. For this example, we are creating a system data source. △

3 Enter the following configuration settings:

Table 2.16 Configuration Settings

Field	Sample Value
`Data Source Name`	**`Oracle_newserver`**
`TNS Service Name`	**`NEWSERVER10G`** (Select the name entered in the tnsnames.ora file created during installation of the Oracle database from the drop-down menu. See the following figure.)
`User`	**`User Name`**

The following display shows the **`tnsnames.ora`** file:



```
# tnsnames.ora Network Configuration File:
C:\oracle\product\10.2.0\client_1\network\admin\tnsnames.ora
# Generated by Oracle configuration tools.

NEWSERVER10G =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = server.na.sas.com)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = server10G)
    )
  )
```

4 Click **`OK`** to save the configuration settings and return to the ODBC Data Source Administrator dialog box. Then, click **`OK`** to save the data source.

Stage 2: Register the Database Server

To register the database server, perform the following steps:

1 Open the SAS Management Console application.

2 Right-click **`Server Manager`** and select the **`New Server`** option to access the New Server wizard.

3 Select **`ODBC Server`** from the **`Database Servers`** list. Click **`Next`**.

4 Enter an appropriate server name in the **`Name`** field (for example, **`ODBC Server`**). You can supply an optional description. One server is required for each DSN. Click **`Next`**.

5 Enter the following server properties:

Table 2.17 Server Properties

Field	Sample Value
`Major Version Number`	**`3`**
`Minor Version Number`	**`7`**
`Data Source Type`	**`ODBC - Oracle`**
`Software Version`	**`10`**
`Vendor`	**`Oracle`**
`Associated Machine`	**`newserver.na.sas.com`** This is the server where the database is running. (Select this value from the drop-down list. If the value that you need is not available, click **`New`** to access the New Machine dialog box. Then enter the appropriate value in the **`Host Name`** field.)

Click **`Next`**.

6 Enter the following connection properties:

Table 2.18 Connection Properties

Field	Sample Value
`Datasrc`	**`Oracle_newserver`** (Use the value entered in the **`Data Source Name`** field in the ODBC Data Source Administrator dialog box.)
`Authentication type`	**`User/Password`**
`Authentication Domain`	**`ODBCAuth`** (You might need to create a new authentication domain. For more information, see "How to Store Passwords for a Third-Party Server" in the *SAS Intelligence Platform: Security Administration Guide*.) Click **`New`** to access the New Authentication Domain dialog box. Then enter the appropriate value in the **`Name`** field and click **`OK`** to save the setting.

Click **`Next`**.

7 Examine the final page of the wizard to ensure that the proper values have been entered. Click **`Finish`** to save the wizard settings.

Stage 3: Register the Database Library

After you have registered the database server, you can register the database library. To register the database library, perform the following steps:

1 In SAS Management Console, expand **`Data Library Manager`**. Right-click **`Libraries`** and select the **`New Library`** option to access the New Library wizard.

2 Select **`ODBC Library`** from the **`Database Data`** list. Click **`Next`**.

3 Enter an appropriate library name in the **`Name`** field (for example, **`ODBC Library`**). Note that you can supply an optional description if you want. Click **`Next`**.

4 Select a SAS server from the list and use the right arrow to assign the SAS server. This step makes the library available to the server and makes the library visible to users of the server. Click **`Next`**.

5 Enter the following library properties:

Table 2.19 Library Properties

Field	Sample Value
`Libref`	**`ODBCREF`**
`Engine`	**`ODBC`**

You can also click **`Advanced Options`** to perform tasks such as pre-assignment and optimization. Click **`Next`** to access the next page of the wizard.

6 Enter the following settings:

Table 2.20 Server and Connection Information

Field	Sample Value
`Database Server`	**`ODBCServer`** (Use the database server that you selected in the New Server wizard.)
`Database Schema Name`	See your Database Administrator for the correct value.
`Connection`	Use the default value of Connection: *server_name*.
`Default Login`	Use the default value of (None).

Click **`Next`**.

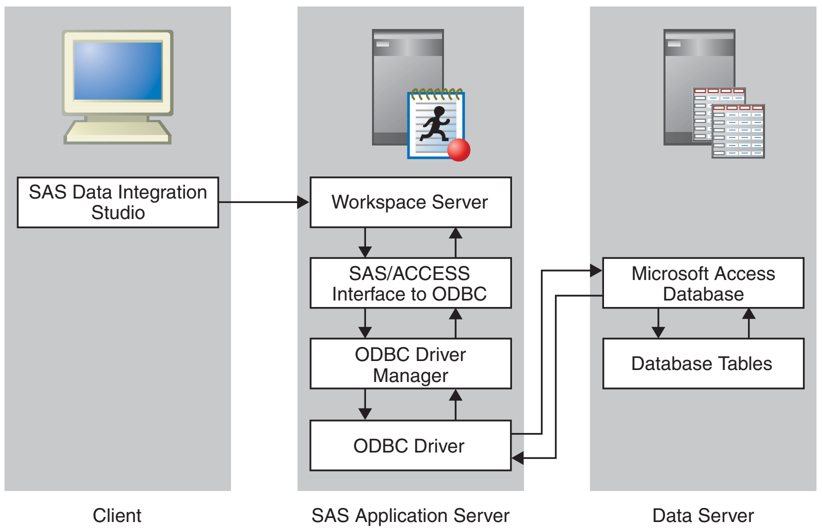
7 Examine the final page of the wizard to ensure that the proper values have been entered. Click **`Finish`** to save the library settings. At this point, you can register tables, as explained in “Registering and Verifying Tables” on page 54.

Establishing Connectivity to a Microsoft Access Database by Using ODBC

Overview of Establishing Connectivity to a Microsoft Access Database by Using ODBC

The following figure provides a logical view of using Microsoft Access as a data source and connecting to the database with a SAS/ACCESS ODBC interface.

Figure 2.10 Establishing Connectivity to Access Databases By Using ODBC



Setting up a connection from SAS to a Microsoft Access database by using ODBC is a three-stage process:

1 Define an ODBC data source.
2 Register the database server.
3 Register the database library.

This example shows the process for establishing a SAS connection to an Access database. It assumes that the software for the database has already been loaded with the standard installation wizard for the database client. In addition, SAS/ACCESS Interface to ODBC must be installed on the SAS server that will access the Access database.

Stage 1: Define the ODBC Data Source

First, you must define the ODBC data source. To define the ODBC data source on Windows systems, perform the following steps:

1 Open the Windows Control Panel. Then, double-click **`Administrative Tools`**. Finally, double-click **`Data Sources (ODBC)`** to access the ODBC Data Source Administrator dialog box.

2 Click **Add** to access the Create New Data Source dialog box. Click the Microsoft Access driver listed in the window (for example, **Microsoft Access Driver [*.mdb]**). Click **Finish** to access the ODBC Microsoft Access Setup dialog box.

Note: System data sources and user data sources store information about how to connect to the indicated data provider. A system data source is visible to all users with access to the system, including Windows services. A user data source is visible only to a particular user, and it can be used on the current machine only. △

3 Enter the following configuration settings:

Table 2.21 Configuration Settings

Field	Sample Value
Data Source Name	**MS Access**
Database	Click **Select** to browse for your Access database file, such as **Northwinds.mdb** in the Microsoft Office Samples directory.

4 Click **OK** to save the configuration settings and return to the ODBC Data Source Administrator dialog box. Then, click **OK** to save the data source.

Stage 2: Register the Database Server

To register the database server, perform the following steps:

1 Open the SAS Management Console application.

2 Right-click **Server Manager** and select the **New Server** option to access the New Server wizard.

3 Select **ODBC Server** from the **Database Servers** list. Click **Next**.

4 Enter an appropriate server name in the **Name** field (for example, **MS Access Server**). One server is required for each DSN. Note that you can supply an optional description if you want. Click **Next**.

5 Enter the following server properties:

Table 2.22 Server Properties

Field	Sample Value
Major Version Number	**3**
Minor Version Number	**7**
Data Source Type	**ODBC - Microsoft Access**
Software Version	**3.7.0**
Vendor	**Microsoft**
Associated Machine	**newserver.na.sas.com** This is the server where the database is running. (Select this value from the drop-down list. If the value that you need is not available, click **New** to access the New Machine dialog box. Then enter the appropriate value in the **Host Name** field.)

Click **Next**.

6 Enter the following connection properties:

Table 2.23 Connection Properties

Field	Sample Value
`Datasrc`	`MS Access` (Use the value entered in the `Data Source Name` field in the ODBC Data Source Administrator dialog box.)
`Authentication type`	`User/Password`
`Authentication Domain`	`(None)`

Click `Next`.

7 Examine the final page of the wizard to ensure that the proper values have been entered. Click `Finish` to save the wizard settings.

Stage 3: Register the Database Library

After you have registered the database server, you can register the database library. To register the database library, perform the following steps:

1 In SAS Management Console, expand `Data Library Manager`. Then, right-click `Libraries` and select the `New Library` option to access the New Library wizard.

2 Select `ODBC Library` from the `Database Data` list. Click `Next`.

3 Enter an appropriate library name in the `Name` field (for example, `MS Access Library`). Note that you can supply an optional description if you want. Click `Next`.

4 Select an application server from the list and use the right arrow to assign the application server. Click `Next`.

5 Enter the following library properties:

Table 2.24 Library Properties

Field	Sample Value
`Libref`	`ACCESREF`
`Engine`	`ODBC`

You can also click `Advanced Options` to perform tasks such as pre-assignment and optimization. Click `Next` to access the next page of the wizard.

6 Enter the following settings:

Table 2.25 Server and Connection Information

Field	Sample Value
`Database Server`	`MS Access Server` (Use the database server that you created in the New Server wizard.)

Click `Next`.

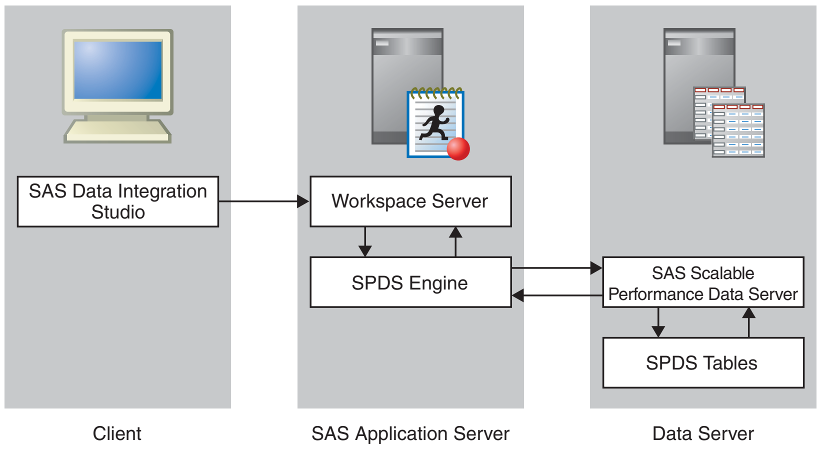
7 Examine the final page of the wizard to ensure that the proper values have been entered. Click `Finish` to save the library settings. At this point, you can register tables, as explained in "Registering and Verifying Tables" on page 54.

Establishing Connectivity to a Scalable Performance Data Server

Overview of Establishing Connectivity to a Scalable Performance Data Server

The following figure provides a logical view of using SPD Server tables as a data source.

Figure 2.11 Establishing Connectivity to an SPD Server



Configuring a connection from SAS to a Scalable Performance Data Server (SPD Server) is a three-stage process:

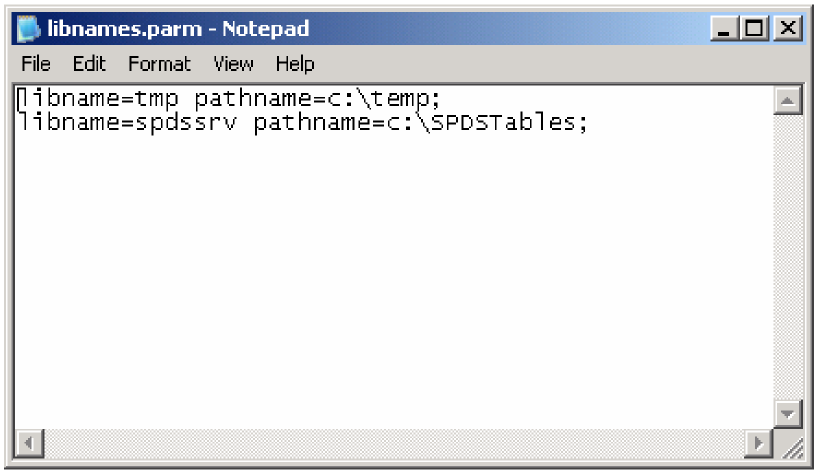
1. Configure the **`libnames.parm`** file.
2. Register the SPD server.
3. Register the SPD server library.

This example shows the process for establishing a SAS connection to SPD Server. It assumes that the software for the database has already been loaded by using the standard installation wizard for the database client. The SPD Server client and server software must be installed before the connection can be established.

Stage 1: Configure the libnames.parm File

When you install the SPD Server software on Windows, a **`libnames.parm`** file is created in the **`C:\Program Files\SAS Institute Inc\`***SPDS-version***`\Site`** directory. You must specify at least a LIBNAME and a pathname for the directory where the SPD Server tables will be saved (for example, **`C:\SPDSTables`**). For the LIBNAME, use the LIBNAME domain that you created earlier for the library (in this case, *spdsrv*).

A sample **`libnames.parm`** file is shown in the following figure:



```
libname=tmp pathname=c:\temp;
libname=spdssrv pathname=c:\SPDSTables;
```

Stage 2: Register the Server

To register the database server, perform the following steps:

1 Open the SAS Management Console application.

2 Right-click **`Server Manager`** and select the **`New Server`** option to access the New Server wizard.

3 Select **`SAS Scalable Performance Data Server`** from the **`SAS Servers`** list. Then, click **`Next`**.

4 Enter an appropriate server name in the **`Name`** field (for example, **`SPDServer`**). You can supply an optional description. Click **`Next`**.

5 Enter the following server properties:

Table 2.26 Server Properties

Field	Sample Value
`Major Version Number`	`4`
`Minor Version Number`	`3`
`Vendor`	`SAS Institute`
`SAS Compatibility`	`SAS 9`

Click **`Next`**.

6 Enter the following connection properties:

Table 2.27 Connection Properties

Field	Sample Value
`Host`	`D1234`
`Port Number or Name`	`5200` (Enter the port number for the SPD Server name server.)

Field	Sample Value
Communication Protocol	**TCP**
Authentication Domain	**SPDSAuth** (You might need to create a new authentication domain. For more information, see "How to Store Passwords for a Third-Party Server" in the *SAS Intelligence Platform: Security Administration Guide*.) Click **New** to access the New Authentication Domain dialog box. Then enter the appropriate value in the **Name** field and click **OK** to save the setting.

7 Examine the final page of the wizard to ensure that the proper values have been entered. Click **Finish** to save the wizard settings.

Stage 3: Register the Library

After you have registered the server, you can register the library. To register the library, perform the following steps:

1 In SAS Management Console, expand **Data Library Manager**. Right-click **Libraries**. Then, select the **New Library** option to access the New Library wizard.
2 Select **SAS Scalable Performance Data Server V4 Library** from the **SAS Data** list. Click **Next**.
3 Enter an appropriate library name in the **Name** field (for example, **SPDServerLibrary**). You can supply an optional description. Click **Next**.
4 Select a SAS server from the list and use the right arrow to assign the SAS server. This step makes the library available to the server and makes the library visible to users of the server. Click **Next**.
5 Enter the following library properties:

Table 2.28 Library Properties

Field	Sample Value
Libref	**spdsrv**
Engine	**SASSPDS**

You can also click **Advanced Options** to perform tasks such as pre-assignment and optimization. Click **Next** to access the next page of the wizard.

6 Enter the following settings:

Table 2.29 Server and Connection Information

Field	Sample Value
SAS SPD Server	**SPDSServer** (Use the database server that you selected in the New Server wizard.)
LIBNAME Domain	**spdsrv** (Select the domain name that you entered in the **libname.parms** file.)
Default Login	**(None)** (Keep this default value.)

Click **Next**.

7 Examine the final page of the wizard to ensure that the proper values have been entered. Click **`Finish`** to save the library settings.

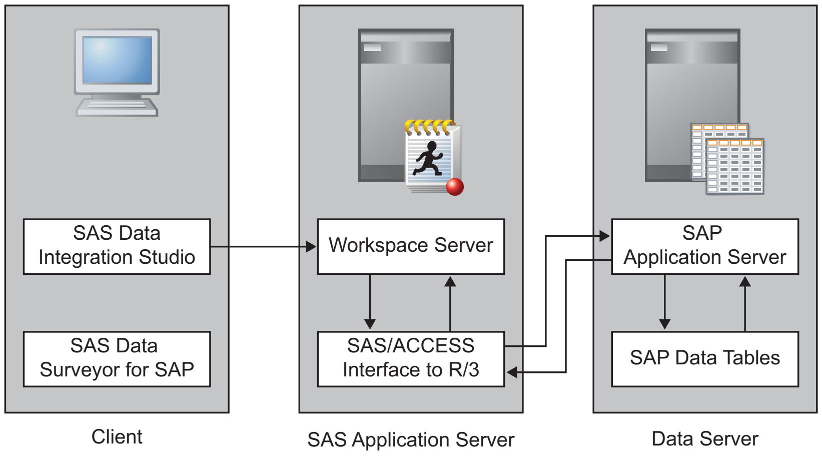
At this point, you can register tables, as explained in “Registering and Verifying Tables” on page 54.

Establishing Connectivity to an SAP Server

Overview to Establishing Connectivity to an SAP Server

The following figure provides a logical view of connecting to an SAP Server as a data source.

Figure 2.12 Establishing Connectivity to an SAP Server



Setting up a connection from SAS to an SAP server is a three-stage process:

1 Register the server.
2 Register the library.
3 Extract SAP metadata, if SAS Data Surveyor for SAP is installed.

This example shows the process for establishing a SAS connection to SAP. It assumes that the following software has already been loaded by using the standard installation wizard:

- SAP RFC library. This is required for communication with SAP.
- SAS/ACCESS Interface to R/3. For z/OS hosts, this installs the SAS RFC server. For these z/OS hosts, this server must be started each time that you start the SAS servers such as the Object Spawner.

Stage 1: Register the Server

To register the SAP server, perform the following steps:

1 Open the SAS Management Console application.

2 Right-click **Server Manager** and select the **New Server** option to access the New Server wizard.

3 Select **SAP Server** from the **Enterprise Applications Servers** list. Then, click **Next**.

4 Enter an appropriate server name in the **Name** field (for example, **SAPServer**). Note that you can supply an optional description if you want. Click **Next**.

5 Enter the following server properties. An SAP 4.6 installation is used as the example:

Table 2.30 Server Properties

Field	Sample Value
Major Version Number	**4**
Minor Version Number	**6**
Software Version	**4.6**
Vendor	**SAP AG**

Click **Next**.

6 Enter the following connection properties:

Table 2.31 Connection Properties

Field	Sample Value
Authentication Domain	**SAPAuth** (You might need to create a new authentication domain. For more information, see “How to Store Passwords for a Third-Party Server” in the *SAS Intelligence Platform: Security Administration Guide*.) Click **New** to access the New Authentication Domain dialog box. Then enter the appropriate value in the **Name** field and click **OK** to save the setting.
Client	**800** (This value is obtained from your SAP administrator.)
Language	**EN** (This value is obtained from your SAP administrator.)

Note: An embedded RFC server is not available for z/OS. For z/OS, click the **Advanced Options** button and enter “host=*rfc-server* port=*rfc-port*” in the **Other option(s) to be appended** text field. Also, select the **Batch Mode** check box. △

7 Select **Application Server** and click **Options** to access the Application Server Host dialog box.

Note: Instead of the Application Server, you might choose other options, as well, including: SAPGUI Logical Name, SAPRFC.INI Logical Name, and Message Servers △

8 Enter the fully qualified name of the server host that was supplied by the SAP administrator (for example, **sapsrv.na.sas.com**) in the **Application Server Host** field. Enter the system number that was supplied by the SAP administrator (for example, **12**) in the **System Number** field. The default access mode is direct

access. In order to run in batch mode, click the **Advanced Options** tab, select the **Batch Mode** check box, and enter into the **Other options** field a value for destgroup such as destgroup="SDSTEST". For batch mode on z/OS, follow the instructions in *Installation Instructions for SAS/ACCESS Interface to R/3 Software*. Then, click **OK** to return to the New Server wizard.

9 Click **Next**.

10 Examine the final page of the wizard to ensure that the proper values have been entered. Click **Finish** to save the wizard settings.

Stage 2: Register the Library

After you have registered the server, you can register the library. To register the library, perform the following steps:

1 In SAS Management Console, expand **Data Library Manager**. Right-click **Libraries**. Then, select the **New Library** option to access the New Library wizard.

2 Select **SAP Library** from the **Enterprise Applications Data** list. Click **Next**.

3 Enter an appropriate library name in the **Name** field (for example, **SAP Library**). You can supply an optional description. Click **Next**.

4 Select an application server from the list and use the right arrow to assign the application server. This step makes the library available to the server and makes the library visible to users of the server. Click **Next**.

5 Enter the following library properties:

Table 2.32 Library Properties

Field	Sample Value
Libref	**SAPLib**
Engine	**SASIOSR3** (Accept the value that is populated automatically.)

Click **Next**.

6 Select the SAP server that you entered in the **Name** field of the New Server wizard (for example, **SAP Server**) by using the **Database Server** drop-down list. Then, click **Next**.

7 Examine the final page of the wizard to ensure that the proper values have been entered. Click **Finish** to save the library settings.

Stage 3: Extract SAP Metadata

If SAS Data Surveyor for SAP is installed, then you can extract metadata about your SAP objects to SAS data sets. Once you have created the SAS data sets, then the tables in your SAP System are available for use in jobs with clients like SAS Data Integration Studio and SAS Enterprise Guide.

The tools for extracting the SAP metadata are provided as a plug-in to SAS Management Console and access to the tool is controlled with role-based access. To enable the extraction tool for role-based access and to extract the SAP metadata, perform the following steps:

1 Using an unrestricted account such as sasadm@saspw, select **Tools ▸ Plug-in Manager** from SAS Management Console.

2 On the Plug-in Manager window, select the **`ExtractionTool`** check box. Click **`OK`**.
3 Assign the ExtractionTool capability to a role with the User Manager plug-in to SAS Management Console, and then associate users or groups with the role. The following list provides two choices:
   - Assign the ExtractionTool capability to an existing role such as Management Console: Advanced.
   - Create a new role, assign the ExtractTool capability to it, and then associate users and groups with the new role.

   For more information about roles, see “How to Assign Capabilities to Roles” in the *SAS Intelligence Platform: Security Administration Guide*.
4 *Important:* Log on to SAS Management Console with an account that has access to the Foundation repository and that is not an unrestricted account.
5 Select **Tools ▶ Extract from BW** or **Tools ▶ Extract from R/3**. For information about using the tools, click **`Help`**.

Special Considerations for SAP

For z/OS operating environments, when you specify the language value on the New Server wizard, use uppercase letters and enclose the value in quotation marks (for example, “EN”).

Registering and Verifying Tables

You need to make sure that the end users of your SAS applications can gain access to tables in your data libraries. The exact steps and authorization requirements vary across applications and data types, but you must always log on to the application, create the needed metadata, and verify the existence of the tables. This example will focus on the process used to verify SAS tables in SAS Management Console.

One important consideration for registering tables is that you must not use an unrestricted user account. This is because the libraries you access often require retrieving database passwords from the metadata repository. An unrestricted user does not have the ability to access passwords stored in metadata. The user account you log in with must have the necessary permissions to access stored database credentials in the metadata. Also, this user account must have the necessary permissions:

- ReadMetadata and WriteMetadata permission in the DefaultACT for the repository
- ReadMetadata and WriteMetadata permission on the library
- WriteMemberMetadata permission to the folder where the table metadata is to be stored
- permission to the underlying data source to read the tables

Verifying your access to tables in SAS Management Console is a two-stage process:

1 Register the tables.
2 View the data in SAS Data Integration Studio.

Stage 1: Register the Tables

To register the tables, perform the following steps:

1 Open SAS Management Console, if necessary. Be sure to select the metadata profile of a user who is not an unrestricted user.

2 Expand the **`Data Library Manager`** node. Then, expand the **`Libraries`** node to see the list of libraries. Right-click the library that contains the tables that you need to import. Then, select the **`Register Tables`** option to access the first page of the Register Tables wizard.

3 Verify that the values shown in the fields in the **`Library details`** group box are correct. Click **`Next`**.

4 Click the tables that you need to select. (Hold down the CTRL key and click to select more than one table.) Click **`Next`**.

5 Examine the final page of the wizard to ensure that the proper values have been entered. Click **`Finish`**.

Note: You can also register tables by using SAS Data Integration Studio and also by using the METALIB procedure. For information about using the METALIB procedure, see Chapter 4, “Managing Table Metadata,” on page 71. △

Stage 2: Verify Access to the Data in a SAS Application

Open an application that can view SAS data in order to view the data in the imported tables and review the data. For example, you can use SAS Data Integration Studio. To use SAS Data Integration Studio to view a registered table, perform the following steps:

1 Navigate to the **`Inventory`** tree and expand the **`Table`** node.

2 Right-click a table that you need to verify and select the **`Open`** option. Examine the data contained in the table in the View Data window.

3 Close the View Data window.

4 (Optional) You can also examine the table’s Properties field. Right-click the table and select the **`Properties`** option.

5 Click the **`Columns`** tab to see column data for the table. Close the Properties window.

Read-only Access for Reporting Libraries

If your site uses libraries for reporting, or for access exclusively by report generating applications such as SAS Information Map Studio and SAS Web Report Studio, then consider setting read-only access for the library. If the library is not set for read-only access, then even when reporting applications raise a query against the library, the underlying SAS session opens the data in read-write mode. In this case, simultaneous queries against the same library might be prevented. Simply put, if clients access the information in read-only mode, then set the library to read the data source in read-only mode.

Table 2.33 Setting a Library for Read-Only Access

Library Type	**Where to Set Read-only Access**
All Database Data Libraries	Advanced Options dialog box, **`Input/Output`** tab, option Data access level for connection
SAS BASE Library	Advanced Options dialog box, **`Options for any host`** tab
SAS Information Map Library	Always read-only, no configuration needed
SAS XML Library	Library properties page of the wizard, **`Library Access`** option

Library Type	Where to Set Read-only Access
SAS Scalable PerformanceData Engine Library	Advanced Options dialog box, **Options for any host** tab
SAS/SHARE REMOTE Engine Library	Advanced Options dialog box, **Libname Options** tab

Setting UNIX Environment Variables for SAS/ACCESS

If you are attempting to connect to data sources located on UNIX by using SAS/ACCESS, you must set environmental variables so SAS servers can access the database. Each database vendor and operating system vendor requires specific environment variables to be set. A typical value is LD_LIBRARY_PATH. When using ODBC, two typical values are ODBCINI and ODBCINSTINI. For more information about the correct environment variables, go to the Install Center at **http://support.sas.com/documentation/installcenter/92/documents/index.html**, and use the operating system and SAS version to locate the appropriate SAS Foundation Configuration Guide.

To set the appropriate environment variables in the **!SASROOT/bin/sasenv_local** file, perform the following steps:

1 Edit the **sasenv_local** file and add the variables. This example uses sample values, substitute the proper paths:

```
ODBCINI=/opt/Composite_Software/CIS_4.6.0/odbc.ini
export ODBCINI

ODBCINSTINI=/opt/Composite_Software/CIS_4.6.0/odbcinst.ini
export ODBCINSTINI

LD_LIBRARY_PATH=/opt/Composite_Software/CIS_4.6.0/apps/odbc/lib:\
/opt/oracle/app/oracle/product/11.1.0/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH

ORACLE_HOME=/opt/oracle/app/oracle/product/11.1.0
export ORACLE_HOME
```

2 In SAS Management Console, right-click the **Workspace Server** connection and select **Validate** to verify that the workspace server starts correctly with the new environment variables.

3 Restart the SAS/SHARE and SAS/CONNECT servers, if they are present in the deployment and reference the SAS/ACCESS library.

Troubleshooting SAS/ACCESS Connections to RDBMS

This section provides information about troubleshooting a SAS/ACCESS library configuration when registering tables fails. To troubleshoot the SAS/ACCESS library, perform the following steps:

1 From SAS Management Console, right-click the library icon and select **Display LIBNAME Statement**.

2 Start SAS on the SAS server host and issue the LIBNAME statement displayed from SAS Management Console.

  a If the SAS log indicates failure, check the following items:

    i If this is UNIX environment, check “Setting UNIX Environment Variables for SAS/ACCESS” on page 56.
    ii Check and revise the LIBNAME statement. For more information about LIBNAME statements for SAS/ACCESS engines, see *SAS/ACCESS for Relational Databases: Reference*. If you are successful at this stage, then use the **Properties** tab of the library to reconfigure the library.
    iii Confirm that SAS/ACCESS is installed correctly. For installation information, go to the Install Center at **http://support.sas.com/documentation/installcenter/92/documents/index.html** and use the operating system and SAS version to locate the appropriate SAS Foundation Configuration Guide.

  b If the connection succeeds, perform the following steps:

    i Run the DATASETS procedure:

```
proc datasets library = libref;
quit;
```

    ii If no members are returned, then check the schema value by running the next step or contacting your database administrator.

3 Log on with the user account to the host where the SAS server is running, and use the native database client to connect to the database. If this fails, confirm the user account has file system privileges to the database client binaries and libraries.

CHAPTER 3

Assigning Libraries

Overview of Assigning Libraries

What Does It Mean to Assign a Library?

In Chapter 2, "Connecting to Common Data Sources," on page 17, you learned how to register libraries in metadata and assigned the libraries to SAS servers. These libraries represented such things as the set of SAS data sets in a directory or the set of tables in a database schema. This chapter explains what assigning a library accomplishes and how to assign a library to SAS servers. By assigning a library to a SAS server, you are identifying the server that can access the libraries, controlling how the SAS server accesses the library, and making the library visible to users of the SAS server.

Assigning a library means letting a SAS session know that a libref—a shortcut name—is associated with the information that the SAS session needs to access a data library. For example, if you were writing a SAS program that needed to access a library of SAS data sets, your program might include the following statement:

```
LIBNAME ORGOLD BASE 'C:\SAS\SASConfig\Lev1\SASApp\Data\orgold'
```

In this case, the libref ORGOLD tells the SAS session that it should access data sets in the directory **C:\SAS\SASConfig\Lev1\SASApp\Data\orgold** using the Base SAS data-access engine.

SAS Intelligence Platform clients such as SAS Data Integration Studio, SAS OLAP Cube Studio, and SAS Information Map Studio generate SAS code that uses librefs. Before this code can execute, the corresponding library must be assigned, and the server that executes the code must know about that assignment.

Pre-assigning Libraries

There are two ways in which a server can find out about a library reference. One way is for you, as the administrator, to configure the environment so that the server finds out about the libref at server startup. This approach is referred to as *pre-assigning* the library, because the libref is established before any code that uses that libref is submitted. The other way is to let the client application define the libref for a server when it generates code for submission to that server.

Deciding whether to pre-assign a library or not has important consequences. One factor to keep in mind is that pre-assigning an excessive number of libraries can slow the execution of SAS jobs for all users. Other factors are described in "Data-Access Engines and the Metadata Engine" on page 61. SAS clients and stored processes can access a library using one of two engines:

- the engine specified in the library's metadata. This is the Base SAS engine for libraries of SAS data sets, the ORACLE engine for Oracle libraries, and so forth.
- the Metadata LIBNAME Engine.

Which engine you use affects security and determines what operations are possible.

Note: If you are defining a pre-assigned DBMS library, do not use the Pre-Assigned Library resource template. Register the library using the appropriate DBMS library template. Specify that the library is pre-assigned using the Advanced Options dialog box in the New Library wizard or the library Properties window. △

If you pre-assign libraries, *you* control which engine is used to access the data. If you do not pre-assign a library, the client that needs to access that library decides which engine to use, and different clients use different strategies. For example, SAS Data Integration Studio and SAS OLAP Cube Studio always use the engine stored in the library's metadata, while SAS Enterprise Guide can use either the metadata engine or its native engine. For more information, see "Managing Libraries" in the chapter "Managing Metadata Objects" in *SAS Management Console User's Guide*.

Having the server process assign libraries upon start-up based on information in the metadata results in library assignments that are identical and guaranteed across all SAS client applications and servers. Some environments where this approach to assigning libraries is desirable include the following:

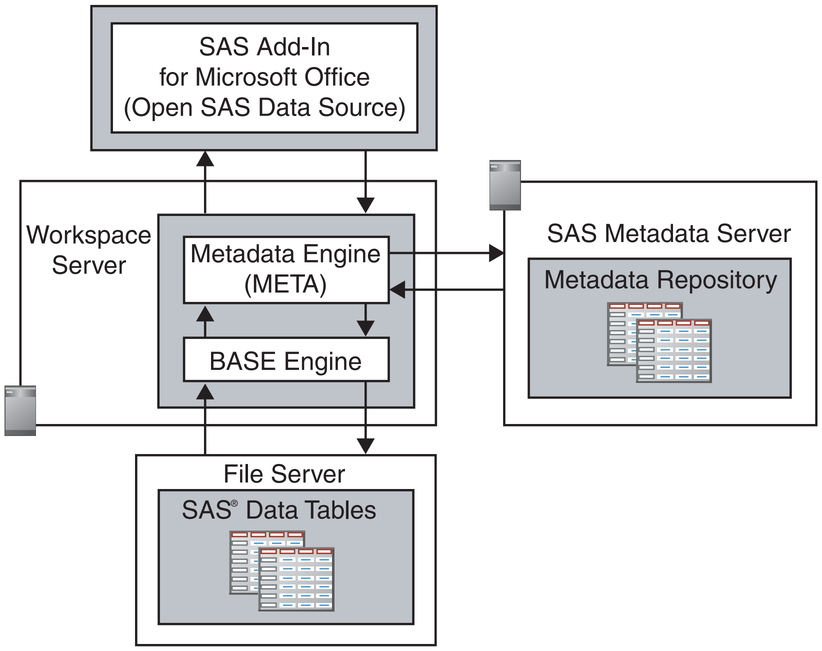
- environments where users are executing stored processes, and you do not want programmers having to manage library assignments in their code or in autoexec files.
- environments where the DATA Step Batch Server is used to execute jobs created by SAS Data Integration Studio, and library assignments for these jobs should be identical to assignments used when the process was created.
- environments where SAS Enterprise Guide or SAS Add-In for Microsoft Office users will be running tasks that need to create tables in the library that is registered in metadata. When you register a client-assigned library (a library that is not pre-assigned), SAS Enterprise Guide and SAS Add-In for Microsoft Office assign the library to use the metadata engine by default. Metadata engine libraries do not update metadata after changes to the underlying data source. Metadata can be updated to reflect changes to the underlying data source with PROC METALIB or with the register tables function of SAS Management Console.

When libraries are assigned by the client application, each application can assign the library in a way that is most suitable for its intended user base, and library connections are established only if needed. When libraries are assigned by the server, each library is available to all back-end server processes and is allocated exactly the same way for all client applications. A mixture of some server-assigned and some client application-assigned libraries will most likely be required to meet the needs of all the users in your environment.

Data-Access Engines and the Metadata Engine

As mentioned previously, when you access the data in a data library, you can use the data-access engine stored in the metadata definition of the library, or you can use the metadata engine. As shown in the following figure, the metadata engine invokes the Base SAS engine stored in the metadata.

Display 3.1 Metadata Engine Invocation of the Base SAS Engine



One of the key enhancements made in SAS®9 has been the introduction of the SAS Open Metadata Architecture authorization facility. This authorization facility gives you, the administrator, the ability to control which users can access which metadata objects, such as SASLibrary, PhysicalTable, and LogicalServer. You manage their access to metadata by setting ReadMetadata and WriteMetadata permissions on the object or on the repository.

As depicted in the previous figure, when SAS users expand a library that they have ReadMetadata access to and that has been assigned to use the metadata engine, the engine first sends a request to the SAS Metadata Server asking for the users' metadata permissions on the tables in the library. A list of tables for which the users have at least ReadMetadata access will be returned and presented to them for selection. If they then attempt to perform some action against one of those tables, such as opening it, the metadata engine sends a query to the metadata server for the users' metadata permission on the table. If the users or the group to which they belong have at least Read access to the table, the metadata engine will call upon the engine specified in the metadata to handle the request, and the table will be opened into the client application for reading.

In contrast, when client applications access a library that does not use the metadata engine, they first contact the metadata server and request access to a metadata object as the user. The metadata server then queries the SAS authorization facility to determine whether users have ReadMetadata, CheckInMetadata, or WriteMetadata permission to the object. These metadata-based permissions are the only permissions checked by the metadata server. Users attempting to access a table in a SAS metadata-based library that is pre-assigned by the server will be successful if they have ReadMetadata access to the library and, in the case of SAS Data Integration Studio, SAS OLAP Cube Studio, and SAS Information Map Studio, ReadMetadata access to the table. Notice that when the library does not use the metadata engine, the data-level authorizations of Read, Write, Create, and Delete are never checked.

If you want to use the SAS authorization facility to control Read, Write, Create, and Delete permissions, then you need to assign the server-side library to use the metadata engine in an autoexec file. When used with its default options, the metadata engine queries the metadata server for these metadata-based permissions. The SAS authorization facility must be queried for data-level permissions. When libraries are defined in an autoexec file through a LIBNAME statement, they are always pre-assigned.

The general form of a LIBNAME statement for the metadata engine is as follows:

```
LIBNAME libref META LIBID=id|LIBURI=URI-format|LIBRARY=name
   <connection-options><engine-options>;
```

Therefore, a META LIBNAME statement for the Orion Gold Customers library that is registered in metadata would look something like the following:

```
LIBNAME ORGOLD META library="Orion Gold Customers";
```

This is the minimum information that you need to supply in the LIBNAME statement itself. However, this statement works only if the META* options that contain the information necessary to connect to the metadata server have already been specified. These options, METASERVER, METAPORT, METAREPOSITORY, and METAPROTOCOL, are configured in the **sasv9.cfg** file already if you used the SAS Deployment Wizard to set up your environment.

Having data requests flow through the metadata engine before they reach the engine that actually fulfills the request provides an important capability: the metadata engine enforces the data-level authorizations that are available in the SAS Authorization Manager. These include the Read, Write, Create, and Delete permissions. The other data-access engines ignore these permissions.

At the same time, using the metadata engine takes away some capabilities. Most important, the metadata engine does not automatically create, update, or delete metadata when changes are made to the underlying data source.

Using Libraries That Are Not Pre-assigned

Default assignment for libraries

By default, newly created libraries are not pre-assigned. When a library is not pre-assigned, the library is assigned by using the data-access engine that best suits the client application and its intended user base. Thus, the default assignments for applications such as SAS Data Integration Studio, SAS Add-In for Microsoft Office, SAS Enterprise Guide, SAS OLAP Cube Studio, SAS Enterprise Miner, and SAS

Information Map Studio are used. For example, if you do not pre-assign the library, SAS Data Integration Studio assigns the library using the engine specified in metadata (such as BASE). This method avoids the data-level authorizations of Read, Write, Create, and Delete. This approach is a best practice, because it is assumed that in most cases SAS Data Integration Studio developers are building processes that create or update tables in the library and that the underlying engine is the only engine that should be used for data-populating tasks.

How Do the Different Platform Clients Assign Libraries?

When libraries are not pre-assigned, each SAS platform client assigns libraries. Allowing each application to assign libraries as it deems appropriate for its user base results in the optimal security model for environments where users have different data access requirements to a library and where you want to capitalize on using metadata decisions enforced by the SAS authorization facility on top of the operating system or RDBMS authorization layer. An example of such an environment would be one with clients running at least SAS Enterprise Guide and SAS Data Integration Studio. In this environment, SAS Data Integration Studio processes update tables that are in turn used in ad hoc analysis within SAS Enterprise Guide. The SAS Data Integration Studio processes need to specify tables in the library as target tables (output), whereas the SAS Enterprise Guide user's activities largely involve querying and analyzing chunks of data (input).

Because SAS Data Integration Studio processes typically update or create target tables, when SAS Data Integration Studio assigns the library it does not use the metadata engine. Instead, it assigns the library using the engine specified in the metadata. Because SAS Data Integration Studio only works with tables that are registered in the metadata repository, you can use the SAS authorization facility to control a client's access to tables by setting ReadMetadata, WriteMetadata, and CheckInMetadata permissions on the library and table metadata objects.

SAS Information Map Studio always assigns the library by using a LIBNAME statement and the engine specified in the metadata, unless the library is explicitly defined by a SAS administrator (or SAS Data Integration Studio administrator) to use the metadata engine.

Note: The metadata authorization layer supplements operating system- and RDBMS-level security. It does not replace it. Operating system and RDBMS authorization layers can and should always be used as the first means of securing access to tables. △

On the other hand, the SAS Add-In for Microsoft Office and SAS Enterprise Guide (shown in the following table) assign the library using the metadata engine by default, so that data-level authorizations of Read, Write, Create, and Delete, which are specified in the metadata, are enforced. If defining libraries so that they are not pre-assigned seems like a potential option for your environment, then you will want to explore this topic a little further and learn how to ensure that these libraries will be available to server processes that do not receive direct requests from client applications. For example, you will need to learn how to manually assign the library in server processes such as the stored process server and DATA Step Batch Server (if present), as discussed in the next section.

Table 3.1 Platform Client Default Library Assignments

Application	Pre-assigned	Library Engine Used	Minimum Metadata Authorizations Required
SAS Add-In for Microsoft Office	No	META	Library: ReadMetadata Table: ReadMetadata and Read
SAS Enterprise Guide	No	META	Library: ReadMetadata Table: ReadMetadata and Read
SAS Data Integration Studio	No	Underlying data engine	Library: ReadMetadata Table: ReadMetadata
SAS OLAP Cube Studio	No	Underlying data engine	Library: ReadMetadata Table: ReadMetadata
SAS Information Map Studio	No	Underlying data engine	Library: ReadMetadata Table: ReadMetadata

Processing Stored Processes When the Library is Not Pre-assigned

In the SAS Intelligence Platform, a stored process is a SAS program that is stored on a server and can be executed as requested by clients who have ReadMetadata access to the stored process program's metadata. SAS Stored Processes can be executed by either a SAS Workspace Server or a SAS Stored Process Server. If a library is not pre-assigned, it is the responsibility of the stored process program's author or the SAS administrator to ensure that the library is assigned to a specific location and physical path. This can be done either directly in each stored process program or from an external file that is linked to the stored process with an %INCLUDE statement.

These methods have the following advantages and disadvantages:

- Method: Define a metadata engine library in the stored process program.
  - Advantage: Data-level authorizations specified for the library and table metadata objects are enforced by the SAS authorization facility. Note that these permissions are enforced for the server's identity (usually SAS General Servers), not the client's.
  - Disadvantage: Library and table metadata for any table called in the program must be registered in the metadata repository, thus preventing a stored process from accessing tables that might reside in the library but are not registered in metadata.
  - Disadvantage: Changes to the library metadata object's name or repository location would require that each stored process that references the library be updated.
  - Disadvantage: Metadata inconsistencies or corruptions can result if the stored process modifies a table's structure through the library. Examples of this modification include adding or removing columns.
- Method: Define the library in the stored process program and use only the underlying data engine.

  - Advantage: A table does not have to be registered in the metadata repository in order for the stored process to access it.
  - Advantage: Tables in the library can be re-created or updated and new tables created without directly impacting the metadata. Note, however, that changes to the structure of a table that has been registered previously in the metadata repository can still cause synchronization issues between the table and the metadata.
  - Disadvantage: The metadata repository is no longer a single point of management, because library definitions are stored in multiple places.
  - Disadvantage: Changes to the library path or directory would require that each stored process that references the library be updated.
  - Disadvantage: The SAS authorization facility has no role in managing access to tables called by the stored process. Thus, the SAS General Server User can access data in any table in the library for which he has been granted Read access at the OS or RDBMS layer.
- Method: Store the library assignments in an external file and then include the file in the stored process program.
  - Advantage: Library assignments are defined in one file or directory location that all stored process programs can reference.
  - Advantage: Multiple files that contain library assignments can be created and referenced as needed in the stored process so that things such as connections to databases are established only when absolutely required.
  - Advantage: Other advantages depend on how the library is defined in the file. See the two preceding methods.
  - Disadvantage: The files referenced in the stored process must be created and maintained by someone who has Write (and Modify) access to the file's location on the system.
  - Disadvantage: Stored processes created through point and click applications such as SAS Enterprise Guide must be modified manually to replace the library assignment with manually generated %INCLUDE syntax.
  - Disadvantage: Changes to the file's location or name requires updating all the stored processes that include the file.

Pre-assigning Libraries Using Engines Other Than the Metadata Engine

Overview of Pre-assigning Libraries Using Engines Other Than the Metadata Engine

Pre-assigning a library ensures that the library will always be available to and assigned by SAS server processes on a server-by-server basis when the server starts, rather than assigned by the client application or later in SAS code. Two types of pre-assignment are possible. First, you can pre-assign a library so that it will be accessed by the engine defined in the metadata. Second, you can pre-assign a library so that it will be accessed by the metadata engine. In either case, pre-assignment allows you to designate an assignment method for use by all of the applications that use the library.

Note: Pre-assigning a large number of libraries can have a negative impact on the execution time of SAS programs for all users. You should therefore be judicious in deciding whether to pre-assign a library or not. △

Pre-assigning a library to an engine other than the metadata engine engine is a two-stage process:

1 Use SAS Management Console to flag the library as pre-assigned and to assign the library to the servers.

2 Edit configuration files so the assigned servers can retrieve library metadata by adding the METAAUTORESOURCES SAS system option to the server's **`sasv9_usermods.cfg`** file.

Stage 1: Flag the Library as Pre-assigned

Assume that we are pre-assigning the Orion Gold Customers library. The library can be configured to be assigned by the server process by either selecting the **`Library is pre-assigned`** advanced option when the library is being registered or by modifying the library's properties after the fact. To pre-assign a library, perform the following steps:

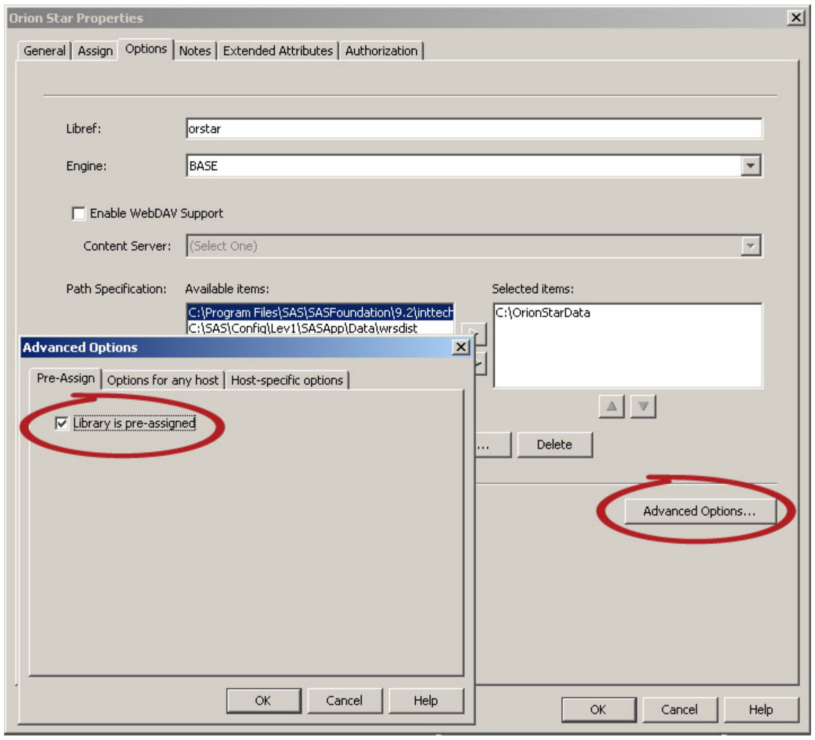
1 From SAS Management Console, select **Data Library Manager ▶ Libraries ▶ Orion Gold Customers ▶ Properties**.

2 Select the **`Options`** tab.

3 Click the **`Advanced Options`** button.

4 Select the check box on the **`Pre-Assign`** tab.

Figure 3.1 Library is Pre-assigned Option



Libref: orstar

Engine: BASE

☐ Enable WebDAV Support

Content Server: (Select One)

Path Specification: Available items: C:\Program Files\SAS\SASFoundation\9.2\inttech; C:\SAS\Config\Lev1\SASApp\Data\wrsdist

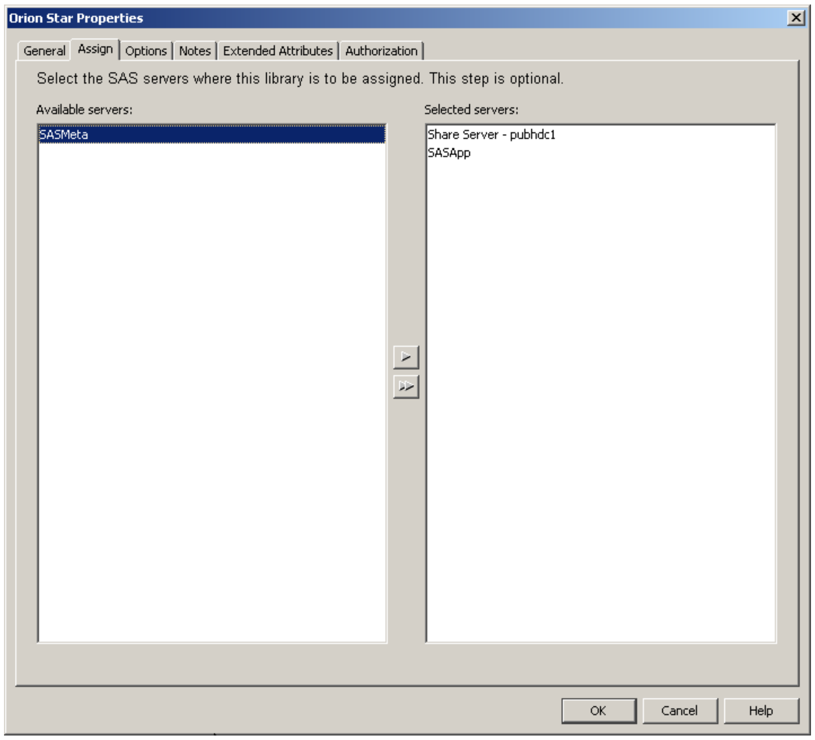
Selected items: C:\OrionStarData

☑ Library is pre-assigned

5 Click **`OK`** on the Advanced Options dialog box.

6 Click the **`Assign`** tab on the library properties window.

7 Select the servers you want to pre-assign the libraries to. Click **`OK`** when you are finished.



Select the SAS servers where this library is to be assigned. This step is optional.

Available servers:

SASMeta

Selected servers:

Share Server - pubhdc1
SASApp

Stage 2: Edit Configuration Files

In the previous stage, we set the libraries that are available to be pre-assigned, and we selected which servers can retrieve the library definitions from metadata. In this stage, we edit the **`sasv9_usermods.cfg`** file for the servers we selected and add a METAAUTORESOURCES SAS system option so that as those servers start, they read the library definitions from metadata.

Note: This stage is not needed for workspace servers, pooled workspace servers, stored process servers, SAS/SHARE servers, or OLAP servers. Those server types automatically read metadata when they start and assign the libraries. △

To edit the configuration files, perform the following steps:

1 For each SAS/CONNECT server, edit the following file:

SAS-config-dir\**`Lev1`**\SASApp\**`ConnectServer\sasv9_usermods.cfg`**

Add the following SAS system option:

```
-metaautoresources "omsobj:ServerComponent?@Name='SASApp'"
```

2 For each DATA Step Batch server, edit the following file:

SAS-config-dir\**`Lev1`**\SASApp\**`DataStep\sasv9_usermods.cfg`**

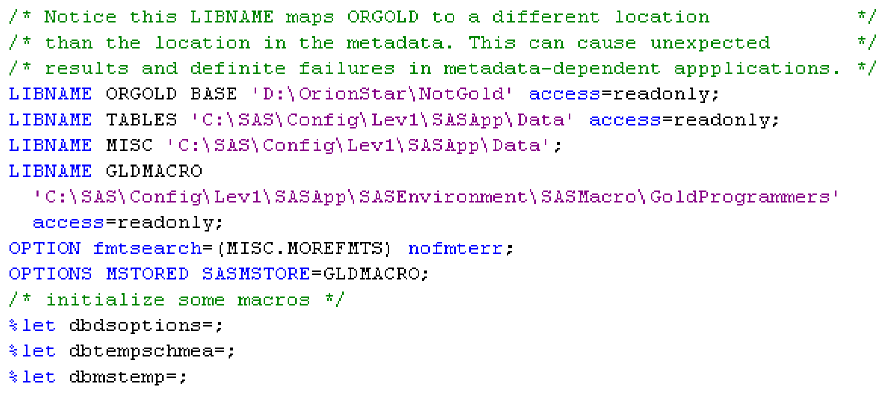
Add the following SAS system option:

```
-metaautoresources "omsobj:ServerComponent?@Name='SASApp'"
```

Pre-assignment Using Information in an Autoexec File

Pre-assigning libraries in an autoexec file is not a recommended practice because library assignments can be stored in two places, the autoexec file and metadata. Having configuration information in two places increases maintenance. An autoexec file is a text file that contains SAS statements that are executed when the server process starts. If an autoexec file is used in your environment, it is important to note that libraries assigned by an autoexec file take precedence over same-named libraries assigned by to the server in metadata. (Use the autoexec file created during installation, which is *SAS-config-dir*`\Lev1\SASApp\appserver_autoexec_usermods.sas`.) For example, if ORGOLD is registered in the metadata to be pre-assigned, and ORGOLD is also defined in an autoexec for the same server, the ORGOLD library is assigned using the LIBNAME information from the autoexec file. Simply put, the library assignment in the autoexec file always takes precedence.

Display 3.2 Library Assignment in an Autoexec File



```
/* Notice this LIBNAME maps ORGOLD to a different location           */
/* than the location in the metadata. This can cause unexpected      */
/* results and definite failures in metadata-dependent appplications. */
LIBNAME ORGOLD BASE 'D:\OrionStar\NotGold' access=readonly;
LIBNAME TABLES 'C:\SAS\Config\Lev1\SASApp\Data' access=readonly;
LIBNAME MISC 'C:\SAS\Config\Lev1\SASApp\Data';
LIBNAME GLDMACRO
  'C:\SAS\Config\Lev1\SASApp\SASEnvironment\SASMacro\GoldProgrammers'
  access=readonly;
OPTION fmtsearch=(MISC.MOREFMTS) nofmterr;
OPTIONS MSTORED SASMSTORE=GLDMACRO;
/* initialize some macros */
%let dbdsoptions=;
%let dbtempschmea=;
%let dbmstemp=;
```

Pre-assigning Libraries to Use the Metadata Engine

The metadata engine is a data access engine that enforces the data-level permissions of Read, Write, Create, and Delete that are set on table objects in the repository. It also enforces the Create and Delete permissions that are set on library objects. The metadata engine acts as a gatekeeper that determines which users can access which metadata-based libraries and tables. Note that this is a level of security implemented in SAS as a supplement to securing the data source. Do not rely on the metadata engine as the single security mechanism.

Note: The METAOUT= option specified in step 3 of the following task is important. For information about the LIBNAME statement for the metadata engine, see *SAS Language Interfaces to Metadata*. △

To register a library that uses the metadata engine, perform the following steps:

1 Register the library in the SAS Metadata Repository.

2 Mark the library as pre-assigned.

3 Construct a LIBNAME statement that uses the same libref specified in the metadata and META as the engine:

```
LIBNAME ORGOLD LIBRARY='Orion Gold'  REPNAME='Foundation' METAOUT=DATA;
```

Note: The METAOUT=DATA option permits read, create, update, and deleting tables. Another value is METAOUT=DATAREG. This choice permits read, update, and delete of tables registered in metadata as well as creating new tables; however, new tables cannot be read until they are registered in metadata. △

4. Add the metadata LIBNAME statement to an autoexec file. During the configuration process, the SAS Deployment Wizard created a single file named **`appserver_autoexec_usermods.sas`** that controls all component servers of the SAS application server and files named **`autoexec_usermods.sas`** for each of the component servers of the application server:
   - **`appserver_autoexec_usermods.sas`**

     Use this file if you want all the SAS application server components registered to the application server, such as an OLAP server, a workspace server, and so on, to access the library with the metadata engine in the same way.
   - **`autoexec_usermods.sas`**

     Use this file to modify one of the SAS application server components, such as the workspace server, to use the metadata engine for accessing a library, but to leave the other server components unchanged. If this is your choice, then note that the autoexec_usermods.sas file is located within a sub-directory, such as **`C:\SAS\Config\Lev1\SASApp\WorkspaceServer\`**.
5. Restart the object spawner and any server processes whose autoexec files have been modified. For information about restarting the servers, see “Starting, Stopping, and Pausing Servers” in the *SAS Intelligence Platform: System Administration Guide*.
6. Use SAS Management Console to grant read, write, create, and delete privileges to users or groups as appropriate for your site.

Note: Remember that for libraries using the metadata engine, an administrator must register tables after any create, update, or delete changes in metadata. You can register table metadata by using PROC METALIB or SAS Management Console. △

Verifying Pre-assignments by Reviewing the Logs

After you specify that a library is to be pre-assigned by the server, the SAS server process will start as follows:

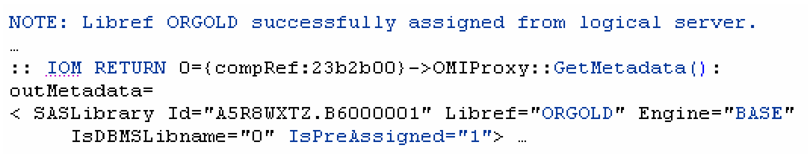
1. Connect to the metadata server.
2. Retrieve library metadata.
3. Assign the library using the engine specified in the library metadata.

For example, if the Orion Gold Customers library is pre-assigned to the workspace server, then the library assignment would be equivalent to a SAS programmer submitting a LIBNAME statement such as the following:

```
LIBNAME ORGOLD BASE "D:\OrionStar\Gold";
```

In the case of an IOM server, such as the workspace server, you can verify the pre-assignment of this library by the server process by enabling logging and observing the note generated from the first GetMetadata method call in the server's log, as in the following sample log:

Display 3.3 Verification of Pre-assignment in a Server Log



```
NOTE: Libref ORGOLD successfully assigned from logical server.
…
:: IOM RETURN 0={compRef:23b2b00}->OMIProxy::GetMetadata():
outMetadata=
< SASLibrary Id="A5R8WXTZ.B6000001" Libref="ORGOLD" Engine="BASE"
     IsDBMSLibname="0" IsPreAssigned="1"> …
```

For non-IOM servers using the METAAUTORESOURCES option, a note like the following is written to its log file:

```
NOTE: Libref ORGOLD successfully assigned from logical server.
```

For information about enabling the logging, see "Capture XML Information in the SAS Metadata Server Log" in the *SAS Intelligence Platform: System Administration Guide*. To verify pre-assignment, set the logging level to **1**.

CHAPTER 4

Managing Table Metadata

Overview of Managing Table Metadata

As explained in "Registering and Verifying Tables" on page 54, one way to create metadata for the tables in a library is to use the Register tables feature of SAS Management Console. You can also create this metadata programmatically by using PROC METALIB. In addition, PROC METALIB provides you with options for maintaining your table metadata that are not available in SAS Management Console. For example, by default PROC METALIB creates metadata definitions for any physical tables that are not registered in the metadata—for instance, tables that have been added since the table definitions were first created—and updates the table definitions for tables that have been altered since they were registered.

By using optional statements, you can also use PROC METALIB to perform the following tasks:

- □ Delete table definitions for tables that have been removed from the library.
- □ Produce a report that lists the changes made by the procedure or the changes that will be made when the procedure is executed.
- □ Operate on a subset of the tables in a library.

Note: For detailed information about PROC METALIB and its syntax, see "METALIB Procedure" in *SAS Language Interfaces to Metadata*. △

Note: PROC METALIB cannot work with a library whose metadata is defined by using the Pre-assigned Library resource template. When pre-assigning a library, be sure to choose the resource template specific to the type of data source library you are creating and select the **`This library is pre-assigned`** check box. Do not use the specialized Pre-Assigned Library template. Also, if the library is pre-assigned and you run the PROC METALIB in a Foundation SAS session, you must make sure the library is allocated by either submitting a LIBNAME statement for the library in that SAS session, or by using the METAAUTORESOURCES option to access the library through a SAS server to which the library is assigned. △

The remainder of the chapter presents examples of how PROC METALIB is commonly used. The examples assume that you have set the following metadata server connection options in your SAS session:

```
options METAUSER = "metadata-server-userid"
        METAPASS = "metadata-server-password"
        METAPORT = metadata-server-port
        METASERVER = "metadata-server-machine";
```

If you have not set these options, you can use PROC METALIB parameters to specify this information.

Creating Table Metadata for a New Library

When you first register a SAS library, it has no related table metadata. You can add this metadata by using the Register Tables wizard in SAS Management Console (see "Overview of Managing Table Metadata" on page 71), or by using PROC METALIB. Before you can successfully run PROC METALIB code, you must have Create, ReadMetadata, and WriteMetadata access to the library metadata object.

The following example shows how to use PROC METALIB to create initial table definitions for the tables in a library. The REPORT statement causes the procedure to write information to SAS output about the table definitions that it creates.

```
proc metalib;
omr (library="sas91 lib2" repname="Meta Proc repos" );
report;
run;
```

The report that this code writes would resemble the following sample.

```
                      The METALIB Procedure

                 Summary Report for Library sas91 lib2
                      Repository Meta Proc repos
                               17MAR2005

                       Metadata Summary Statistics

                   Total tables analyzed          2
                   Tables Updated                 0
                   Tables Added                   2
                   Tables matching data source    0
                   Tables not found               0


--------------------------------------------------------------------------------------
                                      Tables Added
--------------------------------------------------------------------------------------


Metadata Name                     Metadata ID          SAS Name

COUNTRY                           A5HJ58JU.AX001LPV    COUNTRY
POSTAL                            A5HJ58JU.AX001LPW    POSTAL
```

Assessing Potential Changes in Advance

Before you use PROC METALIB to update existing table metadata, it is a good idea to execute the procedure with the NOEXEC and REPORT statements. The NOEXEC statement tells the procedure not to actually add, update, or delete any metadata. The REPORT statement tells the procedure to create a report that explains what actions it would have taken if the NOEXEC statement had not been present. If you want to make all of the changes that are shown in the report, you can then remove the NOEXEC statement and rerun the procedure to update the metadata.

The following example shows how to use the NOEXEC and REPORT statements to assess potential metadata changes:

```
ods html "myfile";
proc metalib;
omr (library="SAS91 lib" repname="Meta Proc repos" );
update_rule=(delete);
noexec;
report;
run;
```

Note: The UPDATE_RULE statement tells the procedure to delete table definitions for any tables that have been deleted from the library. For more information about this statement, see “Changing the Update Rule” on page 75. △

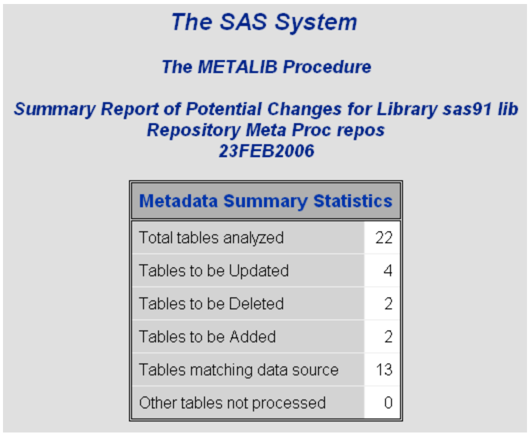
Here is the resulting SAS log:

```
55   proc metalib;
56   omr (library="SAS91 lib"  repname="Meta Proc repos" );
57   update_rule=(delete);
58   noexec;
59   report;
60   run;

NOTE: A total of 22 tables were analyzed for library "SAS91 lib".
NOTE: NOEXEC statement in effect.  No Metadata changes applied.
NOTE: Metadata for 4 tables would have been updated.
NOTE: Metadata for 2 tables would have been deleted.
NOTE: Metadata for 2 tables would have been added.
NOTE: Metadata for 13 tables matched the data sources.
NOTE: 0 other tables were not processed due to error or UPDATE_RULE.
NOTE: PROCEDURE METALIB used (Total process time):
      real time           31.26 seconds
      cpu time            8.12 seconds
```

SAS output is the default. This example specifies ODS output. Specifying ODS produces reports in both ODS and SAS output formats unless you specify the following to suppress SAS output:

```
ods listing close;
```

Here is the resulting ODS output:

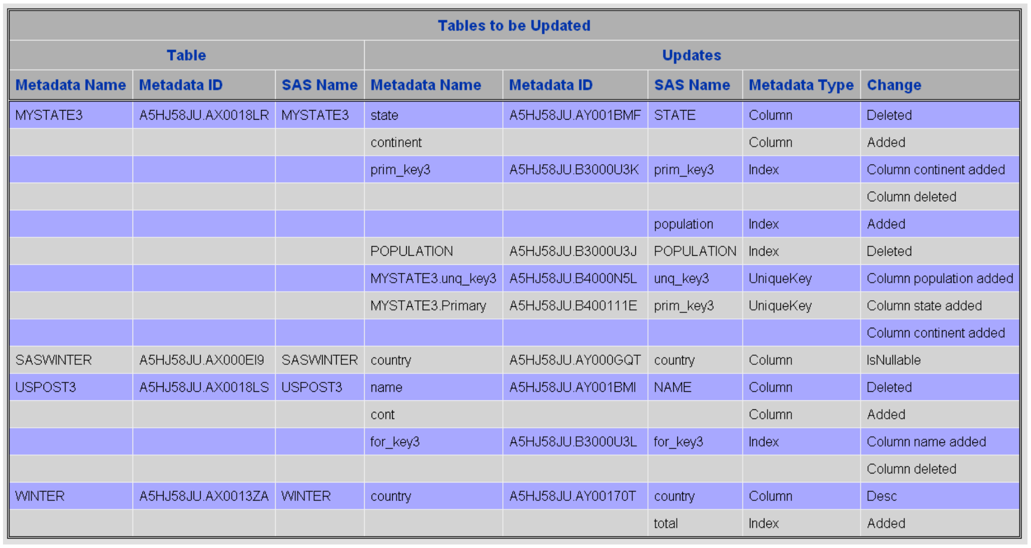


The SAS System

The METALIB Procedure

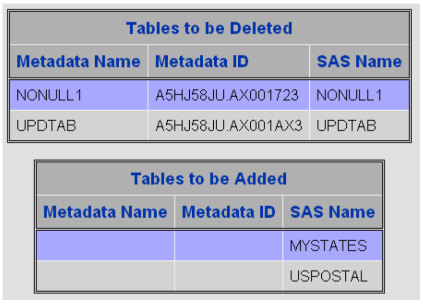
Summary Report of Potential Changes for Library sas91 lib
Repository Meta Proc repos
23FEB2006

Metadata Summary Statistics	
Total tables analyzed	22
Tables to be Updated	4
Tables to be Deleted	2
Tables to be Added	2
Tables matching data source	13
Other tables not processed	0



Tables to be Updated

Table			Updates				
Metadata Name	**Metadata ID**	**SAS Name**	**Metadata Name**	**Metadata ID**	**SAS Name**	**Metadata Type**	**Change**
MYSTATE3	A5HJ58JU.AX0018LR	MYSTATE3	state	A5HJ58JU.AY001BMF	STATE	Column	Deleted
			continent			Column	Added
			prim_key3	A5HJ58JU.B3000U3K	prim_key3	Index	Column continent added
							Column deleted
					population	Index	Added
			POPULATION	A5HJ58JU.B3000U3J	POPULATION	Index	Deleted
			MYSTATE3.unq_key3	A5HJ58JU.B4000N5L	unq_key3	UniqueKey	Column population added
			MYSTATE3.Primary	A5HJ58JU.B400111E	prim_key3	UniqueKey	Column state added
							Column continent added
SASWINTER	A5HJ58JU.AX000EI9	SASWINTER	country	A5HJ58JU.AY000GQT	country	Column	IsNullable
USPOST3	A5HJ58JU.AX0018LS	USPOST3	name	A5HJ58JU.AY001BMI	NAME	Column	Deleted
			cont			Column	Added
			for_key3	A5HJ58JU.B3000U3L	for_key3	Index	Column name added
							Column deleted
WINTER	A5HJ58JU.AX0013ZA	WINTER	country	A5HJ58JU.AY00170T	country	Column	Desc
					total	Index	Added



Tables to be Deleted

Metadata Name	Metadata ID	SAS Name
NONULL1	A5HJ58JU.AX001723	NONULL1
UPDTAB	A5HJ58JU.AX001AX3	UPDTAB

Tables to be Added

Metadata Name	Metadata ID	SAS Name
		MYSTATES
		USPOSTAL

Updating Your Table Metadata to Match Data in Your Physical Tables

Adding and Updating Table Metadata

By default, PROC METALIB creates table definitions for any tables in the library that do not have table definitions and updates any table definition that does not reflect the current structure of the table that it represents. It does not, however, delete table metadata.

Use REPORT when you want an output listing that summarizes metadata changes, either before changes are made (by using NOEXEC) or to see afterward what changes were actually made. SAS output is the default.

Example: Default PROC METALIB Behavior

The following example uses the default PROC METALIB behavior. Summary notes are written to the SAS log regardless of whether you request a report. Unlike the example shown in “Assessing Potential Changes in Advance” on page 73, the summary does not mention any deleted tables.

```
proc metalib;
omr  (library="v9SASlib" repname="Meta Proc repos" );
run;
```

Here is the resulting SAS log.

```
85   proc metalib;
86   omr (library="v9SASlib" repname="Meta Proc repos" );
87   run;

NOTE: A total of 1 tables were analyzed for library "v9SASlib".
NOTE: Metadata for 0 tables was updated.
NOTE: Metadata for 1 tables was added.
NOTE: Metadata for 0 tables matched the data sources.
NOTE: 0 other tables were not processed due to error or UPDATE_RULE.
NOTE: PROCEDURE METALIB used (Total process time):
      real time           19.06 seconds
      cpu time            5.39 seconds
```

Changing the Update Rule

By using the optional UPDATE_RULE statement, you can change the default behavior of PROC METALIB. The principal rules that you can specify are shown as follows:

NOADD	specifies not to add table metadata to the metadata repository for physical tables that have no metadata.
NOUPDATE	specifies not to update existing table metadata to resolve discrepancies with the corresponding physical tables.
DELETE	specifies to delete table metadata if a corresponding physical table is not found in the specified library.

Examples: Adding, Updating, and Deleting Metadata

The following example shows how to use PROC METALIB to add metadata for new tables, update table definitions where necessary, and also delete table definitions that are no longer valid. (You can also perform these functions using SAS Data Integration Studio.)

```
proc metalib;
omr (library="sas91 lib2" repname="Meta Proc repos" );
update_rule=(delete);
report;
run;
```

The following example shows how to use UPDATE_RULE with DELETE, NOADD, and NO UPDATE to delete table definitions that are no longer valid, as well as suppress the default add and update actions:

```
proc metalib;
omr  (library="sas91 lib2" repname="Meta Proc repos" );
update_rule (delete noadd noupdate);
report;
run;
```

The resulting SAS output resembles the following sample:

```
                              The METALIB Procedure

                         Summary Report for Library sas91 lib2
                              Repository Meta Proc repos
                                       17MAR2005

                               Metadata Summary Statistics

                           Total tables analyzed          2
                           Tables Updated                 0
                           Tables Added                   0
                           Tables matching data source    0
                           Tables not found               0
```

Specifying Which Tables Are Affected

You can use the optional SELECT or EXCLUDE statements to perform an operation against a subset of the tables in a library. SELECT and EXCLUDE are mutually exclusive, so you should use only one or the other.

When you set the SELECT statement, you can choose the tables for processing:

- For tables, specify their SAS name. If no table definition is found in metadata, it is created in the repository that contains the library object. If a matching table definition is found in metadata, it is compared to the physical table. If differences are found, the table definition is updated in metadata.
- For tables already registered in metadata, specify either the unique metadata identifier or the value in the SASTableName attribute. If you specify the metadata identifier, only the specified table definition is updated, not the first table definition in the association list.

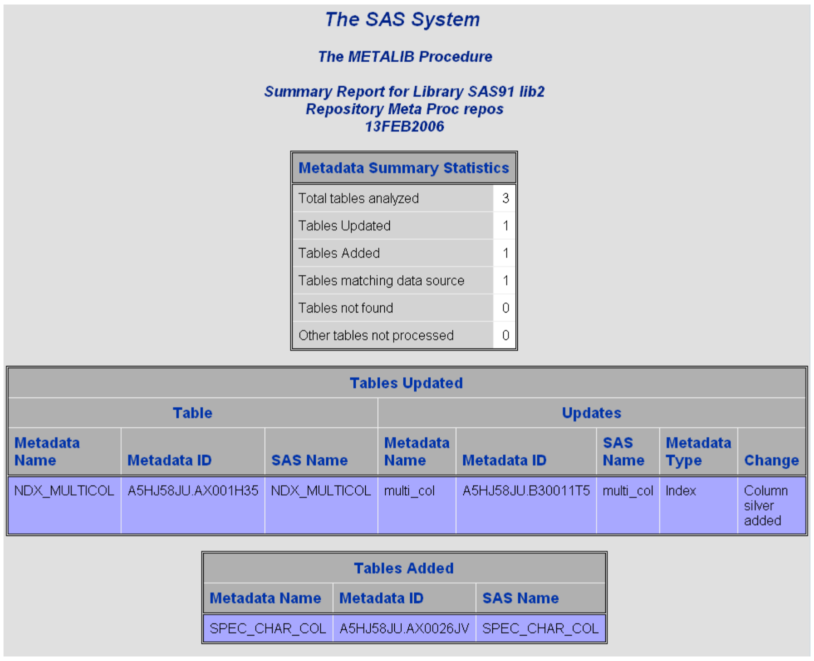
You can use EXCLUDE to specify a single table or a list of tables to exclude from processing.

Examples: Specifying Tables

The following example shows how to use SELECT to process only a subset of tables:

```
ods html "myfile";
proc metalib;
omr (library="SAS91 lib2" repname="Meta Proc repos" );
select(spec_char_col ukeys ndx_multicol);
report;
run;
```

Here is the resulting ODS output:



The SAS System

The METALIB Procedure

Summary Report for Library SAS91 lib2
Repository Meta Proc repos
13FEB2006

Metadata Summary Statistics	
Total tables analyzed	3
Tables Updated	1
Tables Added	1
Tables matching data source	1
Tables not found	0
Other tables not processed	0

Tables Updated

Table			Updates				
Metadata Name	Metadata ID	SAS Name	Metadata Name	Metadata ID	SAS Name	Metadata Type	Change
NDX_MULTICOL	A5HJ58JU.AX001H35	NDX_MULTICOL	multi_col	A5HJ58JU.B30011T5	multi_col	Index	Column silver added

Tables Added

Metadata Name	Metadata ID	SAS Name
SPEC_CHAR_COL	A5HJ58JU.AX0026JV	SPEC_CHAR_COL

The following example shows how to use EXCLUDE to exclude a specific subset of tables:

```
proc metalib;
omr (library="Geography Lib" repname="Foundation");
exclude(country postal mystate2);
noexec;
report;
run;
```

CHAPTER 5 Optimizing Data Storage

Overview of Optimizing Data Storage

For the purposes of querying, cube loading, and creating data marts and data warehouses, all four data storage structures (explained in Chapter 1, "Overview of Common Data Sources," on page 1) can be optimized to improve performance. Some optimization can be achieved, for example, by specifying transformation options in SAS Data Integration Studio. Some optimization requires hardware configuration, as in the case of SPD Engine tables. Cubes can be optimized for querying and loading during the cube loading process. For SAS tables, database tables, and SPD Engine tables, libraries can be defined in the metadata with options that enhance performance.

For more information, see these sections:

- "Compressing Data" on page 80
- "Indexing Data" on page 82
- "Sorting Data" on page 83
- "Buffering Data for Base SAS Tables" on page 85
- "Buffering Data for DB2 (UNIX and PC), ODBC, OLE DB, Oracle, SQL Server, and Sybase Tables" on page 86
- "Using Threaded Reads" on page 87
- "Validating SPD Engine Hardware Configuration" on page 87
- "Grid Computing Data Considerations" on page 93

Compressing Data

Compression is a process that reduces the number of bytes that are required to represent each table row. In a compressed file, each row is a variable-length record. In an uncompressed file, each row is a fixed-length record. Compressed tables contain an internal index that maps each row number to a disk address so that the application can access data by row number. This internal index is transparent to the user. Compressed tables have the same access capabilities as uncompressed tables. Here are some advantages of compressing a file:

- reduced storage requirements for the file
- fewer I/O operations necessary to read from or write to the data during processing

Here are some disadvantages of compressing a file:

- More CPU resources are required to read a compressed file because of the overhead of uncompressing each observation.
- There are situations when the resulting file size might increase rather than decrease.

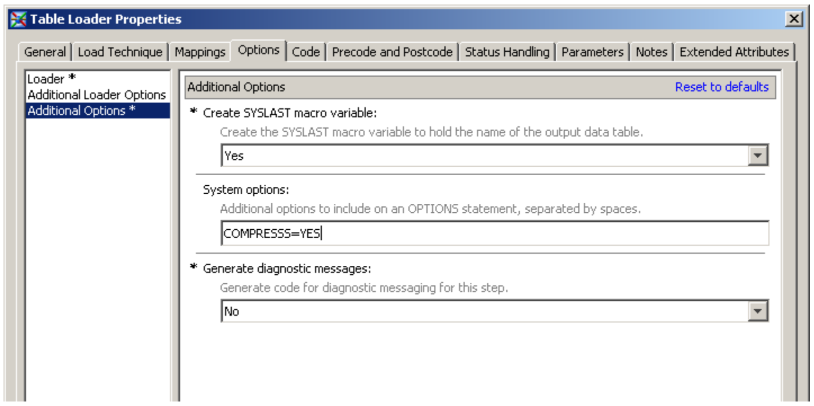
These are the types of compression that you can specify:

- CHAR to use the RLE (Run Length Encoding) compression algorithm, which works best for character data.
- BINARY to use the RDC (Ross Data Compression) algorithm, which is highly effective for compressing medium to large (several hundred bytes or larger) blocks of binary data. (The SPD Engine does not support binary compression.)

You can compress these types of tables:

- all tables that are created during a SAS session. Besides specifying SAS system options on the command line or inside a SAS program with the OPTIONS statement, you can use SAS Data Integration Studio to set system options. For example, you can use the **`System Options`** field to set the COMPRESS= system option on a table loader transformation. (A table loader transformation generates or retrieves code that puts data into a specified target table.)

Figure 5.1 The Options Tab in a Table Loader Properties Dialog Box in SAS Data Integration Studio



Additional Options

* Create SYSLAST macro variable:
Create the SYSLAST macro variable to hold the name of the output data table.
Yes

System options:
Additional options to include on an OPTIONS statement, separated by spaces.
COMPRESSS=YES

* Generate diagnostic messages:
Generate code for diagnostic messaging for this step.
No

- all tables for a particular library. For example, when you register a Base SAS engine library in the metadata, you can specify the COMPRESS= option in the **`Other options to be appended`** field on the **`Options for any host`** tab (see

"Setting LIBNAME Options That Affect Performance of SAS Tables" on page 88). For third-party relational database tables, you can use the **`Options to be appended`** field on the **`Other Options`** tab (see "Setting LIBNAME Options That Affect Performance of SAS/ACCESS Databases" on page 89).

Note: You cannot specify compression for an SPD Engine data library. △

- an individual table. In SAS Data Integration Studio, SAS tables have a **`Compressed`** option that is available from the table properties dialog box. To use CHAR compression, you select **`YES`**. To use BINARY compression, you select **`Binary`**.

Figure 5.2 The Table Options Dialog Box in SAS Data Integration Studio

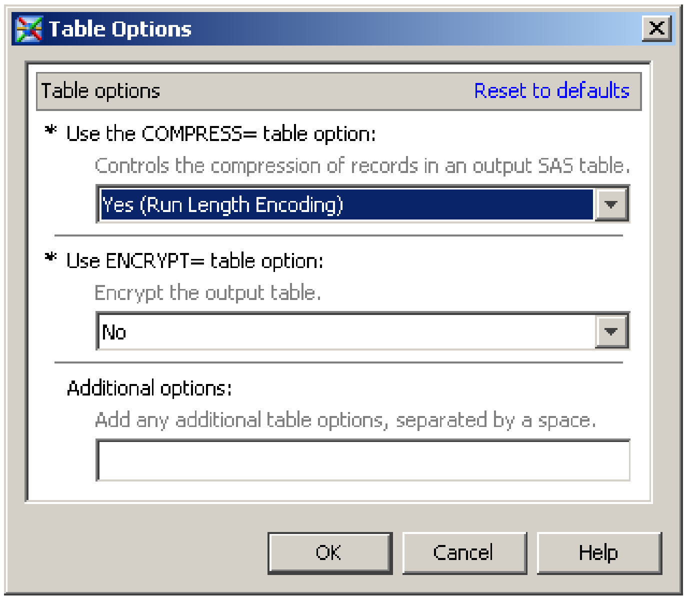


Table Options

Table options — Reset to defaults

* Use the COMPRESS= table option:
Controls the compression of records in an output SAS table.
Yes (Run Length Encoding)

* Use ENCRYPT= table option:
Encrypt the output table.
No

Additional options:
Add any additional table options, separated by a space.

For SPD Engine tables and third-party relational database tables, you can use the **`Table Options`** field in the table properties dialog box to specify the COMPRESS= option.

Note: The SPD Engine compresses the data component (.dpf) file by blocks as the engine is creating the file. (The data component file stores partitions for an SPD Engine table.) To specify the number of observations that you want to store in a compressed block, you use the IOBLOCKSIZE= table option in addition to the COMPRESS= table option. For example, in the **`Table Options`** field in the table properties dialog box, you might enter **`COMPRESS=YES IOBLOCKSIZE=10000`**. The default blocksize is 4096 (4k). △

When you create a compressed table, SAS records in the log the percentage of reduction that is obtained by compressing the file. SAS obtains the compression percentage by comparing the size of the compressed file with the size of an uncompressed file of the same page size and record count. After a file is compressed, the setting is a permanent attribute of the file, which means that to change the setting, you must re-create the file. To uncompress a file, you can, for example, in SAS Data Integration Studio, select **`Default (NO)`** for the **`Compressed`** option in the table properties dialog box for a SAS table.

For more information about compression, see *SAS Language Reference: Dictionary*.

Indexing Data

An index is an optional file that you can create to provide direct access to specific rows. The index stores values in ascending value order for a specific column or columns and includes information about the location of those values within rows in the table. In other words, an index enables you to locate a row by value. For example, if you use SAS to find a specific Social Security number (123-45-6789), SAS performs the search differently depending on whether there is an index on the row that contains the Social Security numbers:

- Without an index, SAS accesses rows sequentially in the order in which they are stored in the table. SAS reads each row, looking for SSN=123-45-6789 until the value is found or all observations are read.
- With an index on column SSN, SAS accesses the row directly. SAS satisfies the condition by using the index and going straight to the row that contains the value. SAS does not have to read each row.

When you create an index, you designate which columns to index. You can create two types of indexes:

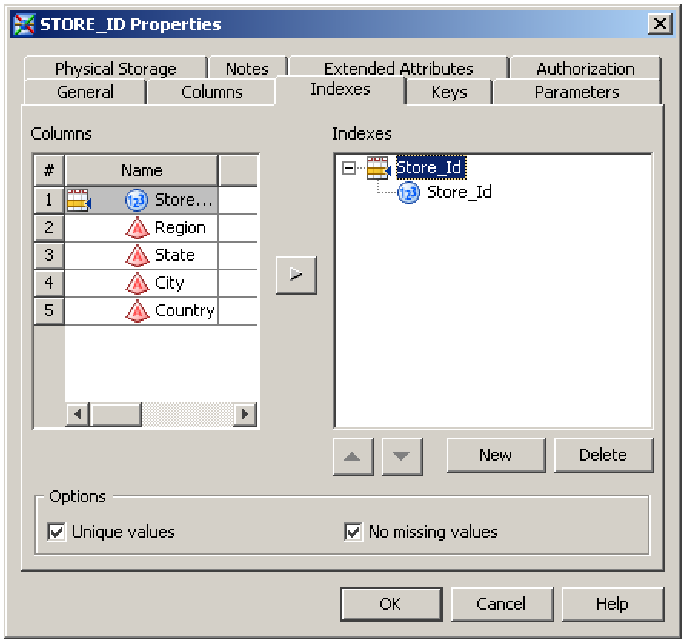
- a simple index, which consists of the values of one column
- a composite index, which consists of the values of more than one column, with the values concatenated to form a single value

For each indexed column, you can also perform these tasks:

- declare unique values. A unique index guarantees that values for one column or the combination of a composite group of columns remain unique for every row in the table. If an update tries to add a duplicate value to that column, then the update is rejected.
- keep missing values from using space in the index by specifying that missing values are not maintained by the index.

In addition to writing SAS code to create indexes, you can create indexes on target tables by using SAS Data Integration Studio. In SAS Data Integration Studio, you use the properties window for the table to index individual columns. When you create the index, you can also specify **`Unique values`** and **`No missing values`**. Note that any indexes registered in metadata for a target table are physically created when the job is run. Simply editing the properties for an existing table and adding indexes does not update the physical table. The following figure shows the SAS Data Integration Studio properties dialog box for a table:

Figure 5.3 The Indexes Tab in the Properties Dialog Box for a Table Named STORE_ID



Columns

#	Name
1	Store...
2	Region
3	State
4	City
5	Country

Indexes

- Store_Id
  - Store_Id

Options

- [✓] Unique values
- [✓] No missing values

In general, SAS can use an index to improve performance in these situations:

- □ For cube loading, a composite index on the columns that make up the cube's hierarchies might provide best results.
- □ For WHERE processing, an index can provide faster and more efficient access to a subset of data. Note that to process a WHERE expression, SAS decides whether to use an index or to read the table sequentially.

  Note: For WHERE processing, the Base SAS engine uses a maximum of one index. The SPD Engine can use multiple indexes. △

Even though an index can reduce the time that is required to locate a set of rows, especially for a large table, there are costs that are associated with creating, storing, and maintaining the index. When deciding whether to create an index, you must consider increased resource usage, along with the performance improvement.

Once an index exists, SAS treats it as part of the table. That is, if you add or delete columns or modify values, the index is automatically updated.

For more information about creating indexes, see *SAS Language Reference: Concepts*.

Sorting Data

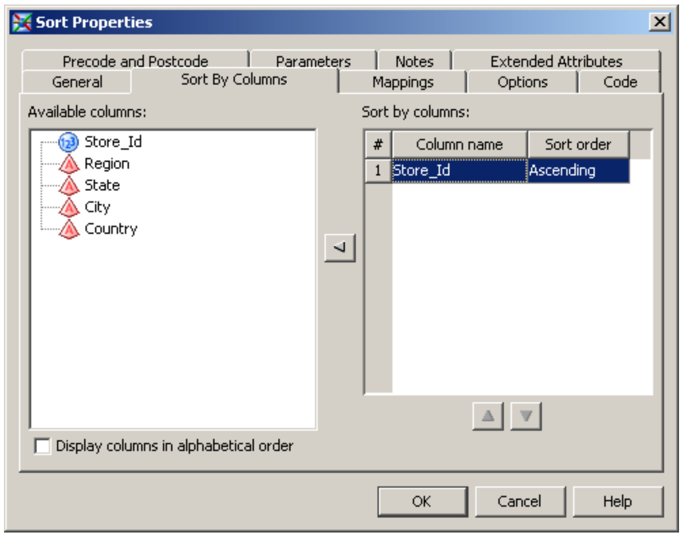
Overview to Sorting Data

You can sort table rows by the values of one or more character or numeric columns. For Base SAS tables and third-party relational database tables, the process either replaces the original table or creates a new table. You can perform sorting in two ways:

- □ using the SAS SORT procedure

- setting properties for a SAS sort template in SAS Data Integration Studio, as shown in the following figure:

Figure 5.4 The Sort By Columns Tab in the Sort Properties Dialog Box



Available columns:

- Store_Id
- Region
- State
- City
- Country

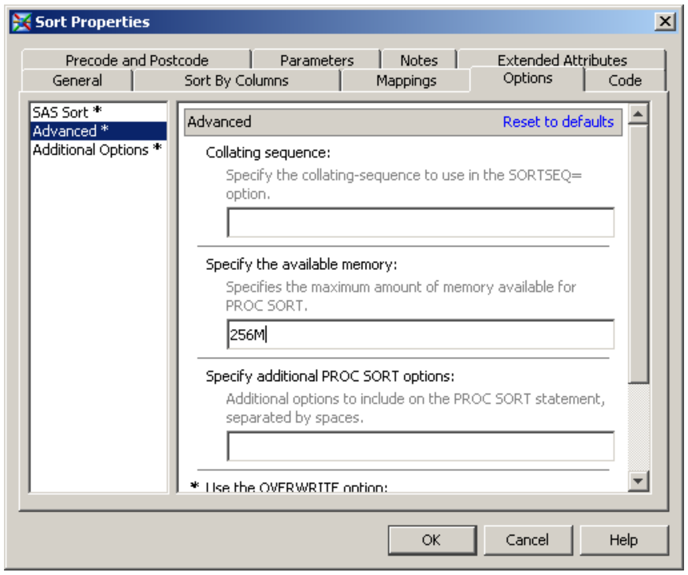
Sort by columns:

#	Column name	Sort order
1	Store_Id	Ascending

☐ Display columns in alphabetical order

To manage the memory that is used for the sorting process, you can specify the maximum amount of memory that is available to the sort. Generally, the sort size should be less than the physical memory available to the process. If the sorting requires more memory than you specify, then SAS creates a temporary utility file on disk. To specify a sort size in SAS Data Integration Studio, access the **`Options`** tab in the properties window for the sort template and enter a value in the **`Sortsize`** field, as shown in the following figure:

Figure 5.5 The Options Tab in the SAS Sort Properties Dialog Box



Collating sequence:
Specify the collating-sequence to use in the SORTSEQ= option.

Specify the available memory:
Specifies the maximum amount of memory available for PROC SORT.
256M

Specify additional PROC SORT options:
Additional options to include on the PROC SORT statement, separated by spaces.

* Use the OVERWRITE option:

The SPD Engine has implicit sorting capabilities, which saves time and resources for SAS applications that process large tables. When the SPD Engine encounters a BY clause, if the data is not already sorted or indexed on the BY column, then the SPD Engine automatically sorts the data without affecting the permanent table or producing a new table. You can change the implicit sorting options when you define an SPD Engine library in the metadata. See “Setting LIBNAME Options That Affect Performance of SPD Engine Tables” on page 91.

For more information about the SORT procedure, see the *Base SAS Procedures Guide*.

Multi-Threaded Sorting

The SAS system option THREADS activates multi-threaded sorting, which achieves a degree of parallelism in the sorting operations. This parallelism is intended to reduce the real time to completion for a given operation; however, the parallelism comes at the possible cost of additional CPU resources. For more information, see "Support for Parallel Processing" in *SAS Language Reference: Concepts*.

The performance of the multi-threaded sort is affected by the value of the SAS system option CPUCOUNT=. CPUCOUNT= indicates how many system CPUs are available for use by the multi-threaded sort. The multi-threaded sort supports concurrent input from the partitions of a partitioned table.

Note: For information about the support of partitioned tables in your operating environment, see the SAS documentation for your operating environment. △

For more information about THREADS and CPUCOUNT=, see the chapter about SAS system options in *SAS Language Reference: Dictionary*.

Sorting a Database Table

When you use a third-party database table, the column ordering that is produced by the SORT procedure depends on whether the DBMS or SAS performs the sorting. If you use the BEST value of the SAS system option SORTPGM=, then either the DBMS or SAS performs the sort. If the DBMS performs the sort, then the configuration and characteristics of the DBMS sorting program affect the resulting data order. Most database management systems do not guarantee sort stability, and the sort might be performed by the database table regardless of the state of the SORTEQUALS or NOSORTEQUALS system options and the EQUALS or NOEQUALS procedure options.

If you set the SAS system option SORTPGM= to SAS, then unordered data is delivered from the DBMS to SAS and SAS performs the sorting. However, consistency in the delivery order of columns from a database table is not guaranteed. Therefore, even though SAS can perform a stable sort on the DBMS data, SAS cannot guarantee that the ordering of columns within output BY groups will be the same, run after run. To achieve consistency in the ordering of columns within BY groups, first populate a SAS table with the database table, then use the EQUALS or SORTEQUALS option to perform a stable sort.

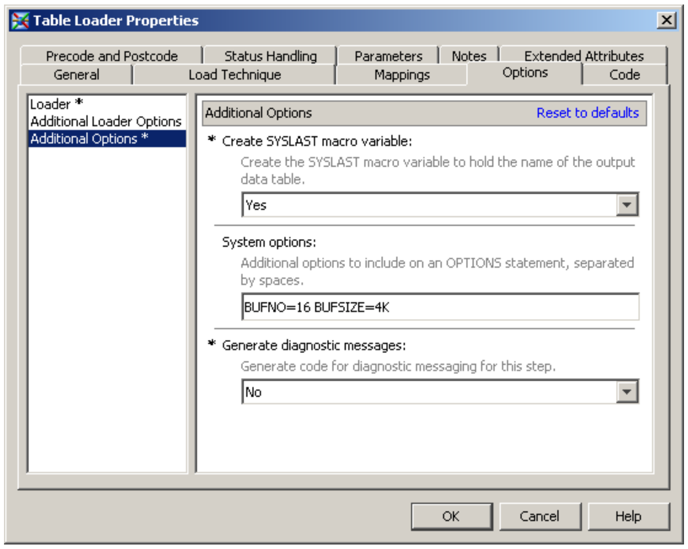
Buffering Data for Base SAS Tables

For Base SAS tables, you might be able to make performance improvements by performing these tasks:

- tuning the size of table pages on disk by using the BUFSIZE= system option. SAS uses the BUFSIZE= option to set the permanent page size for the SAS table. The page size is the amount of data that can be transferred for an I/O operation to one buffer. If you know that the total amount of data is going to be small, you can set a small page size, so that the total table size remains small and you minimize the amount of wasted space on a page. Large tables that are accessed sequentially benefit from larger page sizes because sequential access reduces the number of system calls that are required to read the table.

- adjusting the number of open page buffers when the SAS table is processed. Increasing the value of the BUFNO= option can improve performance by enabling applications to read more data with fewer passes; however, your memory usage increases. You must determine the optimal value for your needs.

Besides specifying SAS system options on the command line or inside a SAS program with the OPTIONS statement, you can set the BUFSIZE= and BUFNO= system options in SAS Data Integration Studio. For example, you can set these **`System Options`** in the properties window for a table loader transformation.



* Create SYSLAST macro variable:
Create the SYSLAST macro variable to hold the name of the output data table.
Yes

System options:
Additional options to include on an OPTIONS statement, separated by spaces.
BUFNO=16 BUFSIZE=4K

* Generate diagnostic messages:
Generate code for diagnostic messaging for this step.
No

For more information about the BUFSIZE= and BUFNO= options, see the *SAS Language Reference: Dictionary* and the documentation for your operating environment.

Note: In addition, the SASFILE statement enables you to store the entire Base SAS table in memory, and the table remains open until you close it because SASFILE caches the data and the open request. For more information about the SASFILE statement, see the *SAS Language Reference: Dictionary*. △

Buffering Data for DB2 (UNIX and PC), ODBC, OLE DB, Oracle, SQL Server, and Sybase Tables

For DB2 (UNIX and PC), ODBC, OLE DB, Oracle, SQL Server, and Sybase, you can adjust page buffers by setting the INSERTBUFF= and READBUFF= options on the library (see “Setting LIBNAME Options That Affect Performance of SAS/ACCESS Databases” on page 89) or on the individual table. The options are described as follows:

- The INSERTBUFF= option specifies the number of rows to insert. SAS allows the maximum that is supported by the DBMS. The optimal value for this option varies with factors such as network type and available memory. You might need to experiment with different values in order to determine the best value for your site.
- The READBUFF= option specifies the number of rows to hold in memory. SAS allows the maximum number that is supported by the DBMS. Buffering data

reads can decrease network activities and increase performance. However, because SAS stores the rows in memory, higher values for READBUFF= use more memory. In addition, if too many rows are selected at once, then the rows that are returned to the SAS application might be out of date. For example, if someone else modifies the rows, you might not see the changes.

For more information about the INSERTBUFF= and READBUFF= options, see *SAS/ACCESS for Relational Databases: Reference*.

Using Threaded Reads

Most SAS/ACCESS interfaces support threaded reads. With a threaded read, the table read time can be reduced by retrieving the result set on multiple connections between SAS and a DBMS. To perform a threaded read, SAS performs these tasks:

1. It creates threads, which are standard operating system tasks that are controlled by SAS, within the SAS session.
2. It establishes a DBMS connection on each thread.
3. It 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.

Threaded reads only increase performance when the DBMS result set is large. Performance is optimal when the partitions are similar in size. In most cases, threaded reads should reduce the elapsed time of the SAS job. However, threaded reads generally increase the workload on the DBMS. For instance, threaded reads for DB2 under z/OS involve a trade-off, generally reducing job elapsed time but increasing DB2 workload and CPU utilization.

Threaded reads are most effective on new, faster computer hardware running SAS, and with a powerful parallel edition of the DBMS. For example, if SAS runs on a fast uniprocessor or on a multiprocessor machine and your DBMS runs on a high-end SMP server, you will receive substantial performance gains.

For information about how to turn the threaded read function on or off for a DBMS library, see “Setting LIBNAME Options That Affect Performance of SAS/ACCESS Databases” on page 89.

For information about threaded reads, see *SAS/ACCESS for Relational Databases: Reference*.

Validating SPD Engine Hardware Configuration

The SPD Engine automatically determines the optimal process to use to evaluate observations for qualifying criteria specified in a WHERE statement. WHERE statement efficiency depends on such factors as whether the columns in the expression are indexed. A SAS configuration validation program that measures I/O scalability with respect to WHERE processing can help you determine whether your system is properly configured for performing WHERE processing with the SPD Engine. The program performs these tasks:

1. It creates a table with two numeric columns.
2. It repeatedly reads the entire table, each time doubling the number of threads used until the maximum number is reached. The maximum number of threads is

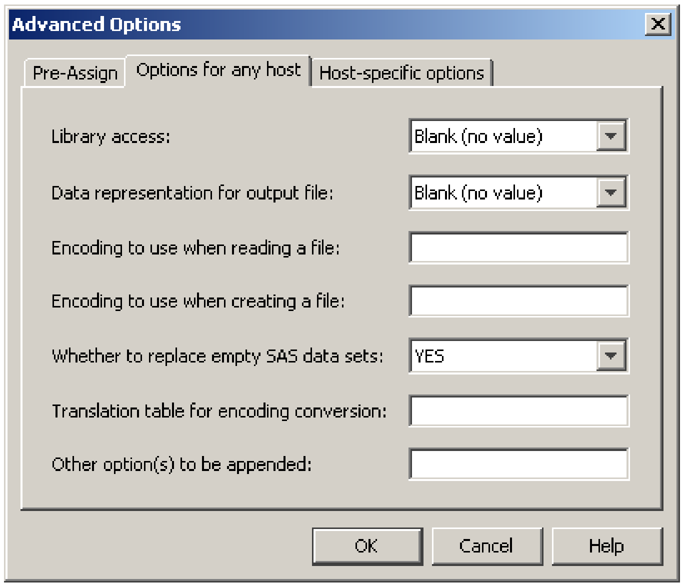
determined by the CPUCOUNT= SAS system option and is specified when SAS is started.

The resulting log file shows timing statistics for each cycle. You can examine this information to determine whether your system is configured correctly. The program is available at **`http://support.sas.com/rnd/scalability/spde/valid.html`**.

Setting LIBNAME Options That Affect Performance of SAS Tables

You can set LIBNAME options that might affect performance of the Base SAS engine. You set these options when you use the New Library wizard to register a Base SAS engine library in the metadata repository. The LIBNAME options are available on the **`Options for any host`** tab and the **`Host-specific options`** tab in the Advanced Options dialog box. To access the Advanced Options dialog box, click the **`Advanced Options`** button on the Library Options window of the New Library wizard.

Figure 5.6 The Options for Any Host Tab in the Advanced Options Dialog Box for a Base SAS Library



Library access: Blank (no value)
Data representation for output file: Blank (no value)
Encoding to use when reading a file:
Encoding to use when creating a file:
Whether to replace empty SAS data sets: YES
Translation table for encoding conversion:
Other option(s) to be appended:

Here are some examples of options that might affect performance:

Data representation for the output file (OUTREP=)
: For all operating environments, you can specify the data representation for the output file. Specifying this option enables you to create files within the native environment by using a foreign environment data representation. For example, an administrator who works in a z/OS operating environment might want to create a file on an HFS system so that the file can be processed in an HP UNIX environment. Specifying HP_UX_64 as the value for this option forces the data representation to match the data representation of the UNIX operating environment that will process the file. This method of creating the file can enhance system performance because the file does not require data conversion when being read by an HP UNIX machine.

Input/output block size (BLKSIZE=)	For Windows, UNIX, and z/OS environments, you can specify the number of bytes that are physically read during an I/O operation. The default is 8 kilobytes, and the maximum value is 1 megabyte.
Number of page caches to use for each open member (CACHENUM=)	For VMS, you can specify the number of page caches to use during I/O operations. The number of caches can potentially reduce the number of I/Os that are required to access the data. You can also set the size of each cache (CACHESIZE= option).

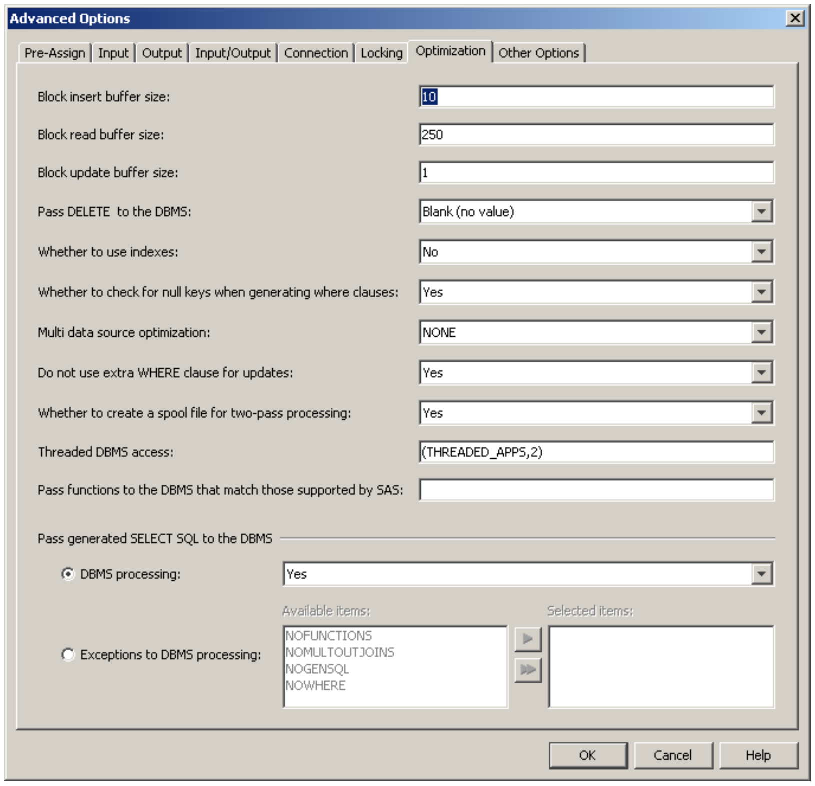
The **`Other option(s) to be appended`** field can be used to specify LIBNAME options such as COMPRESS= (see "Compressing Data" on page 80).

For information about each of the LIBNAME options in the Advanced Options dialog box, click the **`Help`** button.

Setting LIBNAME Options That Affect Performance of SAS/ACCESS Databases

The following LIBNAME options can be used to tune performance of the SAS/ACCESS engines. You can set these options when you use the New Library wizard to register the database libraries in the metadata repository. To access the Advanced Options dialog box, click the **`Advanced Options`** button on the Library Options window of the New Library wizard.

Figure 5.7 The Optimization Tab in the Advanced Options Dialog Box for a DB2 Library for UNIX and PC


Advanced Options
Pre-Assign | Input | Output | Input/Output | Connection | Locking | Optimization | Other Options
OK
Cancel
Help

Block insert buffer size:	10
Block read buffer size:	250
Block update buffer size:	1
Pass DELETE to the DBMS:	Blank (no value)
Whether to use indexes:	No
Whether to check for null keys when generating where clauses:	Yes
Multi data source optimization:	NONE
Do not use extra WHERE clause for updates:	Yes
Whether to create a spool file for two-pass processing:	Yes
Threaded DBMS access:	(THREADED_APPS,2)
Pass functions to the DBMS that match those supported by SAS:	

Pass generated SELECT SQL to the DBMS

[✓ DBMS processing:] Yes

Exceptions to DBMS processing:

Available items: NOFUNCTIONS, NOMULTOUTJOINS, NOGENSQL, NOWHERE

Selected items:

The tabs that are available in the Advanced Options dialog box, as well as the options on each of the tabs, vary between database management systems. The following list provides a description of the options on **`Optimization`** tab for DB2 libraries for UNIX and PC:

Block insert buffer size (INSERTBUFF=)	specifies the number of rows in a single insert operation. See "Buffering Data for DB2 (UNIX and PC), ODBC, OLE DB, Oracle, SQL Server, and Sybase Tables" on page 86.
Block read buffer size (READBUFF=)	specifies the number of rows of DBMS data to read into the buffer. See "Buffering Data for DB2 (UNIX and PC), ODBC, OLE DB, Oracle, SQL Server, and Sybase Tables" on page 86.
Pass functions to the DBMS that match those supported by SAS (SQL_FUNCTIONS=)	when set to ALL, specifies that functions that match functions supported by SAS should be passed to the DBMS. The functions that are passed are: DATE, DATEPART, DATETIME, TIME, TIMEPART, TODAY, QRT, COMPRESS, SUBSTR, DAY, SECOND, INDEX, TRANWRD, HOUR, WEEKDAY, LENGTH, TRIMN, MINUTE, YEAR, REPEAT, MOD, MONTH, BYTE, and SOUNDEX. Use of this option can cause unexpected results, especially if used for NULL processing and date, time, and timestamp handling. Exercise care when using this option.
Pass DELETE to the DBMS (DIRECT_EXE=)	specifies that an SQL delete statement is passed directly to the DBMS for processing. Selecting this option improves performance because SAS does not have to read the entire result set and delete one row at a time.
Whether to use indexes (DBINDEX=)	specifies whether SAS uses indexes that are defined on DBMS columns to process a join. Valid values are YES or NO. For more information about indexes, see "Indexing Data" on page 82.
Whether to check for null keys when generating WHERE clauses (DBNULLKEYS=)	specifies whether the WHERE clause should detect NULL values in columns. Valid values are YES or NO. YES is the default for most interfaces and enables SAS to prepare the statement once and use it for any value (NULL or NOT NULL) in the column.
Multi data source optimization (MULTI_DATASRC_OPT=)	when processing a join between two tables, specifies whether an IN clause should be created to optimize the join. Valid values are NONE and IN_CLAUSE. IN_CLAUSE specifies that an IN clause containing the values read from a smaller table will be used to retrieve the matching values in a larger table based on a key column designated in an equi-join. When processing a join between a SAS table and a DBMS table, the SAS table should be smaller than the DBMS table for optimal performance.
Whether to create a spool file for two-pass processing (SPOOL=)	specifies whether to create a utility spool file during transactions that read data more than once. In some cases, SAS processes data in more than one pass through the same set of rows. Spooling is the process of writing rows that have been retrieved during the first pass of a data read to a spool file. In the second pass, rows can be re-read without performing I/O to the DBMS a second time. In cases where the data needs to be read more than once, spooling improves performance. Spooling also guarantees that the data remains the same between passes. Valid values are YES or NO.
Threaded DBMS access (DBSLICEPARM=)	specifies the scope of DBMS threaded reads and the number of threads. If this option is set to the default, then PROC SQL will not

use threading to read, for example, data for a Web report. To force a specified number of threads for a threaded read from the DBMS server, change the default to (ALL,*number-of-threads*).

Note: If PROC SQL attempts implicit pass-through, then threading will be disabled, regardless of the **`Threaded DBMS access`** setting. To disable implicit pass-through, set the **`Pass generated SELECT SQL to the DBMS - DBMS`** processing option to **`NO`**. △

For more information about threaded reads, see "Using Threaded Reads" on page 87.

Pass generated SELECT SQL to the DBMS - DBMS processing (DIRECT_SQL=)	specifies whether generated SQL is passed to the DBMS for processing. Valid values are YES or NO.
Pass generated SELECT SQL to the DBMS - exceptions to DBMS processing (DIRECT_SQL=)	if the value for the previous option is YES, then this option specifies *how* generated SQL is passed to the DBMS for processing. For example, NOWHERE prevents WHERE clauses from being passed to the DBMS for processing.

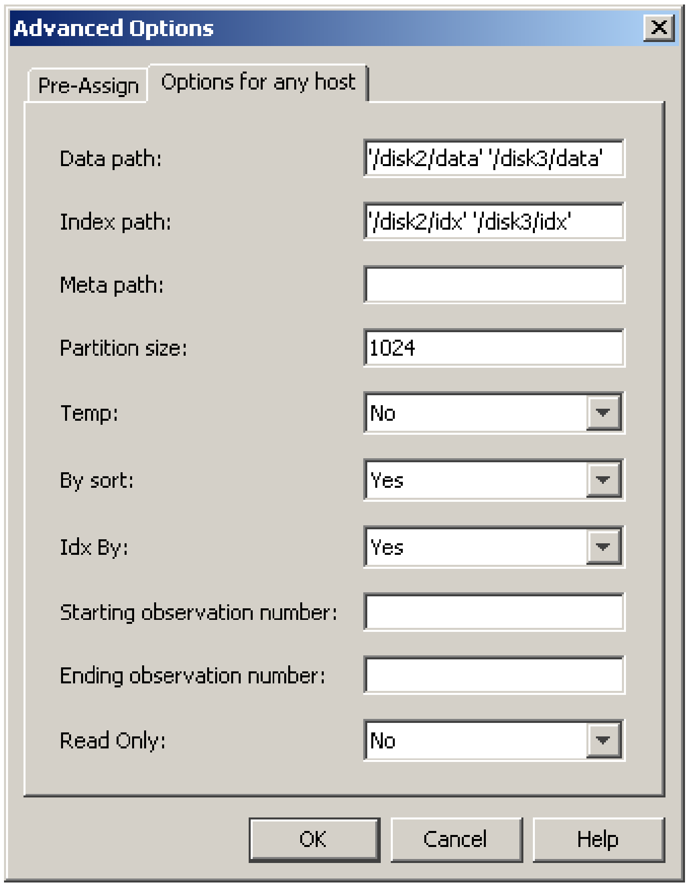
The **`Other Options`** tab, which is available for all database management systems, can be used to specify LIBNAME options such as COMPRESS= (see "Compressing Data" on page 80).

For information about each of the LIBNAME options in the Advanced Options dialog box, click the **`Help`** button. For information about all SAS/ACCESS LIBNAME options, see *SAS/ACCESS for Relational Databases: Reference*.

Setting LIBNAME Options That Affect Performance of SPD Engine Tables

The following LIBNAME options can be used to tune performance of the SPD Engine. You can set these options when you use the New Library wizard to register an SPD Engine library in the metadata repository. The LIBNAME options are available on the **`Options for any host`** tab in the Advanced Options dialog box. To access the Advanced Options dialog box, click the **`Advanced Options`** button on the Library Options window of the New Library wizard. The **`Advanced Options`** dialog box is shown in the following figure:

Figure 5.8 The Options for Any Host Tab in the Advanced Options Dialog Box for an SPD Engine Library



Data path: '/disk2/data' '/disk3/data'

Index path: '/disk2/idx' '/disk3/idx'

Meta path:

Partition size: 1024

Temp: No

By sort: Yes

Idx By: Yes

Starting observation number:

Ending observation number:

Read Only: No

Data path (DATAPATH=)	specifies a list of paths in which to store partitions (.dpf files) for an SPD Engine table. The engine creates as many partitions as are needed to store all the data. The size of the partitions is set using the PARTSIZE= option. Partitions are created in the specified paths in a cyclic fashion. The data path area is best configured as multiple paths. Allot one I/O controller per data path to provide high I/O throughput, which is the rate at which requests for work are serviced by a computer system. The data path area is best configured for redundancy (RAID 1).
Index path (INDEXPATH=)	specifies a path or a list of paths in which to store the two index component files (.hbx and .idx) that are associated with an SPD Engine table. Additional specified paths accept the overflow from the immediately preceding path. The index path area is best configured as multiple paths. Use a volume manager file system that is striped across multiple disks (RAID 0) to enable adequate index performance, both when evaluating WHERE clauses and creating indexes in parallel. Redundancy (RAID 5 or RAID 10) is also recommended.
Meta path (METAPATH=)	specifies a list of overflow paths in which to store metadata component (.mdf) files for an SPD Engine table. The metadata component file for each table must begin in the primary path. When that primary path is full, the overflow is sent to the specified

	METAPATH= location. The metadata path area is best configured for redundancy (RAID 1) so that metadata about the data and its indexes is not lost.
Partition size (PARTSIZE=)	specifies the size (in megabytes) of the data component partitions when an SPD Engine table is created. By splitting the data portion of an SPD Engine table at fixed-size intervals, you can gain a high degree of scalability for some operations. For example, the SPD Engine can spawn threads in parallel, up to one thread per partition for WHERE evaluations.
Temp(TEMP=)	specifies whether to create a temporary subdirectory of the directory specified in the Path field on the Library Properties wizard window. The directory is used to temporarily store the metadata component files associated with table creation. It is deleted at the end of the SAS session.
By sort (BYSORT=)	specifies that the SPD Engine should perform an automatic implicit sort when it finds a BY statement for processing data in the library (unless the data is indexed on the BY column). Valid values are YES (perform the sort) and NO (do not perform the sort). The default is YES.
Starting observation number (STARTOBS=)	specifies the number of the starting observation in a user-defined range of observations that are qualified with a WHERE expression. By default the SPD Engine processes all observations in the table.
Ending observation number (ENDOBS=)	specifies the number of the ending observation in a user-defined range of observations that are qualified with a WHERE expression. By default the SPD Engine processes all observations in the table.

In addition to the LIBNAME options, there are also table and system options that can be used to tune SPD Engine performance. For example, the SPDEUTILLOC= system option allots space for temporary files that are generated during SPD Engine operations. This area is best configured as multiple paths. Use a volume manager file system that is striped across multiple disks (RAID 0) to reduce out-of-space conditions and improve performance. Redundancy (RAID 5 or RAID 10) is also recommended because losing the work area could stop the SPD Engine from functioning.

The *SAS Scalable Performance Data Engine: Reference* includes a "Quick Guide to the SPD Engine Disk-I/O Set-Up" that helps you to do the following:

- determine the amount of space that needs to be allocated to the data, metadata, index, and work areas
- evaluate the advantages and disadvantages of different RAID groups for each of the different types of areas

For more information about table and other system options for the SPD Engine, see **`http://support.sas.com/rnd/scalability/spde/syntax.html`**. For more information about each of the LIBNAME options in the Advanced Options dialog box, click the **`Help`** button.

Grid Computing Data Considerations

Grid computing has become an important technology for organizations that:

- have long-running applications that can benefit from parallel execution

- want to leverage existing IT infrastructure to optimize computing resources and manage data and computing workloads

The function of a grid is to distribute tasks. Each of the tasks that are distributed across the grid must have access to all the required input data. Computing tasks that require substantial data movement generally do not perform well in a grid. To achieve the highest efficiency, the nodes should spend the majority of the time computing rather than communicating. With grid computing using SAS Grid Manager, the speed at which the grid operates is related more to the storage of the input data than to the size of the data.

Data must either be distributed to the nodes before running the application or—much more commonly—made available through shared network libraries. Storage on local nodes is discouraged. The data storage must scale to maintain high performance while serving concurrent data requests.

The parallel data load is monitored throughout.

Application Response Monitoring

SAS implements the Application Response Monitoring 4.0 (ARM) specification. SAS offers macros, system options, and LOG4SAS as an ARM agent for collecting application availability, performance, usage, and transaction response time. For more information about the ARM implementation, see the *SAS Interface to Application Response Measurement (ARM): Reference*.

SAS Data Integration Studio can report the following measures for jobs:

- number of records processed
- duration of step in the job
- I/O statistics

To view the metrics within SAS Data Integration Studio, right-click the diagram background for the job and select **Collect Runtime Statistics**. When the job is run, view the **Statistics** tab in the Details area of the window.

CHAPTER 6

Managing OLAP Cube Data

Introduction to Managing OLAP Cube Data

Online Analytical Processing (OLAP) is a technology that is used to create decision support software. OLAP enables application users to quickly analyze information that has been summarized into multidimensional views and hierarchies. By summarizing predicted queries into multidimensional views before run time, OLAP tools provide the benefit of increased performance over traditional database access tools. Most of the resource-intensive calculation that is required to summarize the data is done before a query is submitted. One of the advantages of OLAP is how data and its relationships are stored and accessed. OLAP systems house data in structures that are readily available for detailed queries and analytics.

Data Storage and Access

Organizations usually have databases and data stores that maintain repeated and frequent business transaction data. This provides simple yet detailed storage and retrieval of specific data events. However, these data storage systems are not well suited for analytical summaries and queries that are typically generated by decision makers. For decision makers to reveal hidden trends, inconsistencies, and risks in a business, they must be able to maintain a certain degree of momentum when querying the data. An answer to one question usually leads to additional questions and review of the data. Simple data stores do not generally suffice.

The data warehouse is a structure better suited for this type of querying. In a data warehouse, data is maintained and organized so that complicated queries and summaries can be run. OLAP further organizes and summarizes specific categories and subsets of data from the data warehouse. One particular kind of data structure derived from a data warehouse is the *cube*. A cube is a set of data that is organized and structured in a hierarchical, multidimensional arrangement. Such an arrangement results in a robust and detailed level of data storage with efficient and fast query returns. Stored, precalculated summarizations called *aggregations* can be added to the cube to improve cube access performance.

Exporting and Importing Cubes

Cubes are exported and imported as part of a SAS package. SAS Management Console is one of the user interfaces that can perform the import and export of packages. For more information about creating SAS packages, see "Overview of the Promotion Tools" in the *SAS Intelligence Platform: System Administration Guide*.

The data administrator impact of exporting and importing cubes is that when cubes are imported, the tables used in the cube must be available and that building the aggregations for the cube is computationally intensive. The following list highlights some best practices:

- It is impractical to package the detail tables and summary data for large cubes. Do not export them in the package.
- If the cube is being imported to a new metadata server (as opposed to being moved to a new folder on the same metadata server) then make sure the same data sources for the detail tables used by the original cube are available and registered. Accomplish this by registering the same data servers and data libraries on the destination metadata server. If the cube uses a drill-through table, ensure that the library is pre-assigned.
- When importing the cubes, be prepared to associate the cube with an OLAP schema. Consider that the OLAP schema determines the group of cubes that an OLAP server can access.
- Once the cube is imported, the cube and its job are registered in metadata with relationships to an OLAP schema, tables, and folders. By default, the aggregations for the cube must be built after the cube is imported. Due to the computational intensity, consider rebuilding the cube during a period of low activity.

About OLAP Schemas

OLAP schemas provide an organizational function. An OLAP schema is a list of cubes that are grouped together so that they can be accessed by one or more SAS OLAP Servers. Each cube is listed in one and only one OLAP schema. Each SAS OLAP Server is required to use one OLAP schema. Multiple servers can use the same schema.To assign cubes to specific servers you create new OLAP schemas. This might be necessary if you have multiple large cubes. In that case you might want to assign one cube to one host, to one SAS OLAP Server, and to one OLAP schema.

New OLAP schemas are created with the Create OLAP Schema wizard in SAS OLAP Cube Studio or SAS Management Console. SAS OLAP Servers are assigned to new OLAP schemas by changing server properties in SAS Management Console. To create a new OLAP schema or assign an OLAP schema to a SAS OLAP Server using SAS Management Console, see "Create or Assign an OLAP Schema" on page 97. A SAS OLAP Server reads its assigned OLAP schema from metadata only as the server starts. Assigning a new OLAP schema to a server requires that you restart the SAS OLAP Server.

When building, updating, or deleting cubes, you can specify OLAP schemas in the Cube Designer wizard of SAS OLAP Cube Studio. Alternatively, if you choose to write SAS code for PROC OLAP, the schema is specified in the OLAP_SCHEMA= option of the METASVR statement.

Create or Assign an OLAP Schema

To create a new OLAP schema or assign an OLAP schema to a SAS OLAP Server, perform the following steps:

1. Open SAS Management Console.
2. In the left pane, expand **`Server Manager`**.
3. Under **`Server Manager`**, locate the SAS Application Server that contains the SAS OLAP Server. The name of one such SAS Application Server might be **`SASApp`**, for example.
4. Right-click the top-level SAS Application Server and select **`Properties`**.
5. In the Properties window, click the **`OLAP Schema`** tab.
6. Click **`New`** to create a new OLAP schema, or select the down arrow to choose an existing OLAP schema.
7. Click **`OK`** to save changes and close the Properties window.
8. Restart the SAS OLAP Server using the SAS OLAP Server Monitor.

Building a Cube

Overview of Building a Cube

The following is a summary of the cube-building process. For additional information about building and modifying SAS OLAP cubes, see the *SAS OLAP Server: User's Guide*.

Before building a cube, you should collect and scrub your data in addition to planning a dimensional design. When you define the cube, you define the dimensions and measures for the cube along with information about how aggregations should be created and stored. There are two methods of creating a cube:

- You can submit PROC OLAP code by using either the SAS Program Editor or a batch job. If you use PROC OLAP, the cube is created, and then the cube definition is stored in a metadata repository. This is referred to as the *long* form of PROC OLAP.
- You can use the Cube Designer interface in SAS OLAP Cube Studio to define and create the cube. The Cube Designer first stores the cube definition in a metadata repository, and then submits a shorter form of PROC OLAP code to create the cube. This is referred to as the *short* form of PROC OLAP.

Note: The Cube Designer can also be launched from SAS Data Integration Studio. △

Preparations for Building a Cube

To build a cube by using either PROC OLAP or SAS OLAP Cube Studio, you must perform several preliminary tasks:

- □ Configure a metadata server.
- □ Define an OLAP server in the metadata. The server does not need to be running to create cubes, but it must be defined in the metadata.
- □ Analyze the data to determine the location of the table or tables that will be used to build your cubes and what dimensions and measures will be created.
- □ Register the table or tables that will be used to create the cube in the metadata. You do this by using SAS Data Integration Studio or by using SAS OLAP Cube Studio and SAS Management Console as follows:
  - □ Use SAS Management Console to register the metadata for the server that will access the tables. This is a SAS Application Server with a workspace server component.
  - □ Use SAS Management Console to register metadata for the library that contains the table.
  - □ In SAS OLAP Cube Studio, specify the server that will access the tables. To set the server, select **Tools ▶ Options**. Or, if the shortcut bar is displayed, select **`Options`** to set the server.
  - □ In SAS OLAP Cube Studio, select **Source Designer** to load the table definitions (or other information source) as follows:
    - □ From the shortcut bar, select **Tools ▶ Source Designer** or select **Source Designer**.
    - □ Select a Source Type (SAS, ODBC, and so on), and then select **`Next`**.
    - □ If you have not specified a server, or if the server that is specified is not valid, then you will be prompted again for a server.
    - □ Select the library that contains the tables that you want to use, and then select **`Next`**.
    - □ Select the tables to register and then select **`Next`**.
    - □ Select **`Finish`**. The table definitions are register in metadata.
  - □ If you start to create a cube and do not see the table that you need to continue, then you can click the **`Define Table`** button in any of the windows that prompt for tables.
- □ In the Finish window of the cube designer, you are given the option to create the physical cube. The metadata definition is always stored as you leave the Finish window. However, you can defer creation of the physical cube because it might be a resource and time intensive process. If you choose to create the cube as you leave the Finish window, then you must have a SAS Workspace Server defined that you can submit PROC OLAP code to. This server is defined in SAS Management Console.

For more information about the different data types that you can use to load cubes from, see “Loading Cubes” in the *SAS OLAP Server: User’s Guide*.

Note: The SAS Metadata Server enables duplicate librefs to be defined in the metadata. To ensure that the correct library definition is found on the metadata server, you should assign the libref by using the LIBNAME statement for the metadata engine before submitting the PROC OLAP code. Otherwise, PROC OLAP will select the first library definition that it finds with your specified libref, and it will associate your cube metadata with that definition. The selected library definition might or might not contain a description of the data that was actually used to build your cube. For more information about using the LIBNAME statement for the metadata engine, see "Statements" in *SAS Language Reference: Dictionary*. △

When a SAS OLAP cube is created, a directory for that cube is also created. This directory is assigned the same name as the cube, but in uppercase letters. For example, if you save a cube in **`c:\olapcubes`** and name the cube Campaigns, the cube is saved in the directory **`c:\olapcubes\CAMPAIGNS`**.

Storage Location Requirements for Cube Metadata and Related Objects

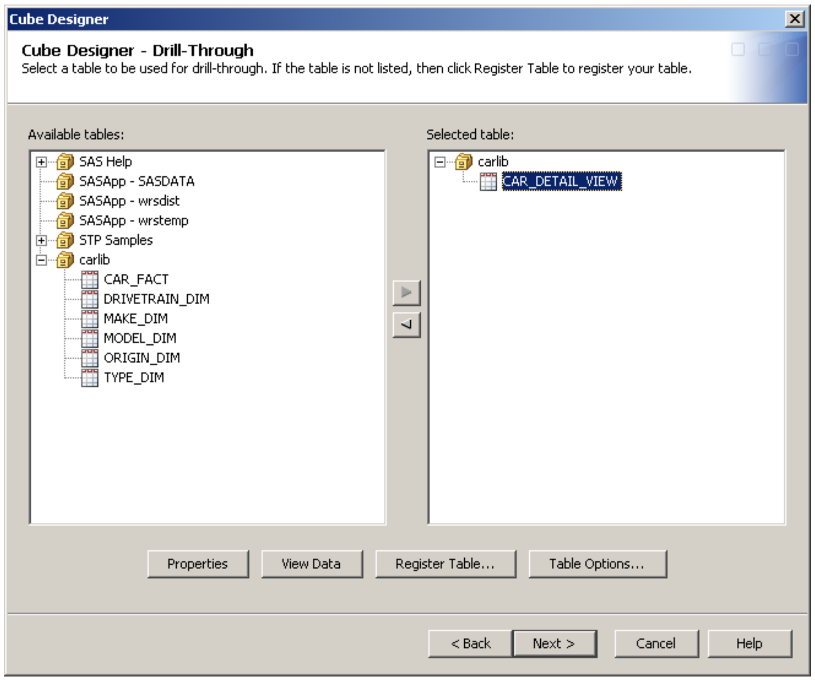
When storing metadata that describes a cube, the metadata objects that describe the cube and the cube's associated libraries and source tables must be stored in the same repository, or the metadata that describes the cube must be in a custom repository that is dependent on the repository that contains the library and table objects. Otherwise, you will not be able to create the cube. In addition, the library and table objects that are referenced by a cube must always be in the same repository. The following options illustrate these conditions:

- The library, table, and cube objects can be in a Foundation repository.
- The library, table, and cube objects can be in Project A, which is dependent on the Foundation repository.
- The library and table objects can be in the Foundation repository, and the cube object can be in Project A.
- The cube object cannot be in the Foundation repository, and the library and table objects cannot be in Project A.
- The table object cannot be in the Foundation repository, and the library and cube objects cannot be in Project A.
- The library object cannot be in the Foundation repository, and the table and cube objects cannot be in Project A.

Making Detail Data Available to a Cube for Drill-Through

You can drill through an OLAP report to the underlying detail data only after you make the detail data available to the cube. You can use either SAS OLAP Cube Studio or the OLAP procedure to make detail data available to the cube:

- In SAS OLAP Cube Studio, you can specify a table for drill-through when you create or edit the cube using the Cube Designer wizard. On the Drill-Through page of the wizard, either select a table and click the right-arrow and then **`Next`** to specify the drill-through table, or just click **`Next`** if drill-through is not needed. The following figure shows the **`Cube Designer - Drill Through`** page of the Cube Designer wizard:



For more information about the Cube Designer wizard, see the SAS OLAP Cube Studio Help. Note that for star schema tables, a view that fully joins the fact and dimension tables is the drill-through table.

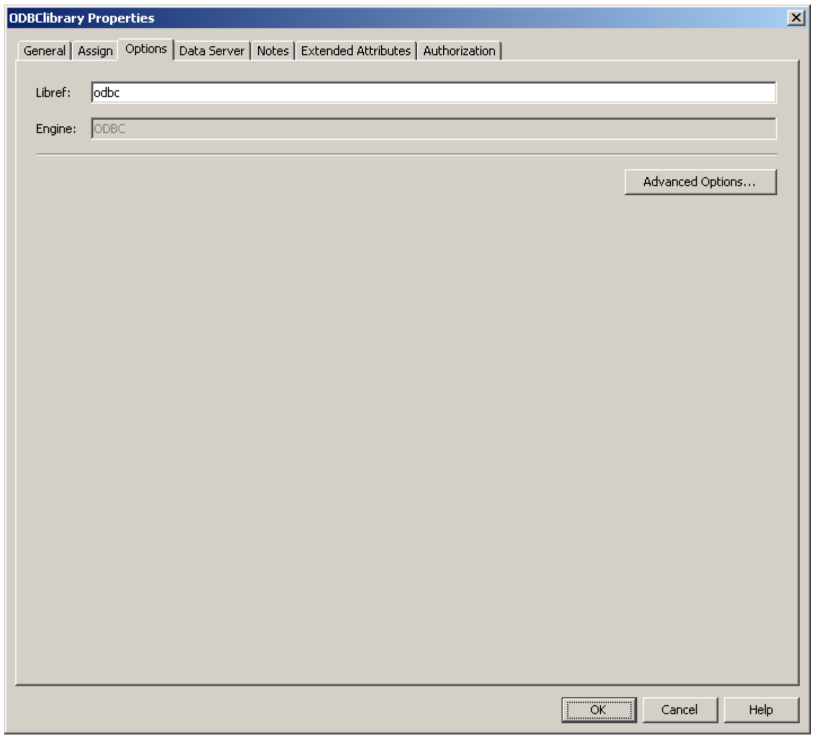
- □ In the PROC OLAP statement, use the DRILLTHROUGH_TABLE option to specify the name of the drill-through table to use. For more information about the DRILLTHROUGH_TABLE option, see "PROC OLAP Statement" in the *SAS OLAP Server: User's Guide*.

Making Detail Data Available to an OLAP Server for Drill-Through

You can drill through an OLAP report to the underlying detail data only after you make the detail data available to the OLAP Server. In order for the OLAP server to make detail data available for a cube, the library for the table that contains the detail data must be registered so that the OLAP server can identify the library to use, and that the library permissions allow ReadMetadata. The simplest way to register the library to the server is to pre-assign it in the metadata repository and store the library in a folder that grants ReadMetadata permission to PUBLIC.

To specify a library as pre-assigned for an OLAP server, perform the following steps:

1 In Data Library Manager (in SAS Management Console), find the **`Libraries`** folder and perform one of the following tasks to get to the dialog box that lets you select advanced options:
   - □ For a new library, right-click the **`Libraries`** folder and select **`New Library`** to start the New Library wizard. Then navigate to the page that enables you to specify the libref.
   - □ For an existing library, open the **`Libraries`** folder and right-click the desired library. Select **`Properties`** from the drop-down menu, and then select the **`Options`** tab in the properties dialog box.

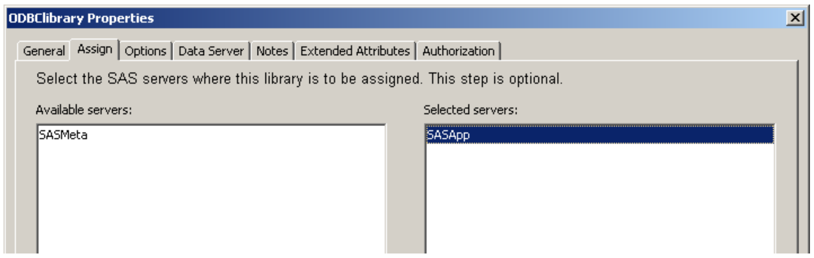


Libref: odbc
Engine: ODBC

2 Click **`Advanced Options`**.

3 Select the **`Library is pre-assigned`** check box on the **`Pre-Assign`** tab in the Advanced Options dialog box.

4 On the **`Assign`** tab of the properties dialog box or the server selection page of the New Library wizard, ensure that the selected application server is the server container that contains your OLAP server.



Select the SAS servers where this library is to be assigned. This step is optional.

Available servers: SASMeta

Selected servers: SASApp

5 Click **`OK`** in the properties dialog box, or finish entering information in the wizard.

6 Restart the OLAP server.

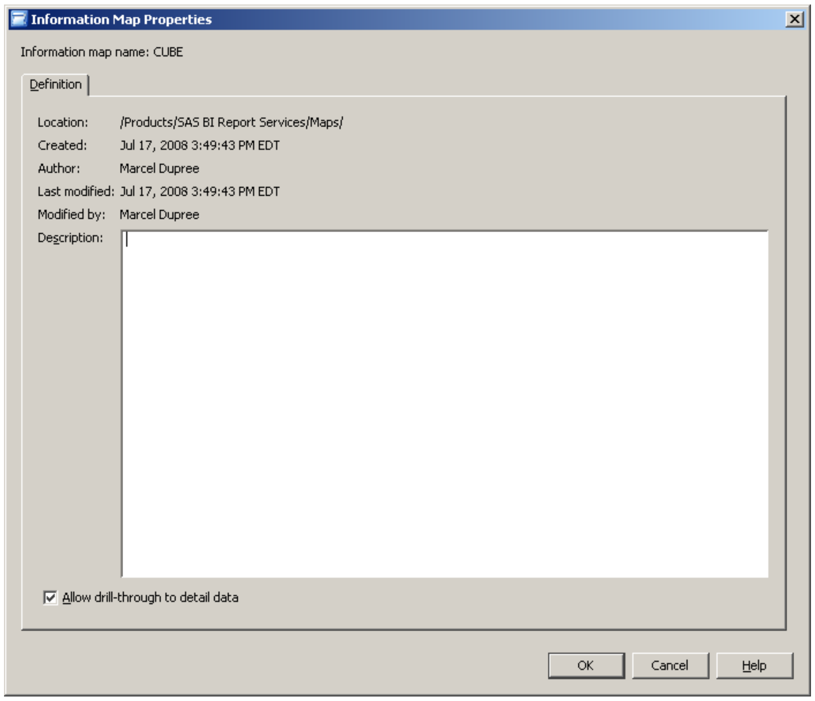
The selected library is assigned after the selected OLAP server starts. After the OLAP server starts, ensure that the library is pre-assigned to the correct SAS OLAP server. The OLAP server also generates a record in the log file stored at *SAS-config-dir*`\Lev1\`*SASApp*`\OLAPServer\Logs\`. The following example shows how pre-assigned libraries are identified in the log file:

```
2008-08-04T13:00:13,068 WARN  [00000010] :SYSTEM@host - NOTE: Libref odbc successfully assigned from logical server.

2008-08-04T13:00:13,068 WARN  [00000010] :SYSTEM@host - NOTE: Libref wrstemp successfully assigned from logical server.

2008-08-04T13:00:13,068 WARN  [00000010] :SYSTEM@host - NOTE: Libref wrsdist successfully assigned from logical server.

2008-08-04T13:00:13,068 WARN  [00000010] :SYSTEM@host - NOTE: Libref stpsamp successfully assigned from logical server.

2008-08-04T13:00:13,068 WARN  [00000010] :SYSTEM@host - NOTE: Libref SASDATA successfully assigned from logical server.
```

Making Detail Data Available to an Information Map for Drill-Through

You can drill through an OLAP report to the underlying detail data only after you make the detail data available to the information map. In order for an information map to produce a report that has drill-through capabilities, an option must first be set in the information map.

For an existing information map, open the information map, right-click it, and then select **`Properties`** from its drop-down menu. Select the **`Allow drill-through to detail data`** check box on the **`Definition`** tab in the Information Map Properties dialog box. This check box is displayed only when a drill-through table is specified for the cube that the OLAP information map is using as its data source.



Information map name: CUBE

Location: /Products/SAS BI Report Services/Maps/
Created: Jul 17, 2008 3:49:43 PM EDT
Author: Marcel Dupree
Last modified: Jul 17, 2008 3:49:43 PM EDT
Modified by: Marcel Dupree
Description:

[✓] Allow drill-through to detail data

Display Detail Data for a Large Cube

If your cube contains an extremely large amount of detail data, then in order to view that data from within SAS Information Map Studio, you might need to increase the Java heap size for SAS Information Map Studio or increase the maximum number of drill-through rows that your SAS OLAP Server can handle. The default number of drill–through rows that can be displayed by a query is 300,000 rows.

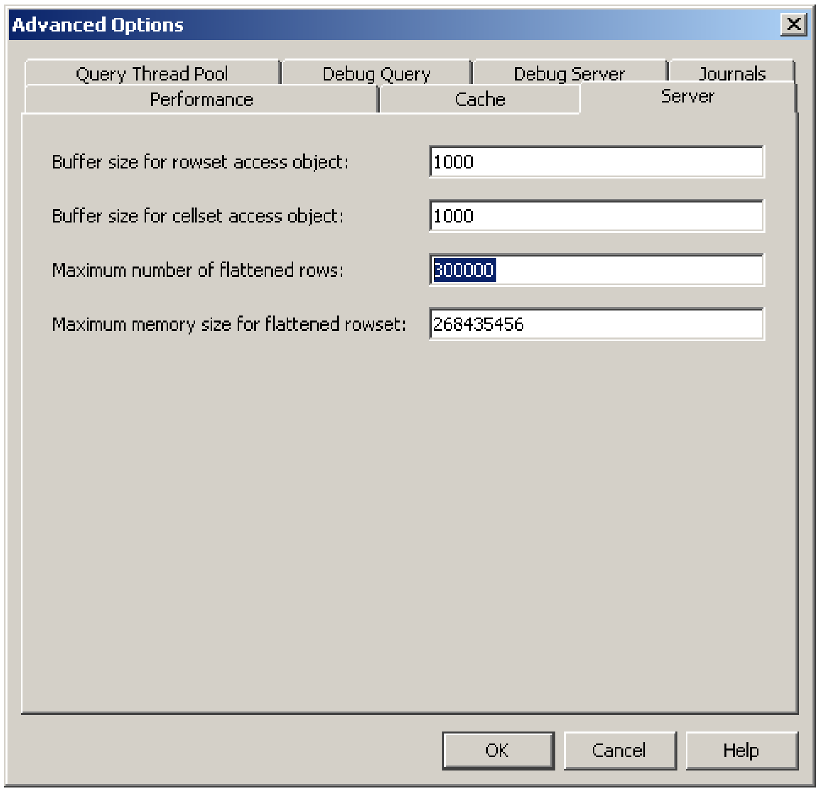
You can increase the number of drill-through rows that your OLAP server can handle by changing the OLAP server definition with the Server Manager plug-in to SAS Management Console. To increase the number of drill-through rows, perform the following steps:

1 In the navigation tree for Server Manager, find the node that represents your physical OLAP server.

2 Right-click the icon and select **`Properties`**

3 In the properties dialog box, select the **`Options`** tab, and then click **`Advanced Options`**.

4 In the Advanced Options dialog box, select the **`Server`** tab, and then enter the desired value for the **`Maximum number of flattened rows`** field.



Buffer size for rowset access object: 1000

Buffer size for cellset access object: 1000

Maximum number of flattened rows: 300000

Maximum memory size for flattened rowset: 268435456

5 Click **`OK`** to save the setting.

APPENDIX 1 Recommended Reading

Recommended Reading

Here is the recommended reading list for this title:

- *SAS Data Integration Studio: User's Guide*
- *SAS Intelligence Platform: Application Server Administration Guide*
- *SAS Intelligence Platform: System Administration Guide*
- *SAS Intelligence Platform: Security Administration Guide*
- *SAS Language Reference: Concepts*
- *SAS Language Reference: Dictionary*
- *SAS Management Console: User's Guide*
- *SAS Metadata LIBNAME Engine: User's Guide*
- *SAS Scalable Performance Data Engine: Reference*
- *Special Considerations for Customers Upgrading to SAS 9.2*

For a complete list of SAS publications, go to **support.sas.com/bookstore**. If you have questions about which titles you need, please contact a SAS Publishing Sales Representative at:

SAS Publishing Sales
SAS Campus Drive
Cary, NC 27513
Telephone: 1-800-727-3228
Fax: 1-919-531-9439
E-mail: **sasbook@sas.com**
Web address: **support.sas.com/bookstore**

Customers outside the United States and Canada, please contact your local SAS office for assistance.

Glossary

aggregation
a summary of detail data that is stored with or referred to by a cube. Aggregations support rapid and efficient answers to business questions.

application server
a server that is used for storing applications. Users can access and use these server applications instead of loading the applications on their client machines. The application that the client runs is stored on the client. Requests are sent to the server for processing, and the results are returned to the client. In this way, little information is processed by the client, and nearly everything is done by the server.

authentication domain
a SAS internal category that pairs logins with the servers for which they are valid. For example, an Oracle server and the SAS copies of Oracle credentials might all be classified as belonging to an OracleAuth authentication domain.

buffer
a portion of computer memory that is used for special holding purposes or processes. For example, a buffer might simply store information before sending that information to main memory for processing, or it might hold data after the data is read or before the data is written.

client application
an application that runs on a client machine.

cube
a logical set of data that is organized and structured in a hierarchical, multidimensional arrangement. A cube is a directory structure, not a single file. A cube includes measures, and it can have numerous dimensions and levels of data.

data mart
a collection of data that is optimized for a specialized set of users who have a finite set of questions and reports.

data warehouse
a collection of data that is extracted from one or more sources for the purpose of query, reporting, and analysis. In contrast to a data mart, a data warehouse is better suited for storing large amounts of data that originates in other corporate applications or which is extracted from external data sources such as public databases.

database management system
a software application that enables you to create and manipulate data that is stored in the form of databases. Short form: DBMS. See also relational database management system.

DBMS
See database management system.

Extensible Markup Language
a markup language that structures information by tagging it for content, meaning, or use. Structured information contains both content (for example, words or numbers) and an indication of what role the content plays. For example, content in a section heading has a different meaning from content in a database table. Short form: XML.

libref
a name that is temporarily associated with a SAS library. The complete name of a SAS file consists of two words, separated by a period. The libref, which is the first word, indicates the library. The second word is the name of the specific SAS file. For example, in VLIB.NEWBDAY, the libref VLIB tells SAS which library contains the file NEWBDAY. You assign a libref with a LIBNAME statement or with an operating system command.

metadata LIBNAME engine
the SAS engine that processes and augments data that is identified by metadata. The metadata engine retrieves information about a target SAS data library from metadata objects in a specified metadata repository.

metadata promotion
in the SAS Open Metadata Architecture, a feature that enables you to copy the contents of a metadata repository to another repository, and to specify changes in the metadata that will be stored in the target repository. For example, you can use this feature to move metadata from a development environment to a testing environment. In such a scenario, you would probably have to change some ports, hosts, and/or schema names as part of the process of moving metadata from one environment to another.

OLAP
See online analytical processing.

OLAP schema
a group of cubes. A cube is assigned to an OLAP schema when it is created, and an OLAP schema is assigned to a SAS OLAP Server when the server is defined in the metadata. A SAS OLAP Server can access only the cubes that are in its assigned OLAP schema.

online analytical processing
a software technology that enables users to dynamically analyze data that is stored in multidimensional database (MDDB) tables. Short form: OLAP.

resource
any object that is registered in a metadata repository. For example, a resource can be an application, a data store, a dimension in an OLAP cube, a metadata item, an access control template, or a password.

resource template
an XML file that specifies the information that is needed for creating a metadata definition for a SAS resource.

SAS Metadata Repository
one or more files that store metadata about application elements. Users connect to a SAS Metadata Server and use the SAS Open Metadata Interface to read metadata

from or write metadata to one or more SAS Metadata Repositories. The metadata types in a SAS Metadata Repository are defined by the SAS Metadata Model.

SAS OLAP Cube Studio
a Java interface for defining and building OLAP cubes in SAS System 9 or later. Its main feature is the Cube Designer wizard, which guides you through the process of registering and creating cubes.

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.

schema
a map or model of the overall data structure of a database. An OLAP schema specifies which group of cubes an OLAP server can access.

XML
See Extensible Markup Language.

Index

Your Turn

We welcome your feedback.

- If you have comments about this book, please send them to **yourturn@sas.com**. Include the full title and page numbers (if applicable).
- If you have comments about the software, please send them to **suggest@sas.com**.