SAS Publishing

SAS® 9.1.3 Intelligence Platform

Data Administration Guide

THE POWER TO KNOW®

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2006. *SAS® 9.1.3 Intelligence Platform: Data Administration Guide*. Cary, NC: SAS Institute Inc.

SAS® 9.1.3 Intelligence Platform: Data Administration Guide

Copyright © 2002-2006, SAS Institute Inc., Cary, NC, USA

ISBN-13: 978–1–59994–167–7
ISBN-10: 1–59994–167–8

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

1st printing, November 2006

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 deals with topics such as setting up shared access to SAS data and explains how using different data-access engines affects security.

Documentation Enhancements

This document contains information that was previously in the *SAS Intelligence Platform: Administration Guide*. Among the enhancements is new information about the following:

- establishing connectivity to XML data
- managing OLAP cube data
- using grid computing to optimize data storage

CHAPTER 1

Overview of Common Data Sources

Overview

This chapter provides a brief introduction to the most common data sources that you encounter as you perform administrative tasks. The features of each data source are described. 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

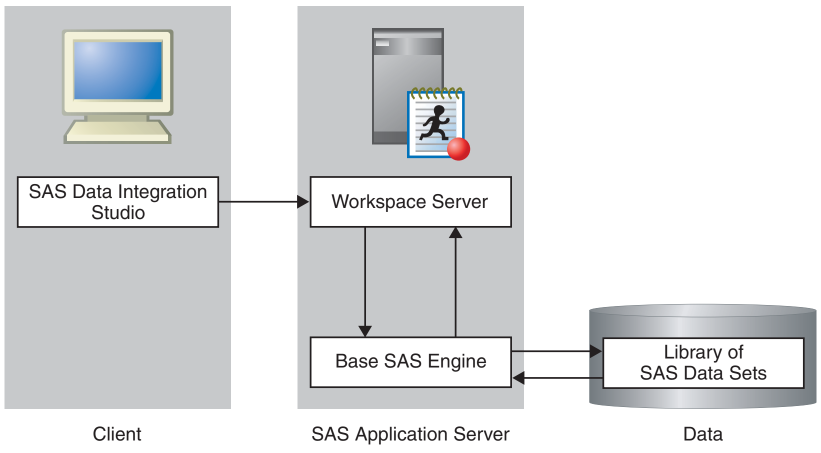
For information about accessibility for any of the products mentioned in this book, see the documentation for that product. If you have questions or concerns about the accessibility of SAS products, send e-mail to **accessibility@sas.com**.

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 data 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 display shows how connectivity to SAS data sets is set up.

Figure 1.1 Establishing Connectivity to SAS Data Sets



See “Establishing Connectivity to a Library of SAS Data Sets” on page 12 for a detailed example of a SAS data set connection.

Shared Access to SAS Data Sets

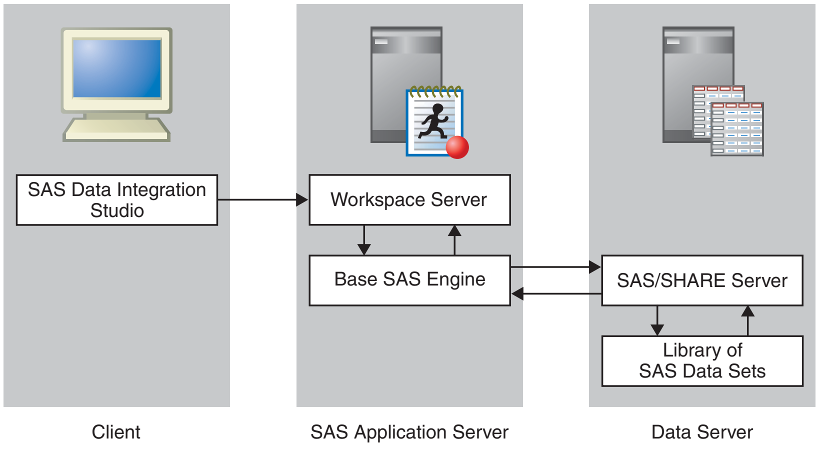
SAS/SHARE software provides concurrent update access to SAS files for multiple users. SAS/SHARE is often required for transaction-oriented applications where multiple users need to update the same SAS data sets at the same time. Data entry applications where multiple users are entering data to the same dataset are a good example of this type of usage. SAS/SHARE software provides both member- 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/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 allows you to make use of SAS software’s ability 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 display illustrates shared access to SAS data sets.

Figure 1.2 Establishing Shared Access to SAS Data Sets



See “Establishing Shared Access to SAS Data Sets” on page 15 for a detailed example of a share SAS data set connection.

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:

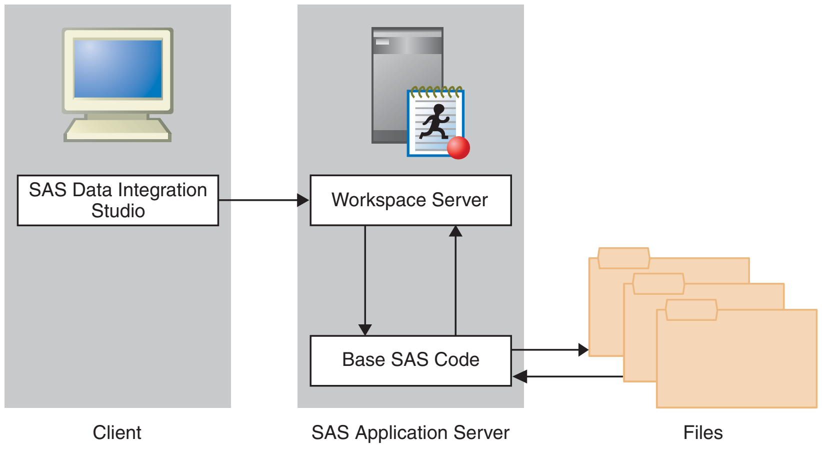
- the delimited external file wizard for external files in which data values are separated with a delimiter character. This wizard enables you to specify multiple delimiters, nonstandard delimiters, missing values, and multi-line records.
- the fixed-width external file wizard for external files in which 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 display illustrates establishing connectivity to external files.

Figure 1.3 Establishing Connectivity to External Files



See "Establishing Connectivity to a Flat File" on page 17 for a detailed example of an external file connection.

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 data 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:

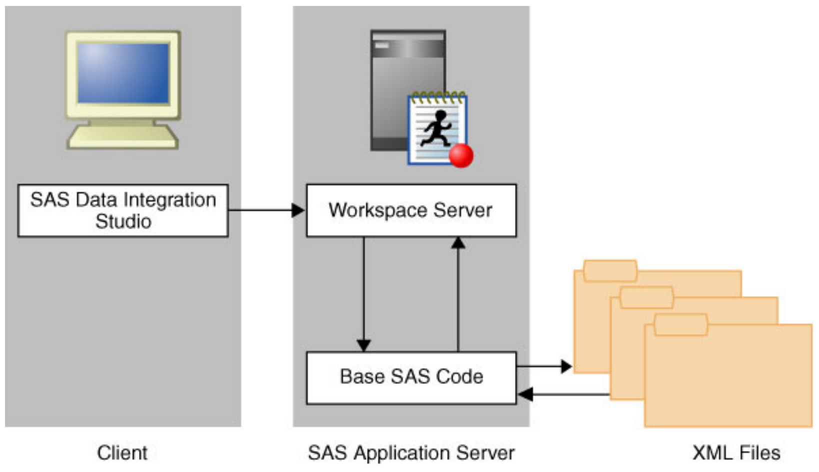
- used by a product that processes XML documents
- moved to another host for the XML LIBNAME engine to then process by translating the XML markup back to a SAS data set

Since 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.

When you are writing an XML file, we recommend that you define the library specifically for your write operation.

The following display illustrates connectivity to XML files.

Figure 1.4 Establishing Connectivity to XML Files



Relational Database Sources

SAS/ACCESS

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

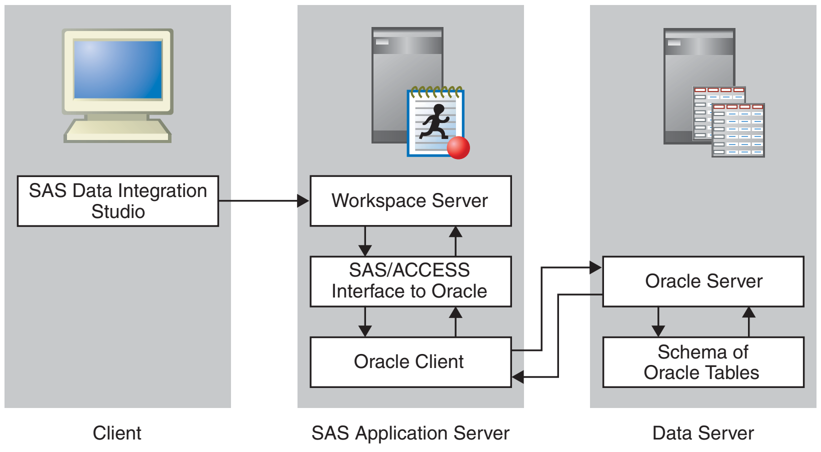
Several of the SAS/ACCESS engines support threaded reads. This enables you to read entire blocks of data on multiple threads instead of reading data just one record at a time. This feature can reduce I/O bottlenecks and enables thread-enabled procedures to read data quickly. These engines and DB2 on 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 display illustrates how connectivity to Oracle databases is set up.

Figure 1.5 Establishing Connectivity to Oracle Databases



See "Establishing Connectivity to an Oracle Database" on page 21 for a detailed example of an Oracle connection.

ODBC Sources

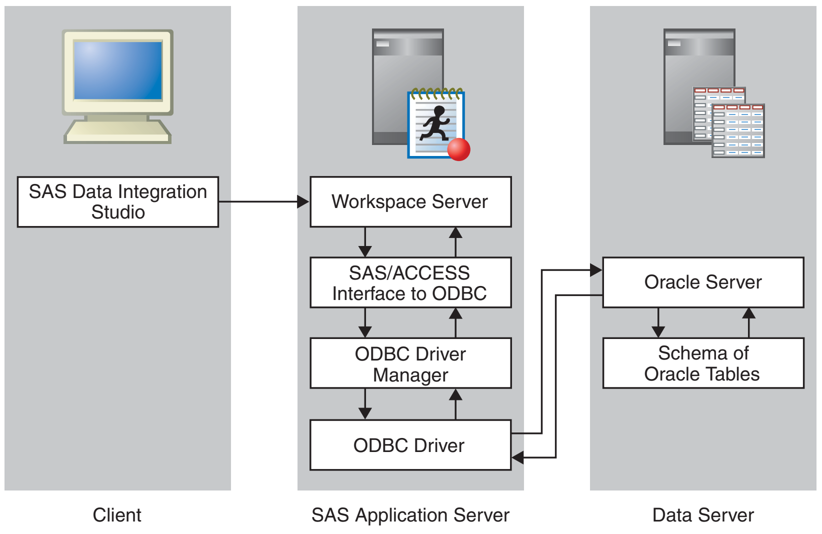
Open database connectivity (ODBC) standards provide a common interface to a variety of databases, including AS/400, dBASE, Microsoft Access, Oracle, Paradox, and Microsoft SQL Server databases. Specifically, ODBC standards define application programming interfaces (APIs) that enable an application to access a database if the ODBC driver adheres to the specification.

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, 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.

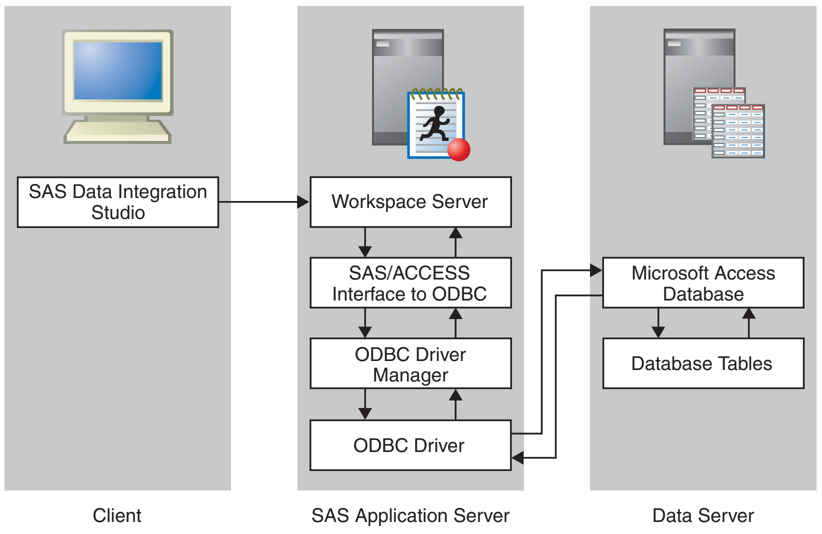
The following display illustrates how ODBC is used to establish connectivity to Oracle databases.

Figure 1.6 Establishing Connectivity to Oracle Databases By Using ODBC



See "Establishing Connectivity to an Oracle Database by Using ODBC" on page 26 for a detailed example of an ODBC-based Oracle connection. The following display illustrates how ODBC is used to establish connectivity to Access databases.

Figure 1.7 Establishing Connectivity to Access Databases By Using ODBC



See "Establishing Connectivity to a Microsoft Access Database by Using ODBC" on page 31 for a detailed example of an ODBC-based Access connection.

Scalable Performance Data Servers

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/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. The SAS SPD Server libraries can now be defined using SAS Management Console.

The 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.

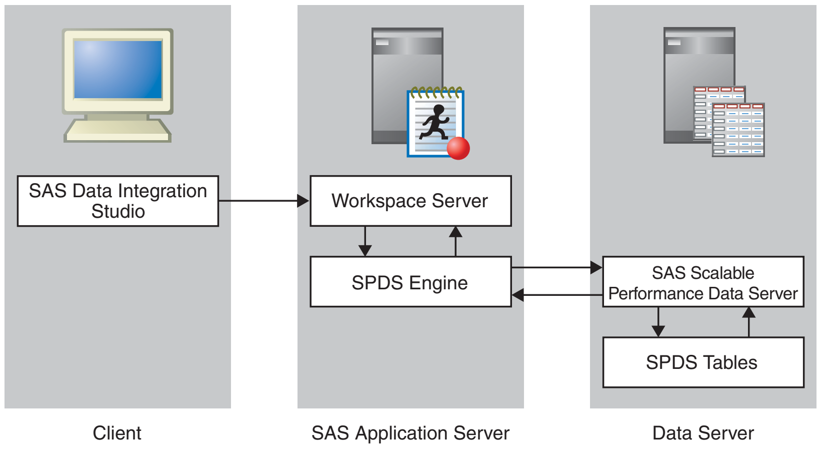
Symmetric Multiprocessing

The SAS SPD Engine 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 Engine 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 Engine 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 Engine, see *SAS Scalable Performance Data Engine: Reference* and **`support.sas.com/rnd/scalability/spde`**.

The following display illustrates how connectivity to SPDS servers is established.

Figure 1.8 Establishing Connectivity to an SPD Server



See "Establishing Connectivity to a Scalable Performance Data Server" on page 34 for a detailed example of an SPDS server connection.

ERP Systems

Enterprise Resource Planning (ERP) systems are composed of thousands of tables, columns, variables and fields. Although they contain a wealth of data, 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)

To make it possible to get to the data the ERP systems contain, SAS provides data surveyors for each of the following ERPs:

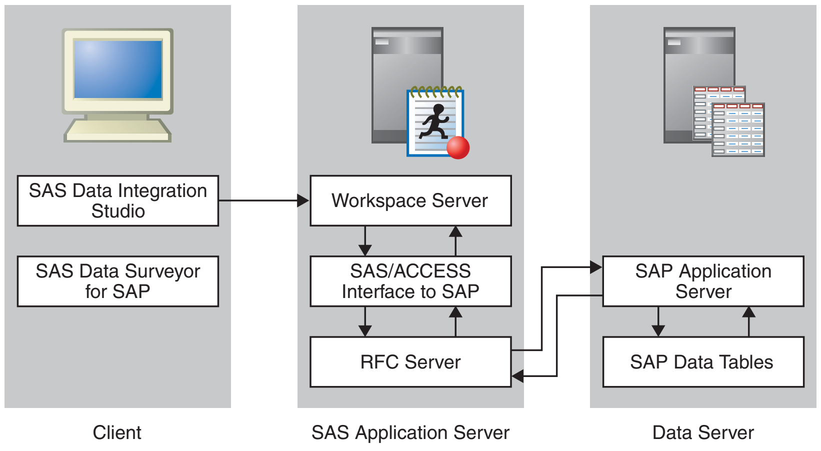
- SAP
- Peoplesoft
- Oracle
- Siebel

Data surveyors contain 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. Each data surveyor has knowledge about the specific application it is designed for. 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 is over 20,000)

The following display illustrates how connectivity to SAP servers is established.

Figure 1.9 Establishing Connectivity to an SAP Server



See "Establishing Connectivity to a SAP Server" on page 38 for a detailed example of an SAP server connection.

CHAPTER 2

Connecting to Common Data Sources

Overview of Connecting to Common Data Sources

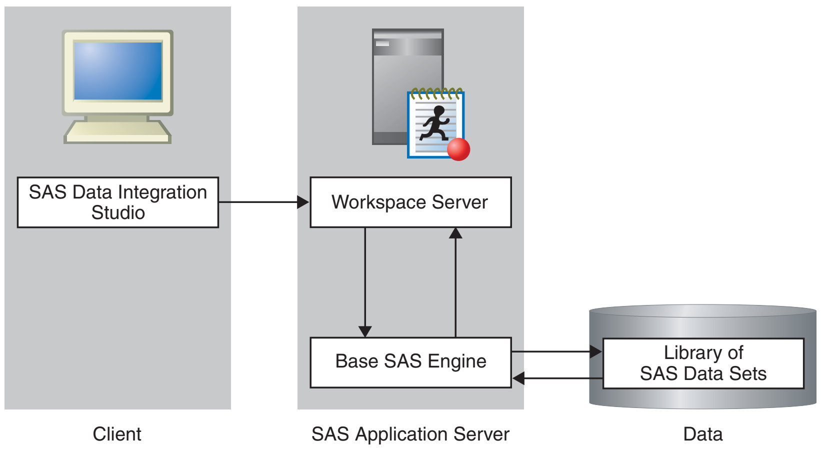
This chapter consists of detailed examples for establishing a connection to each of the common data sources introduced in Chapter 1, "Overview of Common Data Sources," on page 1. Some of the connection processes covered in this chapter have common elements that might be applied to similar data sources. 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.

The chapter also explains a connection verification process that imports tables from the data sources and enables you to view their data in a SAS application. More detailed information about managing table metadata can be found in the Chapter 4, "Managing Table Metadata," on page 57.

Establishing Connectivity to a Library of SAS Data Sets

Define the SAS Base Engine Library

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 define a SAS Base Engine Library metadata object in the metadata repository. 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.

Define the SAS Base Engine Library metadata object 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 dataset contains information about customers of the Orion Gold enterprise. Perform the following steps:

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

2 Select **SAS Base Engine Library** from the **SAS Libraries** 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 wish. Click **Next**.

4 Enter the following library properties:

Table 2.1 Library Properties

Field	Sample Value
Libref	**ORGOLD**
Engine	**BASE**
Path Specification	**C:\SAS\EntBIServer\Lev1\SASMain\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 (Optional) Select one or more SAS servers. (You might not need to select a server. The need to select a server depends on the applications that are included in your environment. A SAS server is not needed for Network File System [NFS] volumes.) The library is assigned to the server or servers that you select from this list. 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 import tables, as explained in “Verifying Access to Tables” on page 42.

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

The preferred way to make a format catalog available is to perform these steps:

1 Name the format catalog formats.sas7bcat.

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

Note: In the z/OS environment, you must perform an additional step. In the SASRX REXX exec, add the following LIBRARY ALLOCATE command, which points to the `SASFormats` directory.

```
allocate dd(library) path('SAS-config-dir/Lev1/SASMain/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.

On Windows and UNIX systems, perform these steps:

1 To the SAS configuration file *SAS-config-dir*`\Lev1\SASMain\sasv9.cfg`, add the CONFIG system option, and use it to point to the user-defined formats configuration file.

```
-config "SAS-config-dir\Lev1\SASMain\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"
-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.

Note: 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 would enter `FMTLIB1` instead of `fmtlib1`. △

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/SASMain/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.

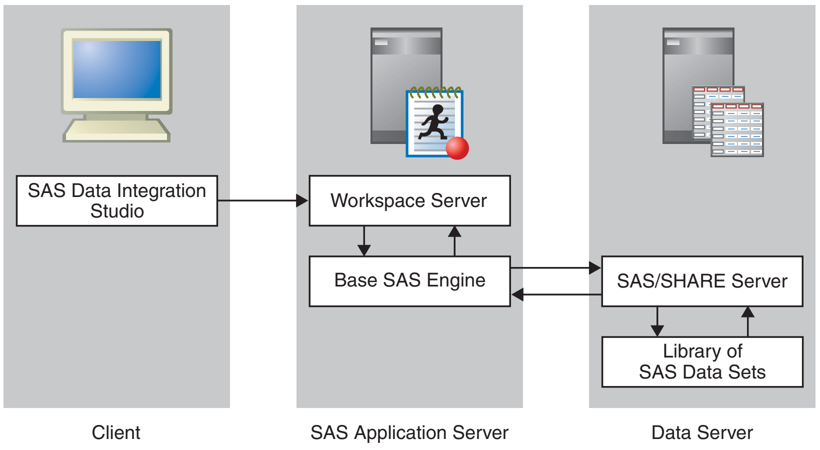
2 In the autoexec file, use the LIBNAME statement to assign the format library and the OPTIONS statement to set the FMTSEARCH system option. For example, you might specify the following statements:

```
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

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 created a SAS/SHARE server metadata object (for example, SHAREServer). Also assume that the other user has already created a SAS Base Engine Library, as described in “SAS Data Sets” on page 1. Setting up shared access to a SAS data set can be seen as a three-stage process, as follows:

1. Assign the SAS Base Engine Library to the SAS/SHARE server.
2. Create a SAS/SHARE Remote Engine Library metadata object.
3. Enable library pre-assignment. In this example, you’re assigning the Orion Gold Customers library to the SHAREServer server.

Stage 1: Assign the SAS Base Engine Library to the SAS/SHARE Server

You need to assign the SAS Base Engine Library for the library that you need to share to the SAS/SHARE Server. In this example, a library that contains information about customers of the Orion Gold enterprise is being shared. Perform the following steps:

1. In SAS Management Console, double-click **`Data Library Manager`**. Double-click **`SAS Libraries`**. Then, right-click the icon for the library that you need to share

(for example, **Orion Gold Customers**). Finally, select the **Properties** option to access the Properties dialog box.

2 Click the **Assign** tab to see a list of available servers. Click the name of the SAS/SHARE server. This step enables the SAS/SHARE server to access the data in the library that you need to share.

3 Click **OK** to save the server assignment.

Stage 2: Create a SAS/SHARE Remote Engine Library

Perform the following steps:

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

2 Select **SAS/SHARE Remote Engine Library** from the **SAS Libraries** list. Click **Next**.

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

4 Enter the following library properties:

Table 2.2 Library Properties

Field	Sample Value
SAS/SHARE Server	**SHAREServer**
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.)
SAS/SHARE Server Library	**Orion Gold Customers** (Use the drop-down menu to select the library that you need to share.)

Click **Next**.

5 Select one or more SAS servers. The library is assigned to the servers included in this list. 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.

Stage 3: Enable Library Pre-Assignment

In order to gain access to the tables, you must perform two additional steps to enable pre-assignment for the library that you need to share, as follows:

1 In SAS Management Console, double-click **Data Library Manager**. Double-click **SAS Libraries**. Then, right-click the icon for the library that you shared (for example, **Orion Gold Customers**). Finally, select the **Properties** option to access the Properties dialog box.

2 Click the **Options** tab. Then, click **Advanced Options** to access the Advanced Options dialog box.

3 Select the **`Library is pre-assigned`** option to enable pre-assignment for the shared library. Click **`OK`** to return to the **`Options`** tab.

Note: When **`Library is pre-assigned`** is selected, SAS applications no longer try to assign a library. Instead, the pre-assigned library is referenced by the applications. △

4 Click **`OK`** to close the Properties dialog box and save the pre-assignment.

5 Update the SAS/SHARE configuration file to include the METAAUTORESOURCES value for the SAS/SHARE server. Perform the following steps:

   a Navigate
   in your local file system to find the SAS/SHARE configuration file (for example, **`C:\SAS\EntBIServer\Lev1\SASMain\ShareServer\sasv9_ShareServer.cfg`**).

   b Open the SAS/SHARE configuration file in a text editor (such as Notepad).

   c Add the following text to the end of the file:

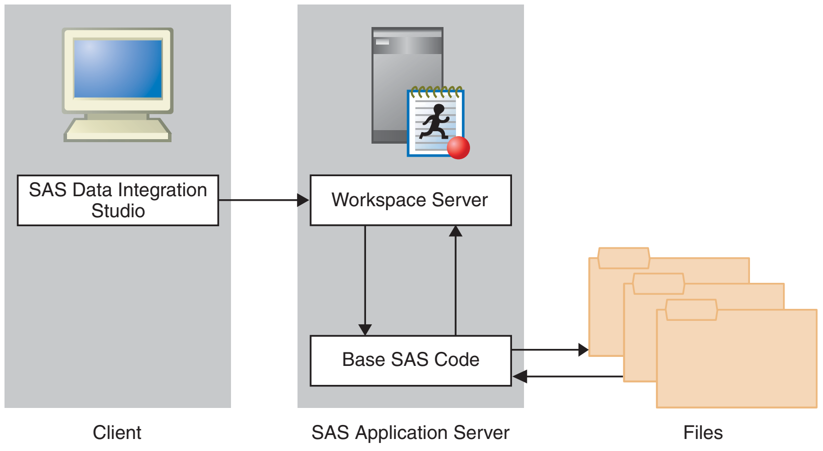
```
-metaautoresources "omsobj:ServerComponent?@Name='SASMain - SHAREServer'"
```

   d Save the configuration file to save the new setting.

At this point, you can import tables, as explained in "Verifying Access to Tables" on page 42.

Establishing Connectivity to a Flat File

Figure 2.3 Establishing Connectivity to External Files



You can connect to a flat file using the External File Source Designer in SAS Data Integration Studio. Setting up a connection from SAS to a flat file can be seen as a three-stage process, as follows:

1 Connect to the flat file.

2 Define the columns in the external file object.

3 Save the external file object.

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.

Stage 1: Connect to the Flat File

Perform the following steps to establish a connection to the flat file:

1 Open the SAS Data Integration Studio application. Then, select **Tools ▶ Source Designer** to access the Source Designer wizard.

2 Select **`Delimited External File`** from the **`External Files`** list. Click **`Next`**.

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

Stage 2: Define the Columns in the External File Object

Perform the following steps to define the columns in an external file object:

1 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.

2 Perform the following steps to define the columns in the external file object:

   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.

   c Click **`OK`** to return to the Column Definitions page.

3 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.

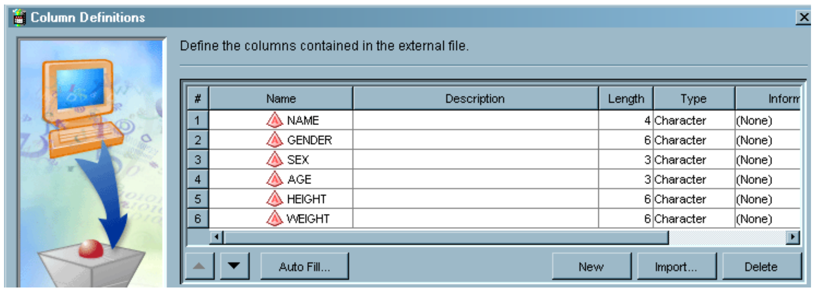
   □ Get column definitions from a COBOL format file.

   □ 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 follow
Namë,C,name,8,,$char8.,cl@ss name column,y
Sëx,C,sex,1,,,cl@ss sex column,n
Agë,N,age,3,,,cl@ss age column,y
# Comma within quotation marks below is not a delimiter
Description,C,Description,32,#char32.,,'Description, Comments, etc.',y
```

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



Define the columns contained in the external file.

#	Name	Description	Length	Type	Inform
1	NAME		4	Character	(None)
2	GENDER		6	Character	(None)
3	SEX		3	Character	(None)
4	AGE		3	Character	(None)
5	HEIGHT		6	Character	(None)
6	WEIGHT		6	Character	(None)

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. △

4 Click **`OK`** to return to the Column Definitions page.

5 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.

6 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 also 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. △

Stage 3: Save the External File Object

Perform the following steps to name and save the external file object:

1 Click **`Next`** to access the Select Group page. Assume that you do not want to specify a custom group for the table in this example.

2 Click **`Next`** to access the General page. Enter an appropriate name in the **`Name`** field (for example, **`Customer Data`**). Note that you can supply an optional description if you wish.

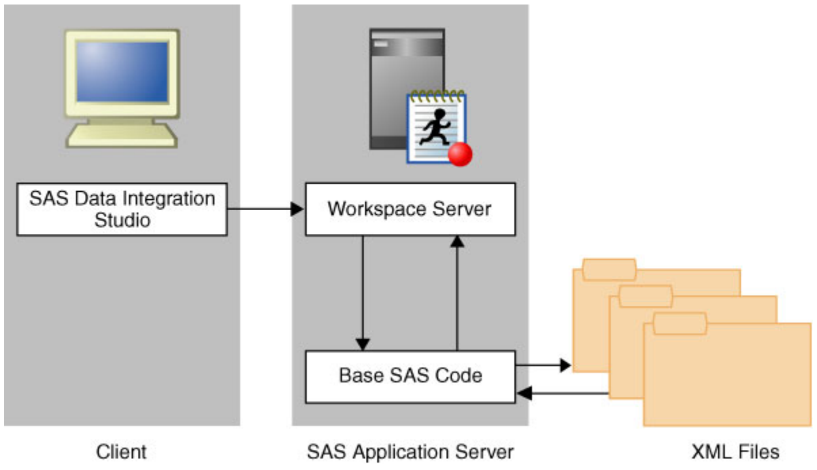
3 Click **`Finish`** to save the metadata and exit the wizard.

At this point, you can import tables, as explained in "Verifying Access to Tables" on page 42.

Establishing Connectivity to XML Data

Connect to the XML Data

Figure 2.4 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 additional information, see the *SAS XML LIBNAME Engine User's Guide*.

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

2 Select **`SAS XML library`** from the **`SAS Libraries`** 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:

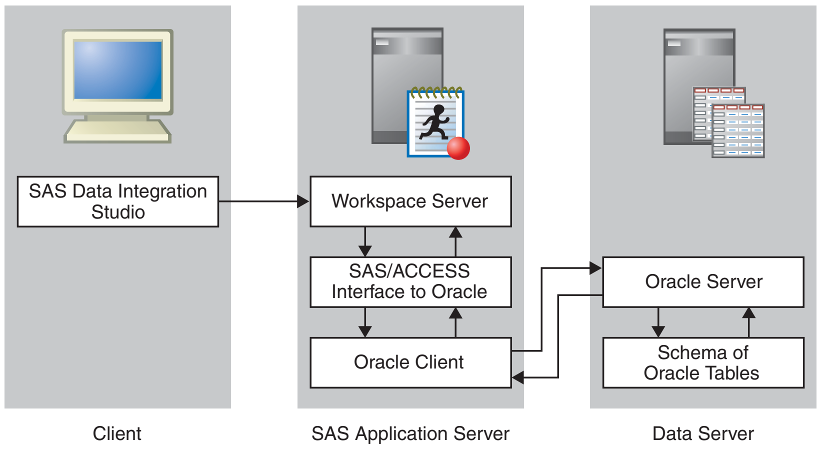
Table 2.3 Library Properties

Field	**Sample Value**
`Name`	**`XML Lib`**
`Libref`	**`xmllib`**
`Engine`	**`XML`**
`XML File`	**`C:\sources\xml\climate.xml`**
`XML Type`	**`GENERIC`**
`Library Access`	**`READONLY`**

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

Establishing Connectivity to an Oracle Database

Figure 2.5 Establishing Connectivity to Oracle Databases



Setting up a connection from SAS to a database management system can be seen as a three-stage process, as follows:

1 Define the database server metadata object.

2 Define the database schema metadata object.

3 Define the database library metadata object.

This example illustrates 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/index.html`** and select the operating system. Then, select the SAS version and click the Planning Installation Edition Kit link. Finally, select the appropriate System Requirements for SAS Foundation document in the Installation category.
- installation of a supported Oracle Database Client.
- validation that the Oracle client can communicate with the Oracle server.
- configuration of SAS/ACCESS environmental variables. Create a UNIX script file that sets up all the environment variables (for example, LD_LIBRARY_PATH_64, LD_LIBRARY_PATH, ODBCHOME, ODBCINI, ORACLE_HOME, SYBASE, INSTHOME, LIBPATH, and SHLIB_PATH). Refer to the Post-Installation Guide or the Configuration Guide in your Installation Kit to see whether this configuration is needed. If this configuration is necessary, follow the instructions in the appropriate document. For information about setting environmental variables when you use SAS/ACCESS to connect to data on UNIX systems, see "Set SAS/ACCESS Environment Variables" on page 22.

Set SAS/ACCESS Environment Variables

If you are attempting to connect to data sources located on UNIX by using SAS/ACCESS, you must set the environmental variables for the workspace server. Use the following steps to set the appropriate environment variables in the SAS Workspace Server:

1 Use the SASENV invocation option to call the DBMS environment script file. From SAS Management Console, select the **`Options`** tab from the Workspace Server Properties dialog box. Add **`-sasenv pathtoscript.txt --`** after **`sas.sh`** in the **`Workspace Server Launch Command`** field. For example, enter the following:

```
/BIArchitecture/Lev1/SASMain/sas.sh -sasenv dbmsscript.txt --
```

Note: The trailing **`--`** is required. The file name **`pathtoscript.txt`** is any script file name containing the DBMS environment variable settings provided by your DBA. △

2 Restart the object spawner using the **`objectspawner.sh`** script file. For example, enter the following:

```
/BIArchitecture/Lev1/SASMain/ObjectSpawner/objectspawner.sh
```

3 In SAS Management Console, right-click the **`Workspace Server`** connection and select the **`Test Connection`** option to verify that the workspace server starts correctly with the new DBMS environment script file.

Stage 1: Define the Database Server

Perform the following steps to define the database server metadata object:

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 wish. Click **`Next`**.

5 Enter the following server properties:

Table 2.4 Server Properties

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

Click **`Next`**.

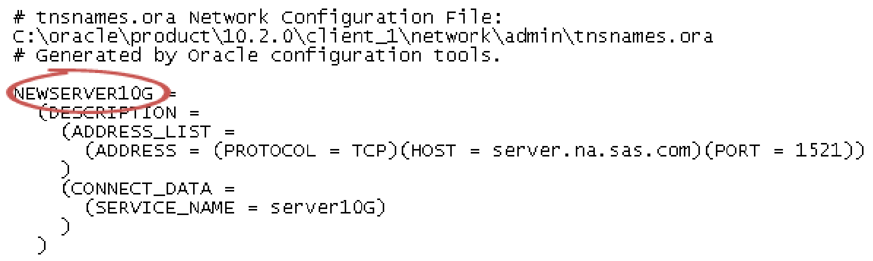
6 Enter the following connection properties:

Table 2.5 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 **`C:\oracle\product\10.2.0\client_1\NETWORK\ADMIN\tnsnames.ora`**. The alias for the connection information is contained in this file. See the following display.)
`Authentication Domain`	**`OracleAuth`** (You need to create a new authentication domain each time you connect to a new database server. The SAS Metadata Server, Workspace Server, and others require one set of credentials, while the database server requires another. Click **`New`** to access the New Authorization Domain dialog box. Then enter the appropriate value in the **`Name`** field and click **`OK`** to save the setting. Make sure that the authentication domain that you create here is added to the appropriate users and user groups. Add a login to these users and groups that includes a user ID, a password, and the authentication domain.)

Note: If you will have multiple users for the connection, consider creating a user group for them to avoid the inefficient process of creating separate user IDs and passwords for each. Use the Access Control Template in the Authorization Manager in SAS Management Console. For more information, see "Example: Use a Custom ACT" in the chapter "Securing a Deployment" in the *SAS Intelligence Platform: Security Administration Guide*. △

The following display 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: Define the Database Schema

After you have defined the database server metadata, you can define the database schema metadata object. The server object must be defined first because the server object must be entered into the wizard when you define the schema object. (If needed, you can define several schemas per server.) Perform the following steps:

1 In SAS Management Console, double-click **Data Library Manager**. Then, right-click **Database Schema** and select the **New Database Schema** option to access the New Database Schema Wizard.

2 Select **Oracle Schema** from the **Database Schemas** list. Click **Next** to access the next page of the wizard.

3 Enter an appropriate schema name in the **Name** field (for example, **Oracle Schema**). Note that you can supply an optional description if you wish. Click **Next**.

4 Enter the following schema properties:

Table 2.6 Schema Properties

Field	Sample Value
Database Schema Name	**Schema Name** (This value needs to be the name of an existing schema).
Oracle Server	**Oracle Server** (Use the value that you entered in the **Name** field in the New Server Wizard when you defined the database server metadata object. In this case it must be **Oracle Server**, which is the name of the metadata object representing the Oracle server.)

Click **Next**.

Note: The case of the schema names depends upon the database type. For example, DB2 schema names are uppercased by default. △

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

Stage 3: Define the Database Library

After you have defined the database server metadata object and the database schema metadata object, you can define the database library metadata object. It is important to define the library object after the server and schema objects because these objects must be entered into the wizard when you define the library object. Perform the following steps:

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

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

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

4 Enter the following library properties:

Table 2.7 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.

5 Enter the following settings:

Table 2.8 Advanced Option Settings

Field	Sample Value
`Database Server`	**`OracleServer`** (Use the database server that you created in the **`New Server Wizard`**.)
`Default Login`	**`Login`** (Select your login from the drop-down list. This default login is used to resolve conflicts between multiple logins to an authentication domain. In such cases, the default login is used.)
`Database Schema`	**`OracleSchema`** (Select the schema name that you entered in the **`New Database Schema Wizard`**.)

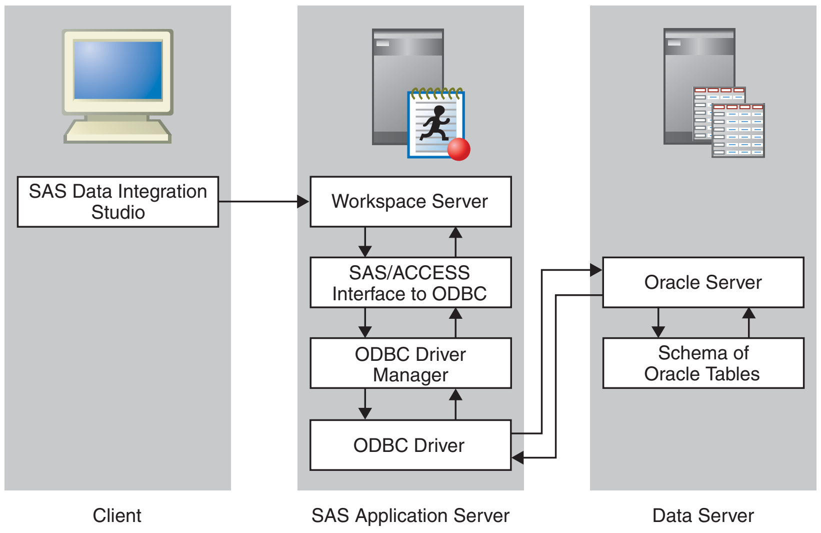
Click **`Next`**.

6 (Optional) Select one or more SAS servers. (You might not need to select a server. The need to select a server depends on the applications that are included in your environment.) The library is assigned to the servers included in this list. 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 import tables, as explained in “Verifying Access to Tables” on page 42.

Establishing Connectivity to an Oracle Database by Using ODBC

Figure 2.6 Establishing Connectivity to Oracle Databases By Using ODBC



Setting up a connection from SAS to an Oracle database management system by using ODBC can be seen as a four-stage metadata definition process, as follows:

1. Define an ODBC data source.
2. Define the database server metadata object.
3. Define the database schema metadata object.
4. Define the database library metadata object.

This example illustrates 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/index.html`** and select the operating system. Then, select the SAS version and click the Planning Installation Edition Kit link. Finally, select the appropriate System Requirements for SAS Foundation document in the Installation category.
- □ installation of a supported Oracle Database Client if your ODBC driver requires a client. Refer to the ODBC driver vendor's documentation to determine if an Oracle client is required.
- □ validation that the Oracle client can communicate with the Oracle server.
- □ configuration of SAS/ACCESS environmental variables. Create a UNIX script file that sets up all the environment variables (for example, LD_LIBRARY_PATH_64, LD_LIBRARY_PATH, ODBCHOME, ODBCINI, ORACLE_HOME, LIBPATH, and SHLIB_PATH). Refer to the Post-Installation Guide or the Configuration Guide in your Installation Kit to see whether this configuration is needed. If this

configuration is necessary, follow the instructions in the appropriate document. For information about setting environmental variables when you use SAS/ACCESS to connect to data on UNIX systems, see "Set SAS/ACCESS Environment Variables" on page 22.

Stage 1: Define the Data Source

First, you must define the ODBC data source. On Window systems, you should perform these 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 Oracle driver listed in the window (for example, **`Oracle in OraClient10g_home1`**). Click **`Finish`** to access the Oracle ODBC Driver Configuration dialog box.

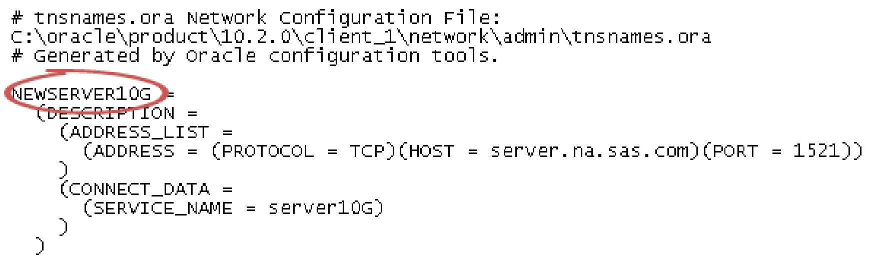
   Note: System data sources and user data sources store information about how to connect to the indicated data provider. A system data source is visible to all users with access to the system, including NT services. A user data source is only visible to a particular user, and it can only be used on the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

3 Enter the following configuration settings:

Table 2.9 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 display.)
`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)
    )
  )
```

Note that the correct **`TNS Service Name`** value is circled. You can click **`Test Connection`** to verify the settings.

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: Define the Database Server

Perform the following steps to define the database server metadata object:

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**). Note that you can supply an optional description if you wish. One server is required for each DSN. Click **Next**.

5 Enter the following server properties:

Table 2.10 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** (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 2.11 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 Domain	**ODBCAuth** (You need to create a new authentication domain each time you connect to a new database server. The SAS Metadata Server, Workspace Server, and others require one set of credentials, while the database server requires another. Click **New** to access the New Authorization Domain dialog box. Then enter the appropriate value in the **Name** field and click **OK** to save the setting. Make sure that the authentication domain that you create here is added to the appropriate users and user groups. Add a login to these users and groups that includes a user ID, a password, and the authentication domain.

Note: If you will have multiple users for the connection, consider creating a user group for them to avoid the inefficient process of creating separate user IDs and passwords for each. Use the Access Control Template in the Authorization Manager in SAS Management Console. △

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: Define the Database Schema

After you have defined the database server metadata, you can define the database schema metadata object. It is important to define the server object first because the server object must be entered into the wizard when you define the schema object. Perform the following steps:

1 In SAS Management Console, double-click **`Data Library Manager`**. Right-click **`Database Schemas`** and select the **`New Database Schema`** option to access the New Database Schema Wizard.

2 Select **`ODBC Schema`** from the **`Database Schemas`** list. Click **`Next`**.

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

4 Enter the following schema properties:

Table 2.12 Schema Properties

Field	**Sample Value**
`Database Schema Name`	**`ODBCSchema`** (The database schema name is case sensitive.)
`ODBC Server`	**`ODBC Server`** (Select the server from the drop-down list.)

Click **`Next`**.

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

Stage 4: Define the Database Library

After you have defined the database server metadata object and the database schema metadata object, you can define the database library metadata object. It is important to define the library object after the server and schema objects because these objects must be entered into the wizard when you define the library object. Perform the following steps:

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

2 Select **`ODBC Library`** from the **`Database Libraries`** 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 wish. Click **`Next`**.

4 Enter the following library properties:

Table 2.13 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.

5 Enter the following settings:

Table 2.14 Advanced Option Settings

Field	**Sample Value**
`Database Server`	**`ODBCServer`** (Use the database server that you selected in the **`New Server Wizard`**.)
`Default Login`	**`Login(ODBCAuth)`** (Select your login from the drop-down list. This default login is used to resolve conflicts between multiple logins to an authentication domain. In such cases, the default login is used.)
`Database Schema`	**`ODBCSchema`** (Select the schema name that you entered in the **`New Database Schema Wizard`**.)

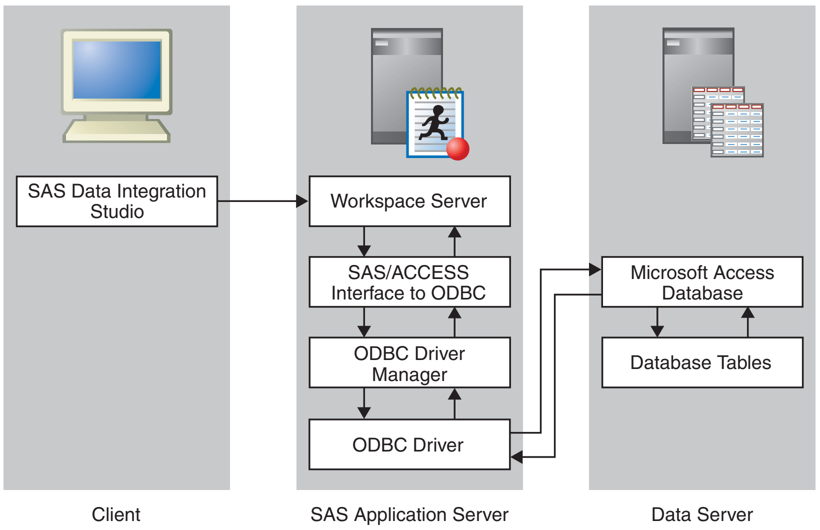
Click **`Next`**.

6 Select one or more SAS servers. The library is assigned to the servers included in this list. 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 import tables, as explained in “Verifying Access to Tables” on page 42.

Establishing Connectivity to a Microsoft Access Database by Using ODBC

Figure 2.7 Establishing Connectivity to Access Databases By Using ODBC



Setting up a connection from SAS to a Microsoft Access database by using ODBC can be seen as a four-stage process, as follows:

1 Define an ODBC data source.
2 Define the database server metadata object.
3 Define the database schema metadata object.
4 Define the database library metadata object.

This example illustrates 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. The following prerequisite has been satisfied:

□ installation of SAS/ACCESS Interface to ODBC

Stage 1: Define the Data Source

First, you must define the ODBC data source. On Windows systems, perform these 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 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 NT services. A user data source is only visible to a particular user, and it can only be used on the current machine. △

3 Enter the following configuration settings:

Table 2.15 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: Define the Database Server

Perform the following steps to define the database server metadata object:

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 wish. Click **`Next`**.

5 Enter the following server properties:

Table 2.16 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`** (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 2.17 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 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: Define the Database Schema

Most databases include schemas that must be identified as part of the connection process. Microsoft Access databases, though, require that you define a separate library for each DSN, and they do not use schemas. Nevertheless, SAS expects to connect to a schema. For this reason, you must create a blank schema object as a placeholder. Perform the following steps:

1 In SAS Management Console, double-click **`Data Library Manager`**. Then, right-click **`Database Schemas`** and select the **`New Database Schema`** option to access the New Database Schema Wizard.
2 Select **`ODBC Schema`** from the **`Database Schemas`** list. Click **`Next`**.
3 Enter an appropriate server name in the **`Name`** field (for example, **`MS Access Schema`**). Note that you can supply an optional description if you wish. Click **`Next`**.
4 Enter the following schema properties:

Table 2.18 Schema Properties

Field	Sample Value
`Database Schema Name`	Leave this field blank. (This field must be left blank because SAS expects a schema value, but Microsoft Access does not use one.)
`ODBC Server`	**`MS Access Server`** (Select the server from the drop-down list.)

Click **`Next`**.

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

Stage 4: Define the Database Library

After you have defined the database server metadata object and the database schema metadata object, you can define the database library metadata object. It is important to define the library object after the server and schema objects because these objects must be entered into the wizard when you define the library object. Perform the following steps:

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

2 Select **ODBC Library** from the **Database Libraries** 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 wish. Click **Next**.
4 Enter the following library properties:

Table 2.19 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.
5 Enter the following setting:

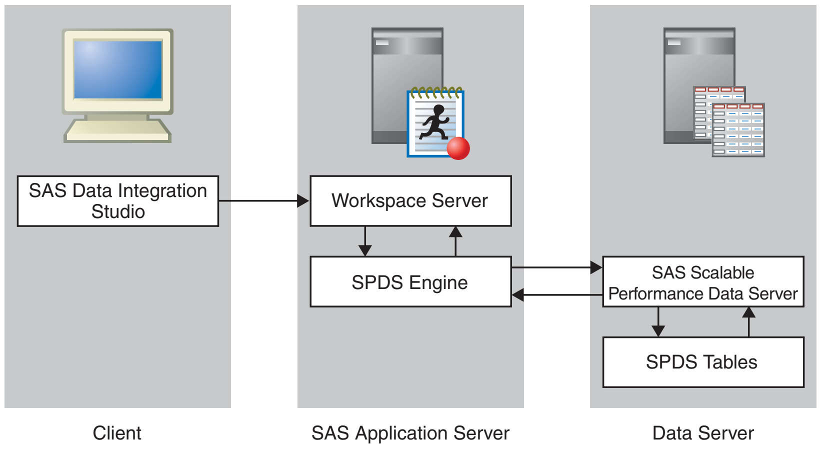
Table 2.20 Advanced Option Setting

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

Click **Next**.
6 Select one or more SAS servers. The library is assigned to the servers included in this list. 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 import tables, as explained in "Verifying Access to Tables" on page 42.

Establishing Connectivity to a Scalable Performance Data Server

Figure 2.8 Establishing Connectivity to an SPD Server



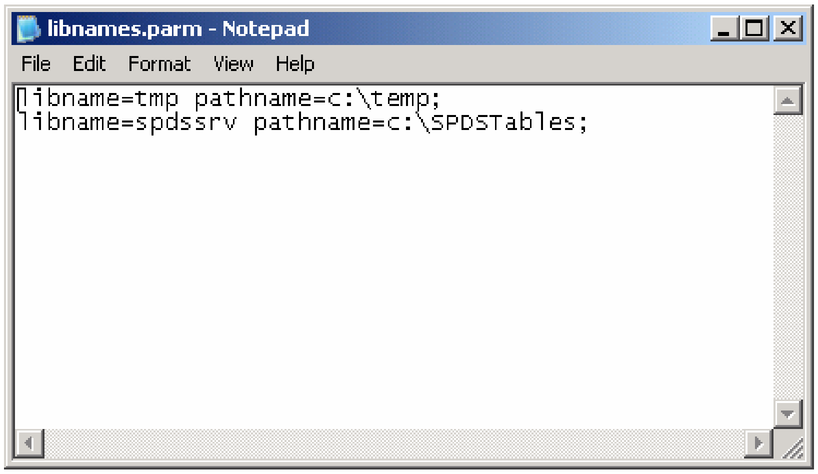
Setting up a connection from SAS to a Scalable Performance Data Server (SPD Server) can be seen as a four-stage process, as follows:

1. Configure the **`libnames.parm`** file.
2. Define the server metadata object.
3. Define the server domain metadata object.
4. Define the library metadata object.

This example illustrates 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 display:



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

Stage 2: Define the Server

Perform the following steps to define the database server metadata object:

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`**). Note that you can supply an optional description if you wish. Click **`Next`**.
5. Enter the following server properties:

Table 2.21 Server Properties

Field	Sample Value
Major Version Number	**4**
Minor Version Number	**0**
Vendor	**SAS Institute**
SAS Compatibility	**SAS 9**

Click **Next**.

6 Enter the following connection properties:

Table 2.22 Connection Properties

Field	Sample Value
Host	**D1234**
Port Number or Name	**5200** (Enter the port number for the SPDS name server.)
Communication Protocol	**TCP**
Authentication Domain	**SPDSAuth** (You need to create a new authentication domain each time you connect to a new server. The SAS Metadata Server, Workspace Server, and others require one set of credentials, while the SPDS requires another. Click **New** to access the New Authorization Domain dialog box. Then enter the appropriate value in the **Name** field and click **OK** to save the setting. Make sure that the authentication domain that you create here is added to the appropriate users and user groups. Add a login to these users and groups that includes a user ID, a password, and the authentication domain.)

Note: If you will have multiple users for the connection, consider creating a user group for them to avoid the inefficient process of creating separate user IDs and passwords for each. Use the Access Control Template in the Authorization Manager in SAS Management Console. △

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: Define the Server Domain

After you have defined the server metadata, you can define the server domain metadata object. It is important to define the server object first because the server object must be entered into the wizard when you define the domain object. Perform the following steps:

1 In SAS Management Console, double-click **Data Library Manager**. Then, right-click **Database Schemas** and select the New Database Schema option to access the New Database Schema Wizard.

2 Select **SAS Scalable Performance Data Server Domain** from the **Database Schemas** list. Then, click **Next**.

3 Enter an appropriate server domain name in the **Name** field (for example, **SPDServerDomain**). Note that you can supply an optional description if you wish. Click **Next**.

4 Enter the following domain properties:

Table 2.23 Server Domain Properties

Field	Sample Value
LIBNAME Domain Name	**spdssrv** (This value will be entered into the **libnames.parm** file).
SPD Server	**SPDSServer** (Use the value that you entered in the **Host** field of the Connection Properties dialog box when you defined the server metadata object.)

Click **Next**.

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

Stage 4: Define the Library

After you have defined the server metadata object and the server domain metadata object, you can define the library metadata object. It is important to define the library object after the server and domain objects because these objects must be entered into the wizard when you define the library object. Perform the following steps:

1 In SAS Management Console, double-click **Data Library Manager**. Right-click **SAS 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 Libraries** list. Click **Next**.

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

4 Enter the following library properties:

Table 2.24 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.

5 Enter the following settings:

Table 2.25 Advanced Option Settings

Field	Sample Value
`SPD Server`	**`SPDSServer`** (Use the database server that you selected in the New Server Wizard.)
`Default Login`	**`(None)`** (Keep this default value. SPD Server does not use a schema, but SAS expects a value in this field.)
`LIBNAME Domain`	**`spdsrv`** (Select the domain name that you entered in the New Database Schema Wizard.)

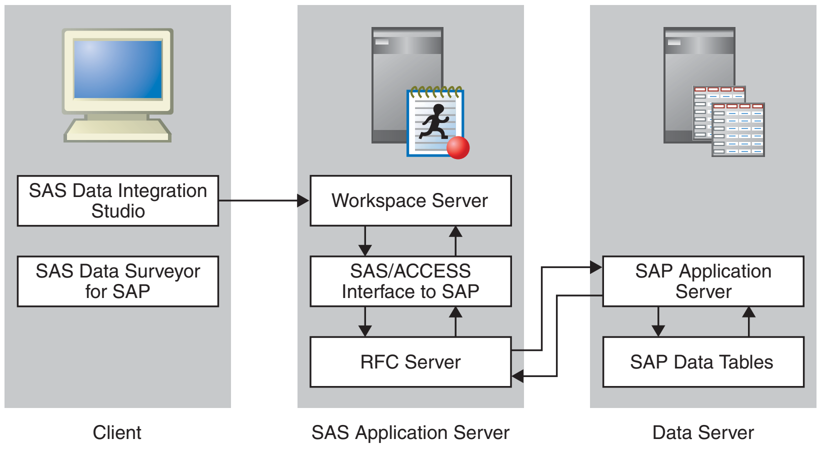
Click **`Next`**.

6 Select one or more SAS servers. (You might not need to select a server. The need to select a server depends on the applications that are included in your environment.) The library is assigned to the servers included in this list. Click **`Next`** to access the next page of the wizard.

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 import tables, as explained in "Verifying Access to Tables" on page 42.

Establishing Connectivity to a SAP Server

Figure 2.9 Establishing Connectivity to a SAP Server



Setting up a connection from SAS to a SAP server can be seen as a three-stage process, as follows:

1 Define the server metadata object.

2 Define the server domain metadata object.

3 Define the library metadata object.

This example illustrates 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 by the SAS RFC service.
- □ SAS/ACCESS Interface to R/3. This installation installs the SAS RFC service. Note that you must manually start this service on both Windows and UNIX each time that you start the SAS server.

Stage 1: Define the Server

Perform the following steps to define the SAS server metadata object:

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 wish. Click **`Next`**.

5 Enter the following server properties:

Table 2.26 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 2.27 Connection Properties

Field	Sample Value
`Authentication Domain`	**`SAPAuth`** (You need to create a new authentication domain each time you connect to a new server. The SAS Metadata Server, Workspace Server, and others require one set of credentials, while the SAP server requires another. Click **`New`** to access the New Authorization Domain dialog box. Then enter the appropriate value in the **`Name`** field and click **`OK`** to save the setting. Make sure that the authentication domain that you create here is added to the appropriate users and user groups. Add a login to these users and groups that includes a user ID, a password, and the authentication domain.)
`RFC Server Host`	**`localhost`** (The SAP Server can be installed on the same machine as the workspace server, on the same machine as the application server, or on any machine in the network. If it is installed on the same machine as the workspace server, this value is populated automatically. If it is installed anywhere else, enter the appropriate value.)

Field	Sample Value
RFC Server Port	**6999** (This is default value, populated automatically. If the SAP server is installed on the same machine as the workspace server, this is the correct value. If the SAP server is installed anywhere else, enter the appropriate value instead.)
Client	**800** (This value is obtained from your SAP administrator.)
Language	**EN** (This value is obtained from your SAP administrator.)

Note: If you will have multiple users for the connection, consider creating a user group for them to avoid the inefficient process of creating separate user IDs and passwords for each. Use the Access Control Template in the Authorization Manager in SAS Management Console. △

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. 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: Define the Schema

After you have defined the server metadata, you can define the schema metadata object. It is important to define the server object first because the server object must be entered into the wizard when you define the schema object. Perform the following steps:

1 In SAS Management Console, double-click **Data Library Manager**. Then, right-click **Database Schema** and select the **New Database Schema** option to access the New Database Schema Wizard.

2 Select **SAP Schema** from the **Enterprise Application Schemas** list. Then, click **Next**.

3 Enter an appropriate server domain name in the **Name** field (for example, **SAP Schema**). Note that you can supply an optional description if you wish. Click **Next**.

4 Enter the following schema properties:

Table 2.28 Server Domain Properties

Field	Sample Value
`Database Schema Name`	Leave this field blank.
`SAP Server`	**`SAPServer`** (Use the value that you entered in the **`Name`** field of the New Server Wizard when you defined the server metadata object.)

Click **`Next`**.

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

Stage 3: Define the Library

After you have defined the server metadata object and the server domain metadata object, you can define the library metadata object. It is important to define the library object after the server and schema objects because these objects must be entered into the wizard when you define the library object. Perform the following steps:

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

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

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

4 Enter the following library properties:

Table 2.29 Library Properties

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

Click **`Next`**.

5 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`**.

6 (Optional) Select one or more SAS servers. (You might not need to select a server. The need to select a server depends on the applications that are included in your environment.) The library is assigned to the servers included in this list. 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 import tables, as explained in "Verifying Access to Tables" on page 42.

Verifying Access to 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.

Verifying your access to tables in SAS Management Console can be seen as a two-stage process, as follows:

1 Import the tables.
2 View the data in SAS Management Console.

Stage 1: Import the Tables

To import 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 Double-click **`Data Library Manager`**. Then, double-click **`SAS Libraries`** to see the list of libraries. Right-click the library that contains the tables that you need to import. Then, select the **`Import Tables`** option to access the Connect to SAS page of the Import Tables wizard.
3 Select a server.
4 Select the library that contains the tables from the **`SAS Library`** drop-down list.
5 Verify that the values shown in the fields in the **`Library details`** group box are correct. Click **`Next`**.
6 Click the tables that you need to select. (Hold down the CTRL key and click to select more than one table.) Click **`Next`**.
7 Examine the final page of the wizard to ensure that the proper values have been entered. Click **`Finish`**.

Note: You can also import tables using the Source Designer in SAS Data Integration Studio or through SAS Data Surveyor. △

Stage 2: View 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 perform the following steps in SAS Data Integration Studio:

1 Navigate to the **`Inventory`** tree and double-click **`Tables`**.
2 Right-click a table that you need to verify and select the **`View Data`** option. Examine the data contained in the table in the View Data window.
3 Close the View Data window.
4 Optionally, you can also examine the table's Properties field. Right-click a table, and select the **`Properties`** option.
5 Click the **`Columns`** tab to see column data for the table.
6 Click the **`Advanced`** tab. Notice that the time and date that the metadata for the table was created is displayed in the **`MetadataCreated`** field. Click **`Cancel`** to dismiss the Properties dialog box.

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 11, you learned how to define SAS libraries in metadata. 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 how to assign libraries so that the SAS servers in the environment know where the libraries are located and how to access them.

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\EntBIServer\Lev1\SASMain\Data\orgold' repname=Foundation;
```

In this case, the libref ORGOLD tells the SAS session that it should access data sets in the directory **`C:\SAS\EntBIServer\Lev1\SASMain\Data\orgold`** using the BASE 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 makes use of librefs. Before this code can execute, the corresponding library must have been assigned, and the server that will execute 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 MLE" on page 45. SAS clients and stored processes can access a library using one of two engines:

- □ the engine specified in the library's metadata. This would be the Base SAS engine for libraries of SAS data sets, the ORACLE engine for Oracle libraries, and so forth.
- □ the metadata LIBNAME engine (MLE).

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

Note: Avoid the "Pre-assigned Library" 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`** checkbox. Do not use the specialized "Pre-assigned Library" template, which is intended for certain system libraries only; it will not work for other libraries. △

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 MLE 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:

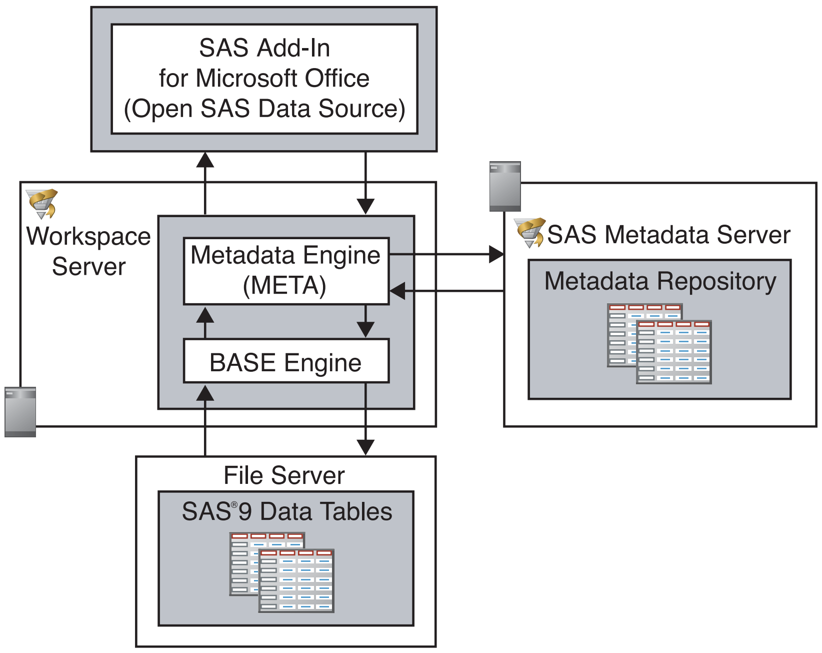
- □ Environments where users are executing stored processes, and you don't want programmers having to manage library assignments in their code or in autoexec files.
- □ Environments where the Data Step Batch Server is used to execute jobs created by SAS Data Integration Studio, and library assignments 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 defined in the metadata. Recall that when you define a client-assigned library (a library that is not pre-assigned), SAS Enterprise Guide and SAS Add-In for Microsoft Office will assign the library to use the META engine by default. Recall further that a library assigned with the MLE should not be used as the location for output or target tables.

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 MLE

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 MLE. As shown in the following display, the MLE invokes the SAS/ACCESS engine stored in the metadata.

Display 3.1 MLE Invocation of the SAS/ACCESS 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 preceding diagram, when SAS users expand a library that they have ReadMetadata access to and that has been assigned to use the MLE, 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 MLE sends a query directly to the authorization facility asking for the their data-level permissions on that table. If the users or the group to which they belong have at least Read access to the table, the MLE 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.

Client applications 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 if 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, access to the table.

In contrast, data-level authorizations of Read, Write, Create, and Delete are never checked by the metadata server, because a metadata object is not involved. As a result, if you need to use the SAS authorization facility to control which users can access a physical table or library, then you need to assign the server-side library using the MLE. When used with its default options, the MLE will query the metadata server for metadata-based permissions. The SAS authorization facility must be queried for data-level permissions. When MLE 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 META 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 defined in the metadata would look something like the following:

```
ORGOLD META library="Orion Gold Customers" METAEEPOSITORY="Foundation";
```

This is the minimum information you would need to supply in the LIBNAME statement itself. However, this statement will work 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 defined in the **sasv9.cfg** file already if you used the Configuration Wizard to set up your environment.

Having data requests flow through the MLE before they reach the engine that actually fulfills the request provides an important capability: the MLE 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 MLE takes away some capabilities. Most important, the MLE should not be used in its default mode to create or delete tables. It is most suited to read-only applications.

Using Libraries That Are Not Pre-assigned

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 employed. For example, if you do not pre-assign the library, SAS Data Integration Studio will assign it for the user in such a way that data-level authorizations of Read, Write, Create, and Delete are not imposed on the clients. Data requests are sent directly to the engine specified in the library's metadata (such as BASE) without checking data-level authorizations. This approach was chosen as a best practice, because it is assumed that in most cases SAS Data Integration Studio developers will be 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 META engine. Instead, it assigns the library using the engine specified in the metadata. Because SAS Data Integration Studio only works with tables that are defined 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 META 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 employed 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 META 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

Application	Pre-assigned	Library Engine Used	Minimum Metadata Authorizations Required
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®9 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 META 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 defined in the metadata repository, thus preventing a stored process from accessing tables that might reside in the library but are not defined in the 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 defined 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 defined 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 assignment(s) in an external file and then include it 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 previous two methods.
    - Disadvantage: The file(s) 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 will have to be modified manually to replace the library assignment with manually generated %INCLUDE syntax.
    - Disadvantage: Changes to the file's location or name would require that all stored processes, including that file, be updated.

Pre-assigning Libraries Using Engines Other Than the MLE

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 MLE. In either case, pre-assignment allows you to designate an assignment method for use by all of the applications that use the library. However, it should not be used when you create output from the applications if the assignment is made by using the META engine.

Note: You should be aware that pre-assigning a large number of libraries can have an 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. △

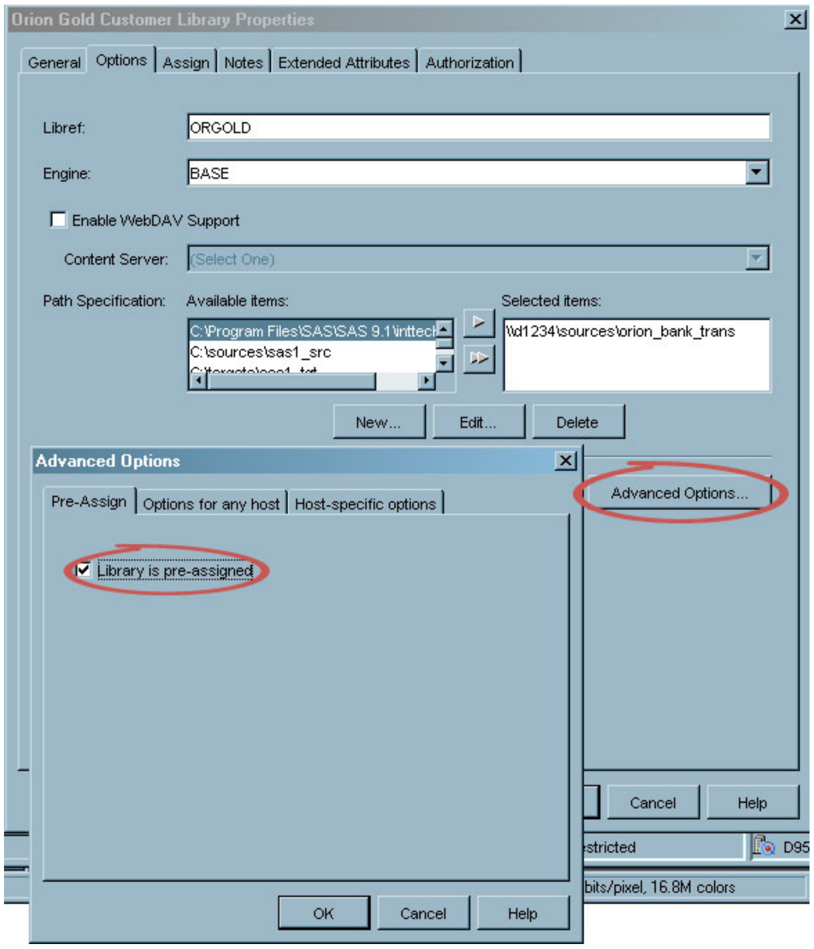
Pre-assigning a library to a non-MLE server requires the following steps:

1. Flag the library as already assigned by the server by selecting the **`Library is pre-assigned`** advanced option in SAS Management Console.
2. Set up the server process to retrieve library metadata by adding the METAAUTOINIT object server parameter option or the METAAUTORESOURCES SAS system option to the server's start-up definition.

Note: Library definitions (and therefore, table definitions) that are pre-assigned using the METAAUTORESOURCES option must exist in the same repository in order to support pre-assigned librefs. The code that pre-assigns the libraries on start-up of the server does not look down the repository dependency chain to find library definitions. This is a known limitation with pre-assigned libraries and the METAAUTORESOURCES option. Libraries must be defined in the same repository as the server context definition. △

Assume that we are pre-assigning the Orion Gold Customers library. The library can be set up to be assigned by the server process by either selecting the **`Library is pre-assigned`** advanced option when the library is being defined or by modifying the library's properties after the fact. When you select the **`Library is pre-assigned`** option, any server processes that have been configured to do so will connect into the metadata server on start-up and assign the library. The libraries can be assigned to SAS Application Servers (which may include a workspace server, a stored process server, and/or an OLAP server) or to a server defined outside an application server context, such as a SAS/SHARE server. To access the advanced options for the library, select the following in SAS Management Console: **Data Library Manager ▶ SAS Libraries ▶ Orion Gold Customers ▶ Properties**. Then, click the **`Options`** tab and click **`Advanced Options`**.

Display 3.2 Library is pre-assigned Option



Libref: ORGOLD
Engine: BASE
☐ Enable WebDAV Support
Content Server: (Select One)
Path Specification: Available items: C:\Program Files\SAS\SAS 9.1\inttech; C:\sources\sas1_src
Selected items: \\d1234\sources\orion_bank_trans
☑ Library is pre-assigned

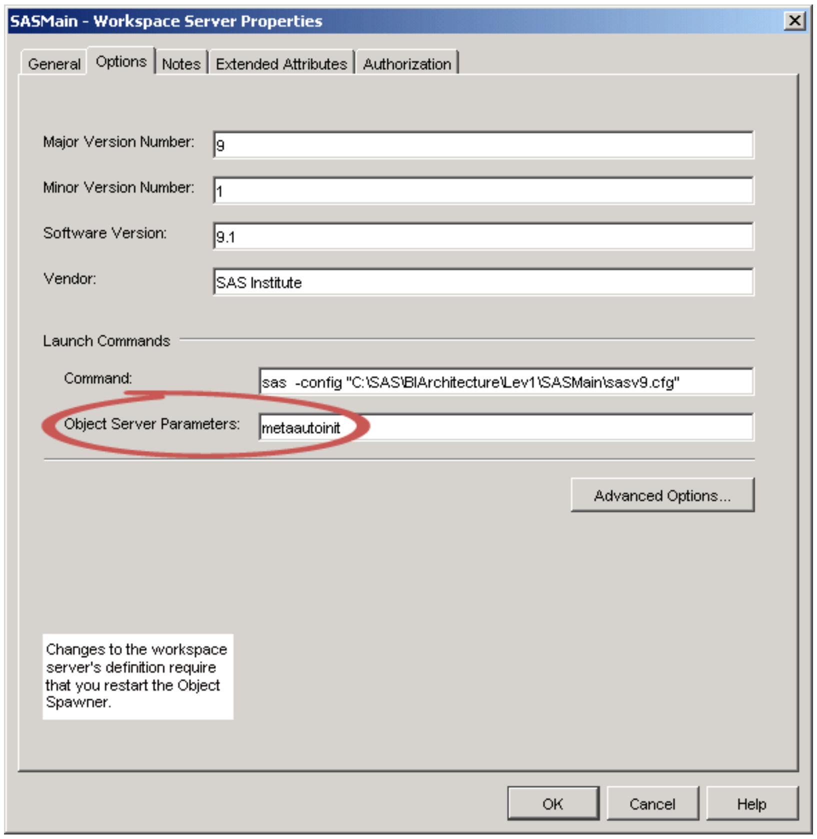
With the Orion Gold Customers library flagged to be assigned at start-up, the next step is to define which server processes should assign libraries. The option or parameter that you use to tell a server process to connect into the SAS Metadata Server during start-up depends on the type of server that is handling the request.

The server process must contain the METAAUTOINIT object server parameter in the server start-up definition for the following IOM (Integrated Object Model) servers: the workspace server, the stored process server, and the OLAP server. If the server process is any other SAS session, you use the METAAUTORESOURCES system option. (These servers include Data Step Batch Servers, the SAS Display Manager System, the SAS/CONNECT Server, and the SAS/SHARE Server.)

Pre-assignment Using Information Stored in the Metadata

The METAAUTOINIT option can be specified in the IOM server's definition through SAS Management Console or in the configuration file defined by the configuration option in the start-up command. The following example shows how you supply the METAAUTOINIT option in the **`Object Server Parameters`** field in the server's properties to servers that are started by the object spawner. Modifying the server's metadata definition requires that the object spawner be restarted in order for it to obtain the changes. To access the server's properties, select the following in SAS Management Console: **Server Manager** ▶ **SASMain** ▶ **Workspace Server** ▶ **Properties**. (Obviously, the path would be different for different servers).

Display 3.3 METAAUTOINIT OBJECTSERVERPARMS Option



Major Version Number: 9
Minor Version Number: 1
Software Version: 9.1
Vendor: SAS Institute
Command: sas -config "C:\SAS\BIArchitecture\Lev1\SASMain\sasv9.cfg"
Object Server Parameters: metaautoinit

Changes to the workspace server's definition require that you restart the Object Spawner.

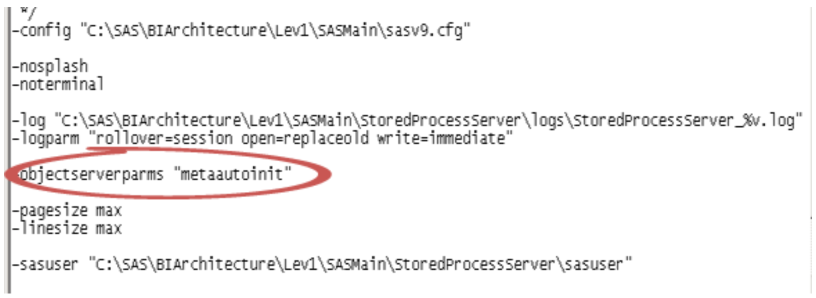
Note: Server processes that are currently running will not pick up the new configuration options until they are restarted. Therefore, load-balanced processes on stored process servers that use the default configuration of staying open once started will need to be shut down. The same is true for pooled workspace servers. The object spawner will restart them as needed.

The METAAUTOINIT OBJECTSERVERPARMS option could be referenced in a configuration file called by the workspace server instead of being supplied in the server's definition. However, because the default configuration file called by the workspace server (**sasv9.cfg**) is also used by non-IOM servers executing on the host, it is recommended that you create a custom configuration file and place it in the workspace server directory, which is *SAS-config-dir*`\Lev1\SASMain\WorkspaceServer`. Use this file to store the METAAUTOINIT option. Options specified in a configuration file take precedence over conflicting options in the server's definition. The following custom configuration files are created during installation for the other IOM servers:

- Metadata server: *SAS-config-dir*`\Lev1\SASMain\MetadataServer\sasv9_MetadataServer.cfg`
- OLAP server: *SAS-config-dir*`\Lev1\SASMain\OLAPServer\sasv9_OLAPServer.cfg`
- Stored Process server: *SAS-config-dir*`\Lev1\SASMain\StoredProcessServer\sasv9_StorProcSrv.cfg`

△

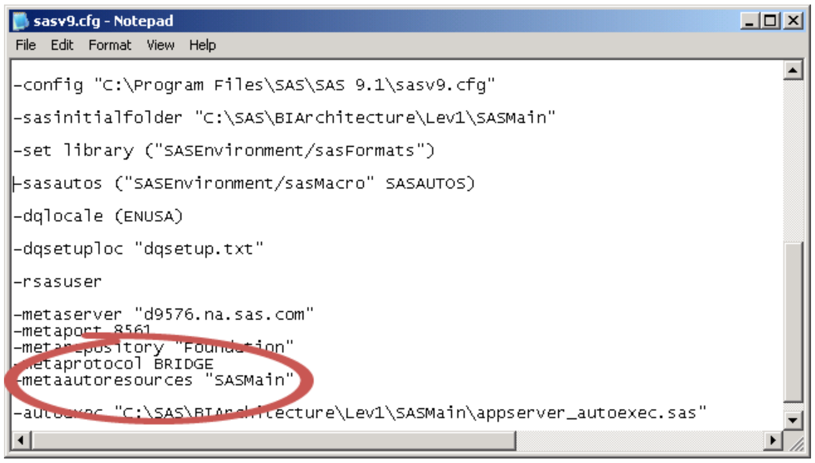
Display 3.4 Options Specified in a Configuration File



```
 */
-config "C:\SAS\BIArchitecture\Lev1\SASMain\sasv9.cfg"

-nosplash
-noterminal

-log "C:\SAS\BIArchitecture\Lev1\SASMain\StoredProcessServer\logs\StoredProcessServer_%v.log"
-logparm "rollover=session open=replaceold write=immediate"

-objectserverparms "metaautoinit"

-pagesize max
-linesize max

-sasuser "C:\SAS\BIArchitecture\Lev1\SASMain\StoredProcessServer\sasuser"
```

SAS servers that are not IOM servers connect into the metadata server to retrieve definitions through the use of META* system options, which are typically specified in the configuration file. These options provide a SAS session with metadata information such as the name of the metadata server host, its port number, and optionally the metadata identity, to use for authentication decisions in the metadata repository. In addition, the METAUTORESOURCES option tells the SAS session which specific metadata object it should retrieve in order to determine libraries to assign at start-up. The following example shows how the METAAUTORESOURCES option can be used to reference a specific SAS Application Server in the configuration file for the Data Step Batch Server or SAS Display Manager session, causing it to retrieve pre-assigned libraries that are associated with the SASMain server context. The configuration file is located in *SAS-config-dir*`\Lev1\SASMain\sasv9.cfg`.

Display 3.5 Metaautoresources Option



```
-config "C:\Program Files\SAS\SAS 9.1\sasv9.cfg"

-sasinitialfolder "C:\SAS\BIArchitecture\Lev1\SASMain"

-set library ("SASEnvironment/sasFormats")

-sasautos ("SASEnvironment/sasMacro" SASAUTOS)

-dqlocale (ENUSA)

-dqsetuploc "dqsetup.txt"

-rsasuser

-metaserver "d9576.na.sas.com"
-metaport 8561
-metarepository "Foundation"
-metaprotocol BRIDGE
-metaautoresources "SASMain"

-autoexec "C:\SAS\BIArchitecture\Lev1\SASMain\appserver_autoexec.sas"
```

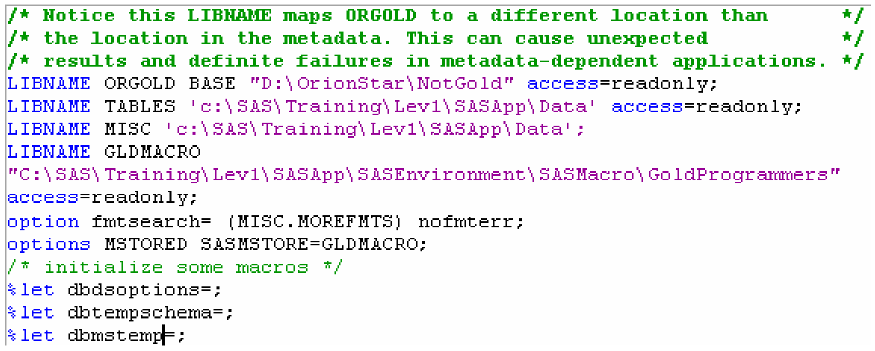
The METAAUTORESOURCES system option is set differently for a SAS/SHARE server. Recall that a SAS/SHARE server is not defined within a server context, such as SASMain, for example. Because it is not defined within a server context, the SAS/SHARE server will not use the METAAUTORESOURCES option specified in the previous example. To tell the SAS/SHARE server to connect into the metadata and retrieve pre-assigned metadata-based libraries, you must add the specific SAS/SHARE server component that it should retrieve to its configuration file in the following path: *SAS-config-dir*`\Lev1\SASMain\ShareServer`. To do this, add a line similar to the following to the `sasv9.cfg` file:

```
-metaautoresources "omsobj:ServerComponent?@Name=SASMain - SAS/SHARE Server"
```

Pre-assignment Using Information in an Autoexec File

Recall that a SAS autoexec file is a text file that contains SAS statements that are executed upon start-up of the server process. If an autoexec file is being used in your environment, it is important to note that libraries assigned by an autoexec file take precedence over same-named libraries assigned by the server via METAAUTORESOURCES or METAAUTOINIT. (Use the autoexec file created during installation, which is *SAS-config-dir*`\Lev1\SASMain\appserver_autoexec.sas`.) For example, if ORGOLD is defined in the metadata to be server-assigned, and ORGOLD is defined in a LIBNAME statement in an autoexec file defined to the server, the ORGOLD library will be assigned using the information in the autoexec file. Simply put, the library assignment in the autoexec file always wins.

Display 3.6 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 applications. */
LIBNAME ORGOLD BASE "D:\OrionStar\NotGold" access=readonly;
LIBNAME TABLES 'c:\SAS\Training\Lev1\SASApp\Data' access=readonly;
LIBNAME MISC 'c:\SAS\Training\Lev1\SASApp\Data';
LIBNAME GLDMACRO
"C:\SAS\Training\Lev1\SASApp\SASEnvironment\SASMacro\GoldProgrammers"
access=readonly;
option fmtsearch= (MISC.MOREFMTS) nofmterr;
options MSTORED SASMSTORE=GLDMACRO;
/* initialize some macros */
%let dbdsoptions=;
%let dbtempschema=;
%let dbmstemp=;
```

Pre-assigning Libraries to Use the MLE

The MLE 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 META engine acts as a gatekeeper that determines which users can access which metadata-based libraries and tables.

To define a library that uses the MLE, perform the following steps:

1 Define metadata for 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.

4 Add the metadata LIBNAME statement to an autoexec file. During the configuration process, the Configuration Wizard created a file named **appserver_autoexec.sas** and placed it in the same directory as the SAS Intelligence Platform's configuration file, **sasv9.cfg**. In platform implementations that do not include a SAS Solution, such as Marketing Automation, this file is typically empty. The purpose of this text file is to serve as the location where SAS solutions along with administrators like you can place SAS statements that need to be executed immediately after the SAS server process initializes and before any user input is accepted. This is considered the SAS Intelligence Platform's autoexec file. Add the LIBNAME statement created in Step 3 to this file as shown.

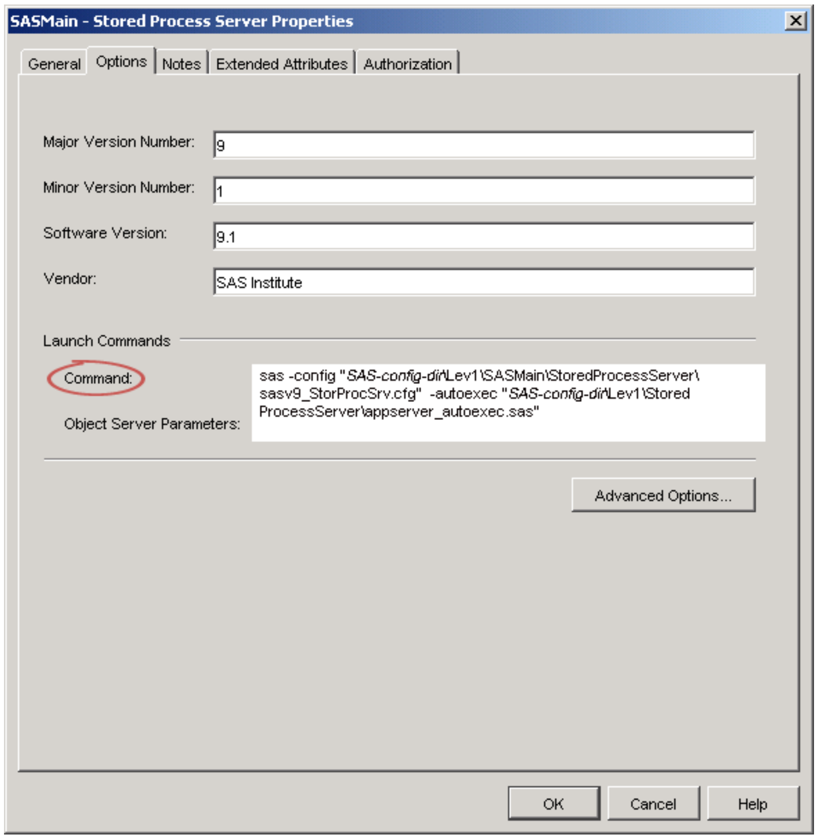
5 Add the autoexec file to the start-up definition of any server that should assign the library using the META engine. There are a variety of ways to tell a SAS server process that it should use the autoexec file created in the previous step. Here are the two approaches you've already seen in this chapter:

   □ Call the autoexec file from the server process's configuration file (ideally a shared configuration file). Find the **sasv9.cfg** file and enter text similar to the following:

   ```
   -autoexec 'SAS-config-dir\Lev1\SASMain\appserver_autoexec.sas'
   ```

   □ Add the autoexec system option to each server's metadata object definition. Note that this option is only available to IOM servers.

Display 3.7 Adding the Autoexec System Option



Major Version Number: 9

Minor Version Number: 1

Software Version: 9.1

Vendor: SAS Institute

Command:

Object Server Parameters:

```
sas -config "SAS-config-dir\Lev1\SASMain\StoredProcessServer\
sasv9_StorProcSrv.cfg" -autoexec "SAS-config-dir\Lev1\Stored
ProcessServer\appserver_autoexec.sas"
```

6 Restart the object spawner and any server processes whose autoexec files have been modified. The SAS/SHARE server must be restarted in order to pick up changes to the configuration file. On Windows it can be restarted using the **`NET STOP`** and **`NET START`** commands at a command prompt on the SAS/SHARE server's host, or through the Windows Service Manager. For more information about restarting the servers, see "Starting, Stopping, and Pausing Servers" in the *SAS Intelligence Platform: System Administration Guide*.

Note: We recommended that you treat libraries mapped with the META engine as read-only and do not allow any users Create access to the library or its tables. △

Verifying Pre-assignments by Reviewing the Logs

After you specify that a library is to be assigned by the server (by specifying the METAAUTOINIT or METAAUTORESOURCES system option), the SAS server process will start up as follows:

1 Connect to the metadata server.

2 Retrieve library metadata.

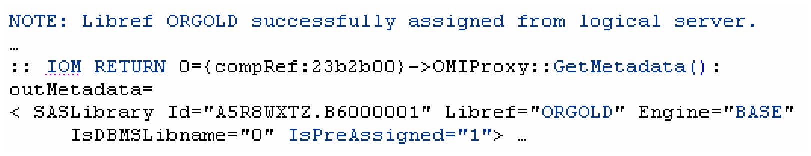
3 Assign the library using the engine specified in the library definition.

For example, if the Orion Gold Customers library is assigned by 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" repname=Foundation;
```

In the case of an IOM server (using METAAUTOINIT), 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 3.8 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 would be written to its log file:

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

See *SAS Intelligence Platform: Administration Guide* for information about setting logging levels. see "Configure the Logging of XML Information" in the *SAS Intelligence Platform: System Administration Guide*.To verify pre-assignment, you should set the logging level to **1**.

CHAPTER 4

Managing Table Metadata

Overview of Managing Table Metadata

As explained in “Verifying Access to Tables” on page 42, one way to create metadata for the tables in a library is to use the Import 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 table definitions that do not reflect the current structure of the tables that they represent.

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 the *SAS Open Metadata Interface: Reference*. △

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`** checkbox. Do not use the specialized “Pre-Assigned Library” template. △

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 define a SAS library, it has no related table metadata. You can add this metadata by using the Import Tables wizard in SAS Management Console (see “Overview of Managing Table Metadata” on page 57), or by using PROC METALIB, as shown in the example below.

Before you can successfully run this PROC METALIB code, you must have Create, ReadMetadata, and WriteMetadata access to SASLibrary metadata object. To check your permissions, use the Authorization Manager in the SAS Management Console. (See “Managing Permissions” in the chapter “Managing Authorizations” in the *SAS Management Console: User’s Guide*.)

Example: Creating Table Metadata

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=&mlibname repname= &mrepname );
  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 had the NOEXEC statement 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.

Example: Using the NOEXEC and REPORT Statements

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

```
ods html "myfile";
proc metalib;
omr (library=&mlibname repname= &mrepname );
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 62. △

Here is the resulting SAS log.

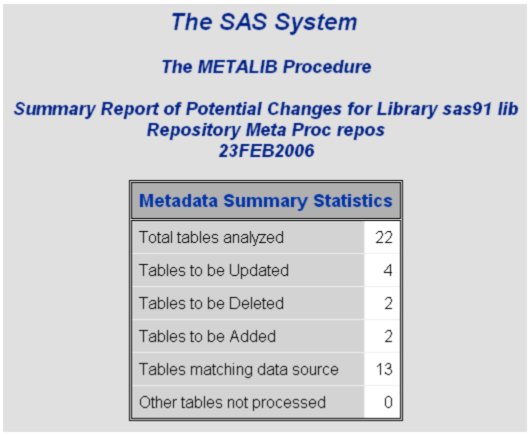
```
55   proc metalib;
56   omr (library=&mlibname  repname= &mrepname );
57   update_rule=(delete);
58   noexec;
59   report;
60   run;

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.

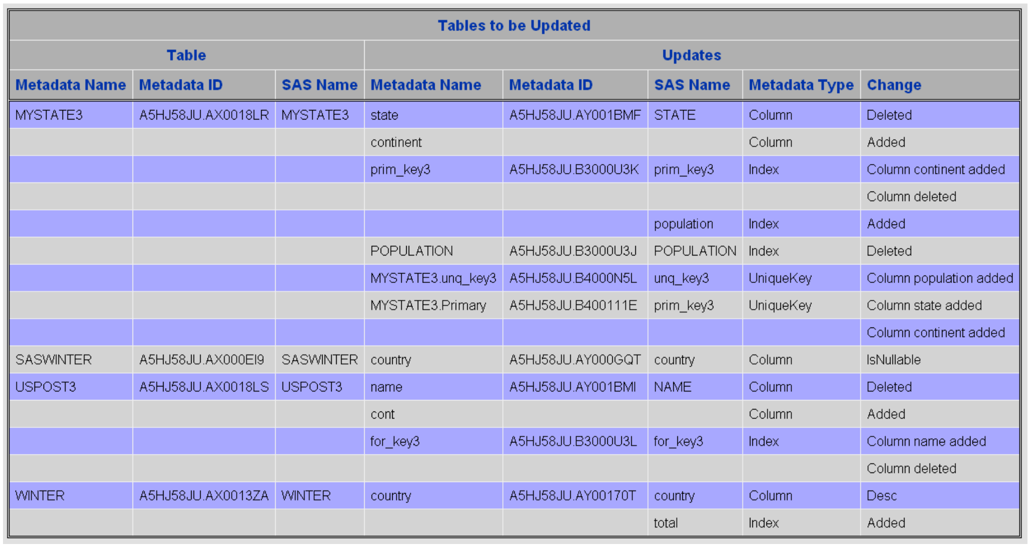


The SAS System

The METALIB Procedure

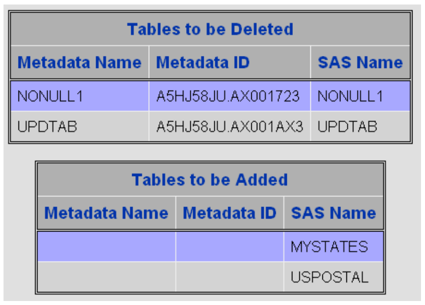
Summary Report of Potential Changes for Library sas91 lib
Repository Meta Proc repos
23FEB2006

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 59, the summary does not mention any deleted tables.

```
proc metalib;
  omr  (library=&mlibname repname= &mrepname );
run;
```

Here is the resulting SAS log.

```
85   proc metalib;
86   omr (library=&mlibname repname= &mrepname );
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 PROC METALIB's default behavior. The principal rules that you can specify are shown below:

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 Data Integration Studio.)

```
proc metalib;
omr (library=&mlibname repname= &mrepname );
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=&mlibname repname= &mrepname );
  update_rule (delete noadd noupdate);
  report;
run;
```

The resulting SAS output 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
```

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 select tables or table definitions for processing:

- For tables, specify their SAS name. If no table definition is found, it is created in the repository that contains the SASLibrary object. If a matching table definition is found, it is compared to the physical table. If differences are found, the table definition is updated.
- For table definitions, 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.

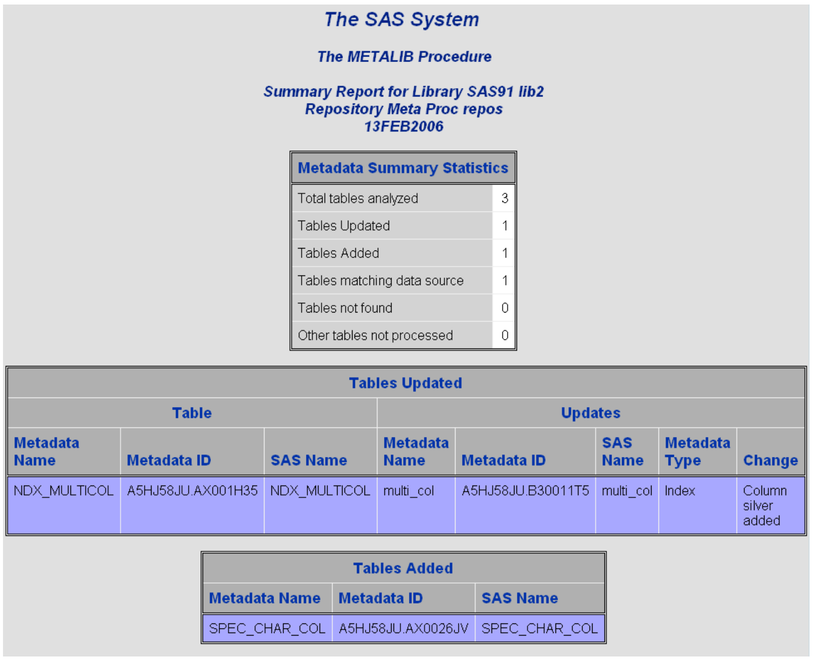
You can use EXCLUDE to specify a single table or a list of tables to exclude from processing.

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=&mlibname repname= &mrepname );
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=&mlibname repname= &mrepname);
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 66
- “Indexing Data” on page 67
- “Sorting Data” on page 69
- “Buffering Data” on page 70
- “Using Threaded Reads” on page 72
- “Validating SPD Engine Hardware Configuration” on page 72
- “Setting LIBNAME Options That Affect Performance” on page 73
- “Grid Computing Data Considerations” on page 79

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, while 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.

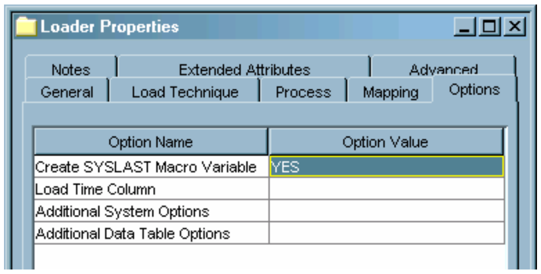
These are the types of compression that you can specify:

- 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 **`Additional System Options`** field to set the COMPRESS= system option on a loader transformation. (A loader transformation generates or retrieves code that puts data into a specified target.)

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



Option Name	Option Value
Create SYSLAST Macro Variable	YES
Load Time Column	
Additional System Options	
Additional Data Table Options	

- all tables for a particular SAS data library. For example, when you define 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 73). 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 74).

  Note: You cannot specify compression for a 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`**.

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

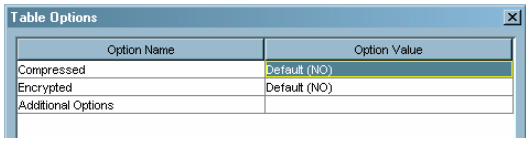


Table Options

Option Name	Option Value
Compressed	Default (NO)
Encrypted	Default (NO)
Additional Options	

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 on 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 (465-33-8613), 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=465-33-8613 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.

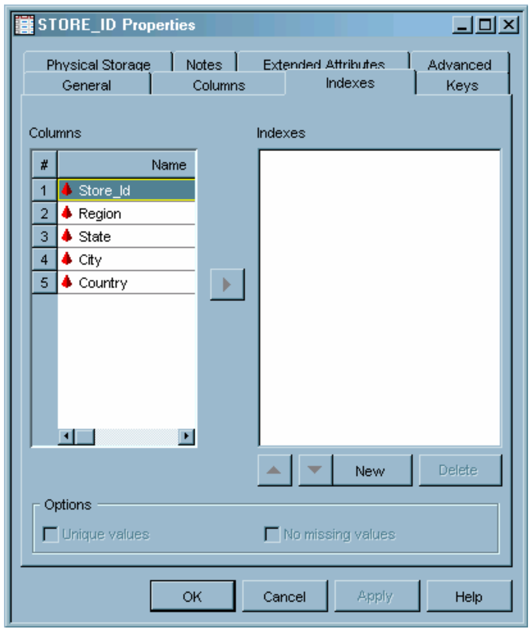
- a composite index, which consists of the values of more than one column, with the values concatenated to form a single value.

For each indexed column, you also can 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 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 also can specify **`Unique values`** and **`No missing values`**.

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



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.

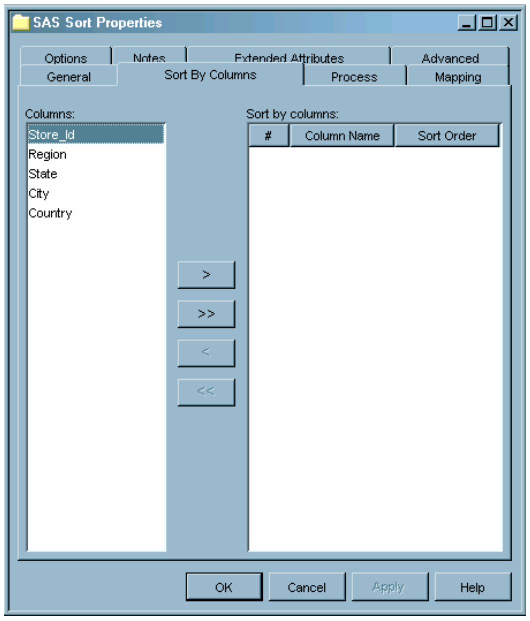
For more information about creating indexes, see *SAS Language Reference: Concepts*.

Sorting Data

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:

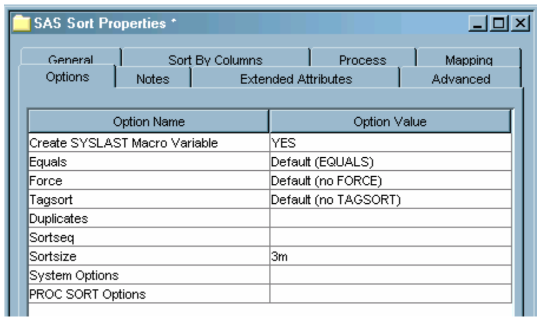
- using the SAS SORT procedure.
- setting properties for a SAS sort template in SAS Data Integration Studio.

Display 5.4 The Sort By Columns Tab in the SAS Sort Properties Dialog Box



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.

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



Option Name	Option Value
Create SYSLAST Macro Variable	YES
Equals	Default (EQUALS)
Force	Default (no FORCE)
Tagsort	Default (no TAGSORT)
Duplicates	
Sortseq	
Sortsize	3m
System Options	
PROC SORT Options	

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 a SPD Engine library in the metadata. See “Setting LIBNAME Options That Affect Performance of SPD Engine Tables” on page 77.

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 the section on "Support for Parallel Processing" in *SAS Language Reference: Concepts*.

The performance of the multi-threaded sort will be 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 on 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 will perform the sort. If the DBMS performs the sort, then the configuration and characteristics of the DBMS sorting program will 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/NOSORTEQUALS system option and EQUALS/NOEQUALS procedure option.

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

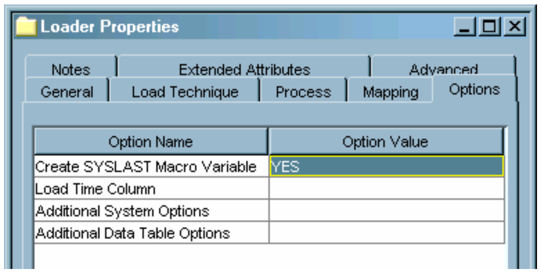
For Base SAS tables and some relational database tables, you can adjust page buffer settings to optimize CPU and I/O use. Different options are used for each type of table.

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 **`Additional System Options`** in the properties window for a loader transformation.



Option Name	Option Value
Create SYSLAST Macro Variable	YES
Load Time Column	
Additional System Options	
Additional Data Table Options	

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

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 74) or on the individual table.

- □ 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*.

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*. △

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 Creates threads, which are standard operating system tasks that are controlled by SAS, within the SAS session.
2 Establishes a DBMS connection on each thread.
3 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 74.

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`**.

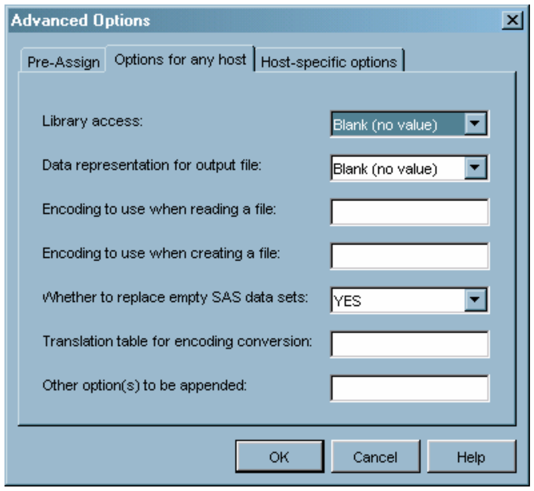
Setting LIBNAME Options That Affect Performance

When you use SAS Management Console to define a library, there are options available for the library definition that correspond to the LIBNAME options for the selected engine. Some of those options can be used to optimize use of the tables within the libraries.

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.

Display 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 also can set the size of each cache (CACHESIZE= option).

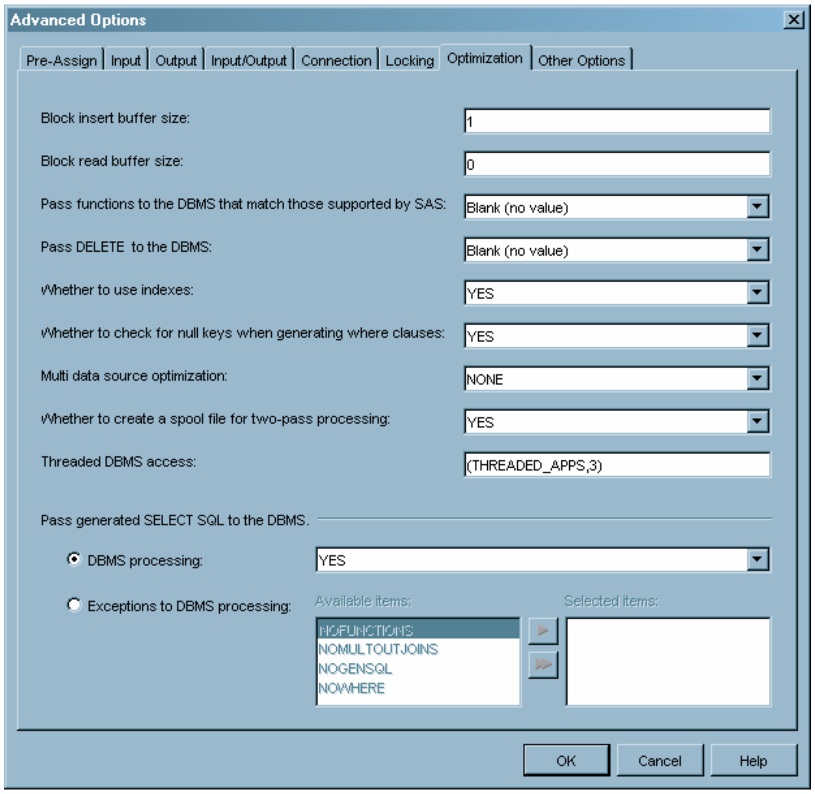
The **`Other option(s) to be appended`** field can be used to specify LIBNAME options such as COMPRESS= (see "Compressing Data" on page 66).

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. For example, here are the **`Optimization`** tab default settings for DB2 libraries for UNIX and PC.

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



Block insert buffer size: 1
Block read buffer size: 0
Pass functions to the DBMS that match those supported by SAS: Blank (no value)
Pass DELETE to the DBMS: Blank (no value)
Whether to use indexes: YES
Whether to check for null keys when generating where clauses: YES
Multi data source optimization: NONE
Whether to create a spool file for two-pass processing: YES
Threaded DBMS access: (THREADED_APPS,3)

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. Here are descriptions 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” on page 70.
Block read buffer size (READBUFF=)	specifies the number of rows of DBMS data to read into the buffer. See “Buffering Data” on page 70.
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/timestamp handling. Exercise care when using this option.
Pass DELETE to the DBMS (DIRECT_EXE=)	specifies that a 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 67.

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 72.

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.

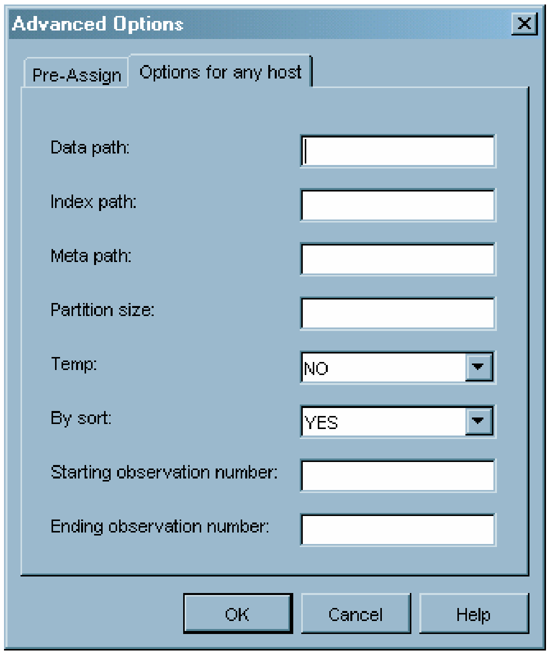
The **`Other Options`** tab, which is available for all database management systems, can be used to specify LIBNAME options such as COMPRESS= (see "Compressing Data" on page 66).

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 a 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.

Display 5.8 The Options for Any Host Tab in the Advanced Options Dialog Box for a SPD Engine Library



Data path:	
Index path:	
Meta path:	
Partition size:	
Temp:	NO
By sort:	YES
Starting observation number:	
Ending observation number:	

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 may 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 since the loss of 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

- □ 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 information about table and other system options for the SPD Engine, see **`http://support.sas.com/rnd/scalability/spde/syntax.html`**. For 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. Therefore, the data must either be distributed to the nodes prior to running the application or— much more commonly—made available through shared network libraries. Storage on local nodes is discouraged. 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.

The parallel data load is monitored throughout.

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 prior to 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.

About OLAP Schemas

OLAP schemas are lists of cubes that are grouped together so that they can be exclusively 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.

The initial installation of the SAS OLAP Server software creates an OLAP schema named SAS Main — OLAP Schema. By default, all SAS OLAP Servers are assigned to the single OLAP schema. The OLAP schema lists all cubes as they are built.

To assign cubes to specific servers you create new OLAP schemas. This might be necessary if you have multiple large cubes, in which 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 required to reside in the same repository as the SAS OLAP Servers that use those schemas.

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 82.

OLAP schemas are read from metadata only at the start-up of SAS OLAP Servers. Assigning a new OLAP schema requires a restart of 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. In batch mode, the OLAP schema is specified in PROC OLAP in the OLAP_SCHEMA= option of the METASVR statement.

OLAP schemas help determine security inheritance. Permissions that are set on SAS OLAP Servers are inherited by their respective OLAP schemas. Schema permissions are inherited by the cubes that are listed in the schema. Cube permissions are inherited by their dimensions and measures. Dimension permissions are inherited by their levels and hierarchies.

Create or Assign an OLAP Schema

Perform these steps to create a new OLAP schema or assign an OLAP schema to a SAS OLAP Server:

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 **`SASMain`**, for example.

4 Right-click the top-level SAS Application Server and select **`Properties`**.

5 In the Properties window, click `New` to create a new OLAP schema, or click the down arrow to select an existing OLAP schema.

6 Click `OK` to save changes and close the Properties window.

7 Restart the SAS OLAP Server using the SAS OLAP Server Monitor.

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(s) that will be used to build your cubes and what dimensions and measures will be created.
- Define the table(s) 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 define, in the metadata, the server that will be used to access the tables. This is a SAS Application Server with a workspace server component.
  - Use SAS Management Console to define, in the metadata, the SAS library that contains the table.
  - In SAS OLAP Cube Studio, specify the server that will be used to 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, etc.), 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 SAS Library that contains the tables that you want to define, and then select **`Next`**.
        - Select the tables to define, and then select **`Next`**.
        - Select **`Finish`**. The table definitions are loaded into the metadata.

    - If you start to create a cube and do not see the table that you need to continue, then you can select 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. 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 further 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 SAS 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 SAS data set 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: When you save a cube in c:\olapcubes and name the cube Campaigns, the cube is saved in the directory c:\olapcubes\CAMPAIGNS. For further information about preliminary setup and configuration steps see the *SAS OLAP Server Administrator’s Guide.*

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.

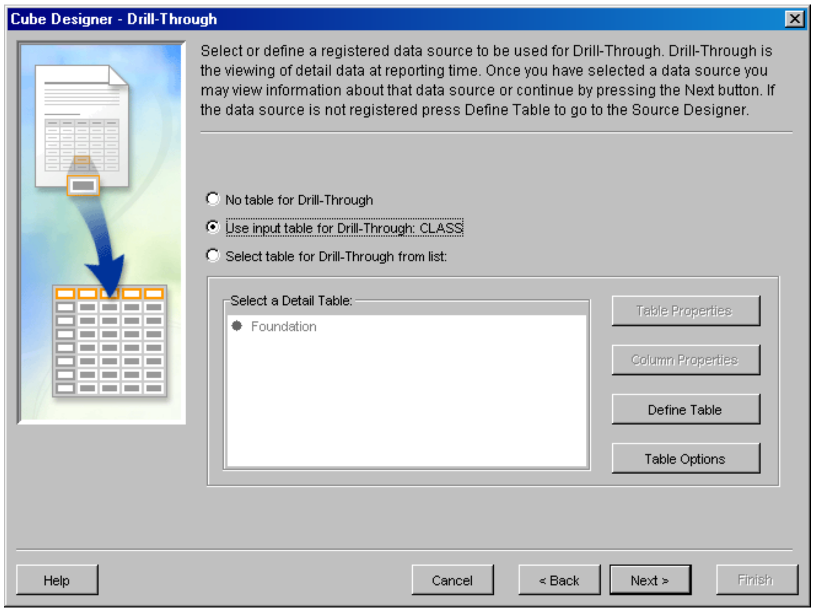
Making Detail Data Available for Drill-Through

You can drill through an OLAP report to the underlying detail data only after you make the detail data available to the cube, its SAS OLAP Server, and the information map that you use for creating the report.

Making Detail Data Available to a 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, select one the following radio buttons:
  - ☐ **`Use input table for Drill-Through`** (to use the table that you selected on the Input page of the wizard).



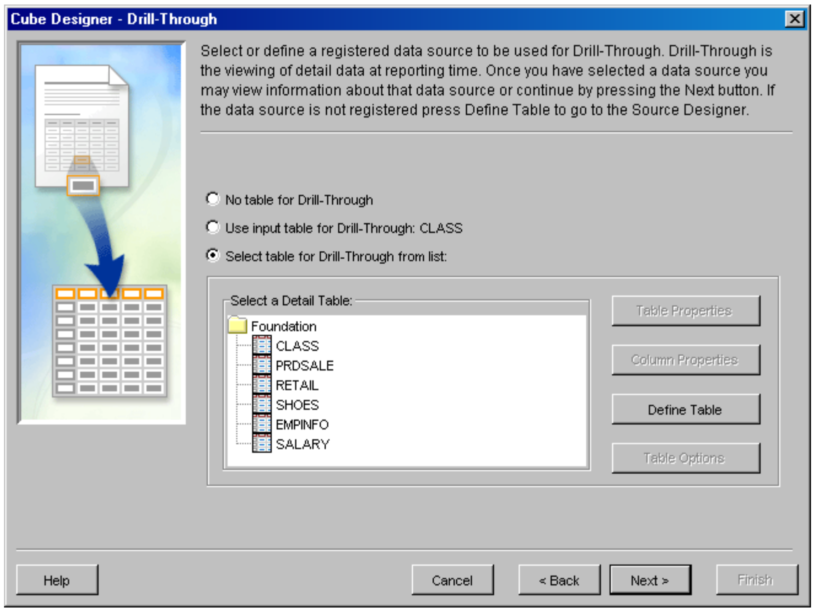
Cube Designer - Drill-Through

Select or define a registered data source to be used for Drill-Through. Drill-Through is the viewing of detail data at reporting time. Once you have selected a data source you may view information about that data source or continue by pressing the Next button. If the data source is not registered press Define Table to go to the Source Designer.

○ No table for Drill-Through
● Use input table for Drill-Through: CLASS
○ Select table for Drill-Through from list:

Select a Detail Table:
Foundation

  - ☐ **`Select table for Drill-Through from list`** (to select a table from the list of tables on the Drill-Through page. If only a repository folder is listed, then select the repository folder in order to see the tables).


Cube Designer - Drill-Through

Select or define a registered data source to be used for Drill-Through. Drill-Through is the viewing of detail data at reporting time. Once you have selected a data source you may view information about that data source or continue by pressing the Next button. If the data source is not registered press Define Table to go to the Source Designer.

No table for Drill-Through
Use input table for Drill-Through: CLASS
[✓ Select table for Drill-Through from list:]

Select a Detail Table:
Foundation
CLASS
PRDSALE
RETAIL
SHOES
EMPINFO
SALARY

For more information about the Cube Designer wizard, see the SAS OLAP Cube Studio Help.

- 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*.

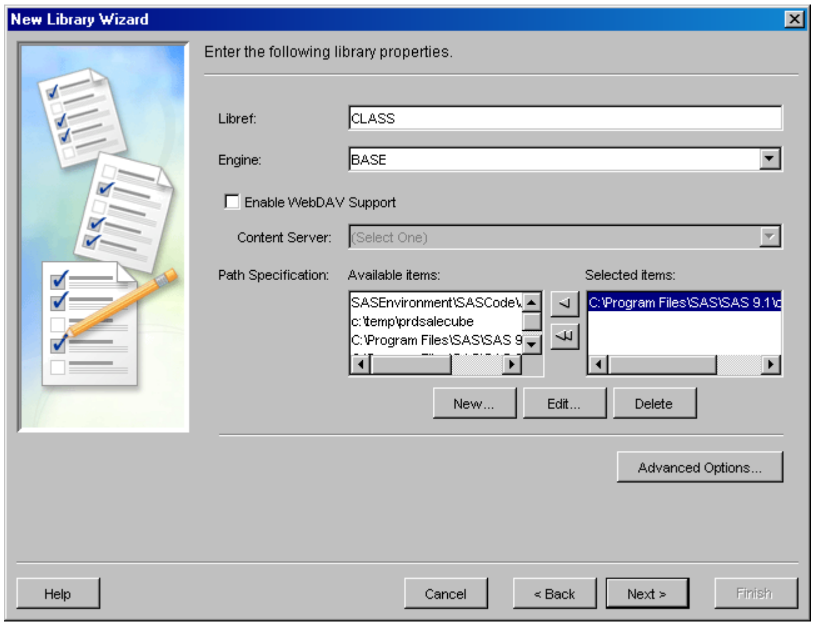
Making Detail Data Available to an OLAP Server

In order for the OLAP server to make detail data available for a cube, the SAS library for the table that contains the detail data must be defined so that the OLAP server can access it. The simplest way to define the library to the server is to pre-assign it in the metadata repository.

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

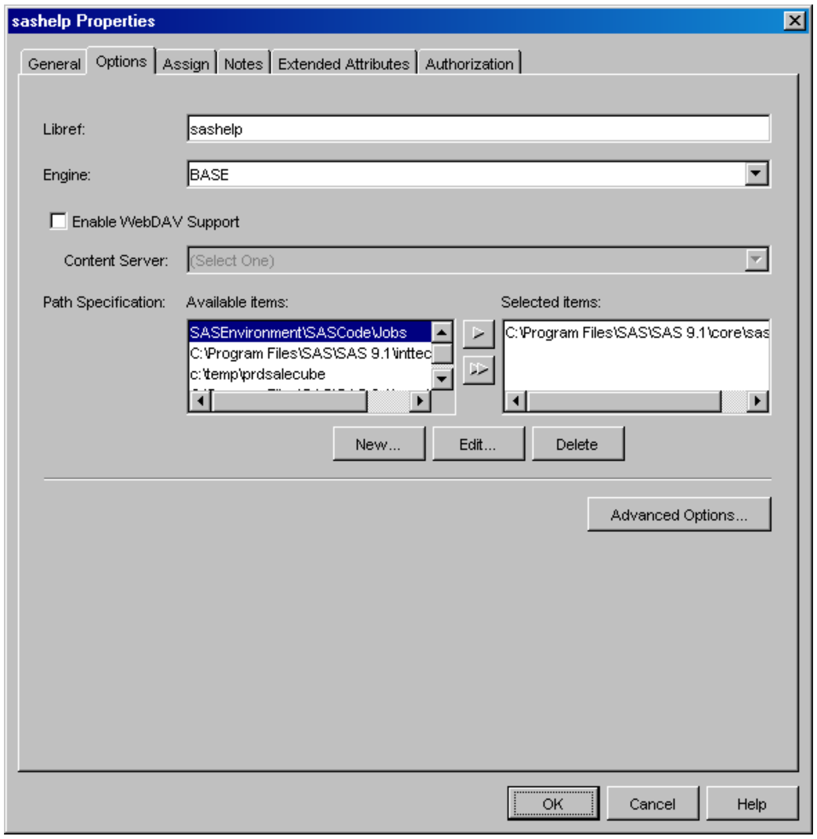
1 In Data Library Manager (in SAS Management Console), find the **SAS Libraries** folder and perform one of the following tasks to get to the dialog box that lets you select advanced options:

   - For a new library, right-click the **SAS Libraries** folder and select **New Library** to start the New Library Wizard. Then navigate to the wizard page that enables you to specify library options such as the libref.



Libref: CLASS
Engine: BASE
Available items: SASEnvironment\SASCode\, c:\temp\prdsalecube, C:\Program Files\SAS\SAS 9
Selected items: C:\Program Files\SAS\SAS 9.1\c

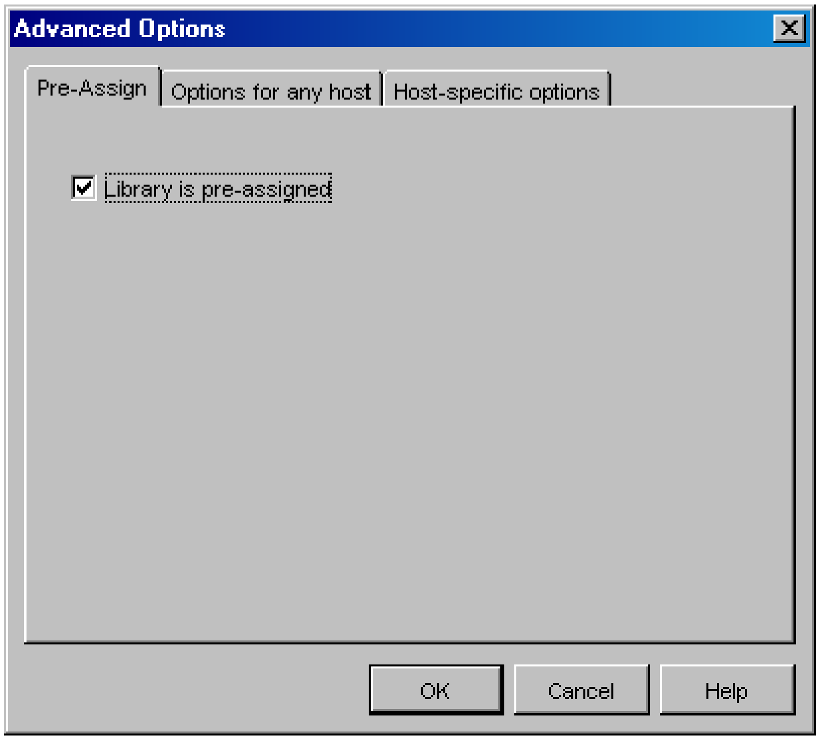
□ For an existing library, open the **SAS Libraries** folder and right-click the desired library. Select **Properties** from the drop-down menu, and then select the **Options** tab in the properties dialog box.



Libref: sashelp
Engine: BASE
Available items: SASEnvironment\SASCode\Jobs, C:\Program Files\SAS\SAS 9.1\inttec, c:\temp\prdsalecube
Selected items: C:\Program Files\SAS\SAS 9.1\core\sas

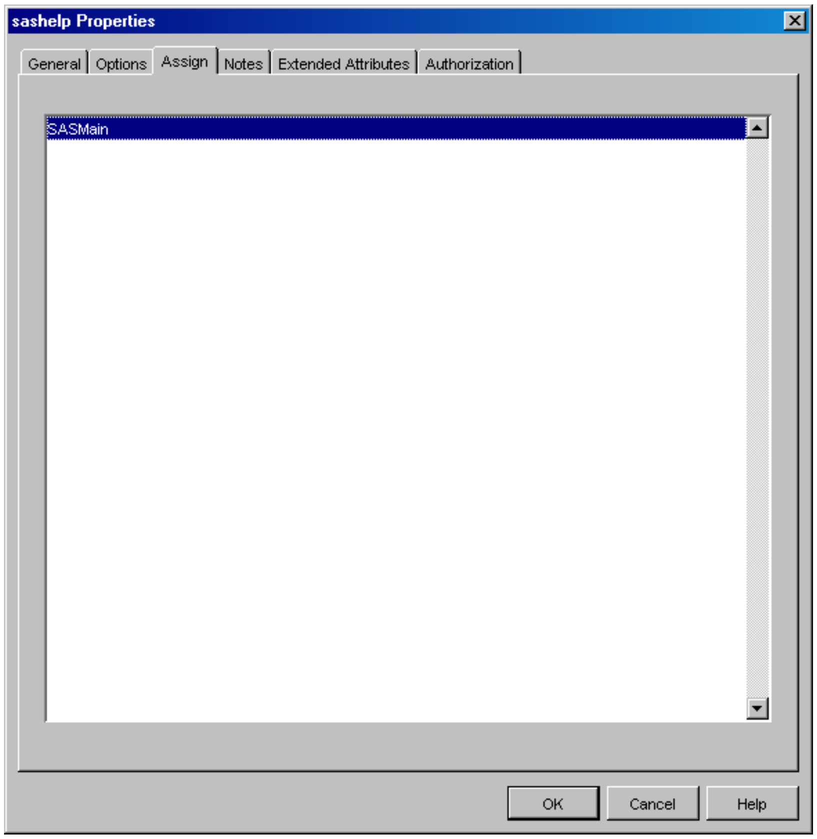
2 Click **Advanced Options**.

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



☑ Library is pre-assigned

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.



SASMain

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

6 Restart the OLAP server.

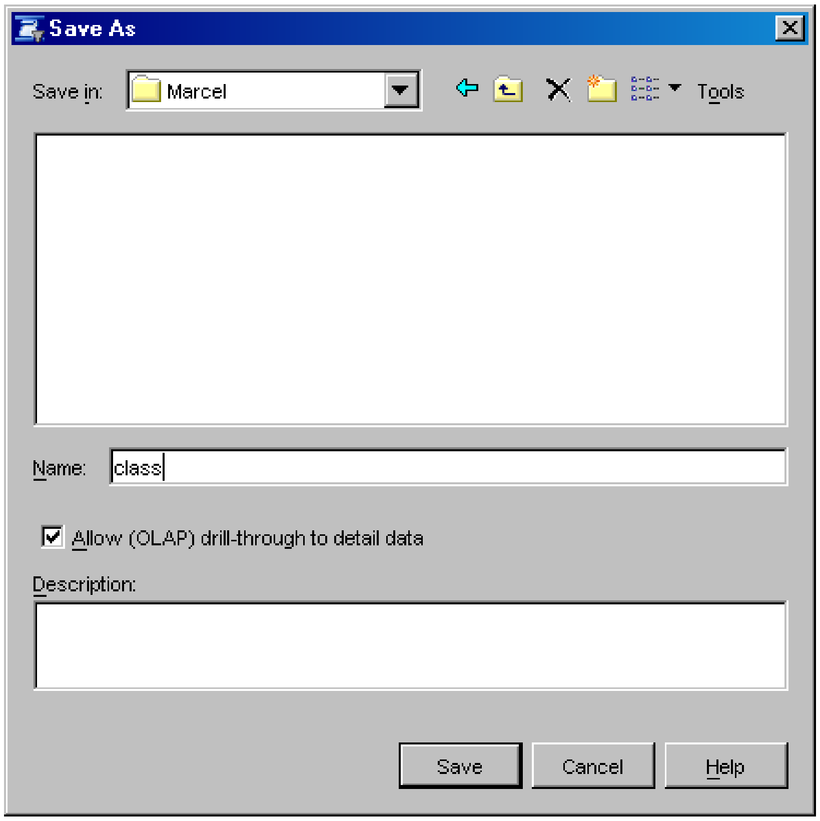
Note: If you want to enable users to view column labels when they drill through to detail data, then before you restart the server, perform the steps in "Make the Column Labels of Drill-Through Tables Available" on page 90. △

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.

Making Detail Data Available to an 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. You can set this option in one of two ways:

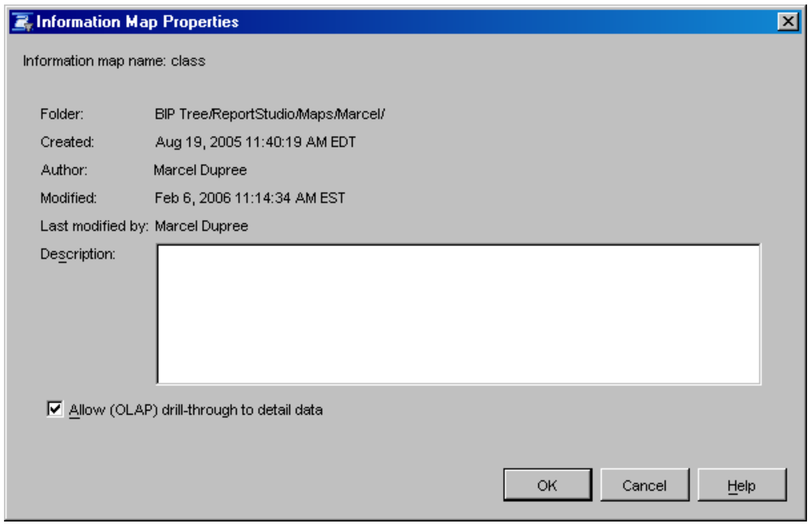
- When you save a new information map, select the **Allow (OLAP) drill-through to detail data** check box in the Save As dialog box before you save the information map.



Name: class

☑ Allow (OLAP) drill-through to detail data

- For an existing information map, open the information map, right-click it, and then select **Properties** from its drop-down menu. Select the **Allow (OLAP) drill-through to detail data** check box on the **Definition** tab in the Information Map Properties dialog box.



Information map name: class

Folder: BIP Tree/ReportStudio/Maps/Marcel/
Created: Aug 19, 2005 11:40:19 AM EDT
Author: Marcel Dupree
Modified: Feb 6, 2006 11:14:34 AM EST
Last modified by: Marcel Dupree
Description:

☑ Allow (OLAP) drill-through to detail data

Make the Column Labels of Drill-Through Tables Available

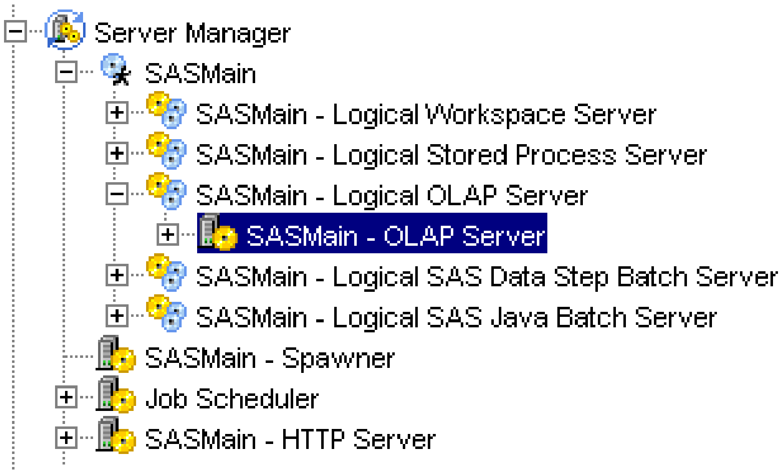
If you want to view column labels when you drill through to detail data for a cube, then you must set an option in Server Manager (in SAS Management Console) to make the column labels available.

Note: This option is set for the physical OLAP server, so the setting applies to all the cubes that are assigned to that server. △

Before you can make column labels available, you must first ensure that SAS 9.1.3 Service Pack 4 is installed and then upgrade the metadata for any existing metadata repositories. For more information, see "Upgrading Repository Metadata" in the Metadata Manager Help in SAS Management Console.

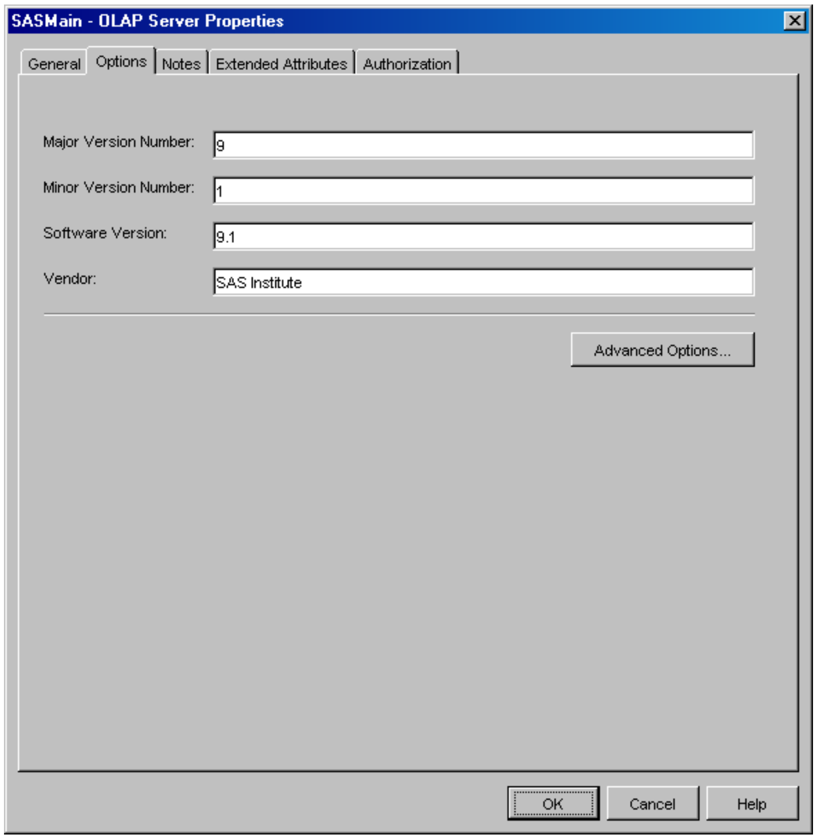
After the metadata is upgraded, perform the following steps to make the column labels of drill-through tables available:

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


Server Manager
SASMain
SASMain - Logical Workspace Server
SASMain - Logical Stored Process Server
SASMain - Logical OLAP Server
SASMain - OLAP Server
SASMain - Logical SAS Data Step Batch Server
SASMain - Logical SAS Java Batch Server
SASMain - Spawner
Job Scheduler
SASMain - HTTP Server

2 Select the server, and then select **File ▶ Properties** from the menu bar.

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



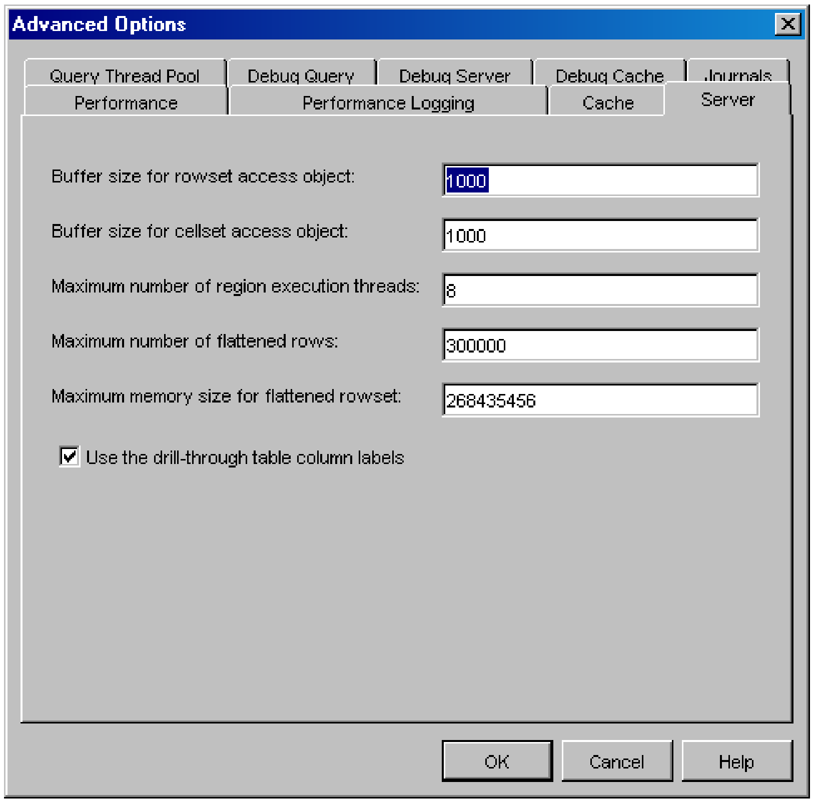
Major Version Number: 9

Minor Version Number: 1

Software Version: 9.1

Vendor: SAS Institute

4 In the Advanced Options dialog box, select the **`Server`** tab, and then select the **`Use the drill-through table column labels`** check box.



Buffer size for rowset access object: 1000

Buffer size for cellset access object: 1000

Maximum number of region execution threads: 8

Maximum number of flattened rows: 300000

Maximum memory size for flattened rowset: 268435456

☑ Use the drill-through table column labels

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

6 Restart the OLAP server.

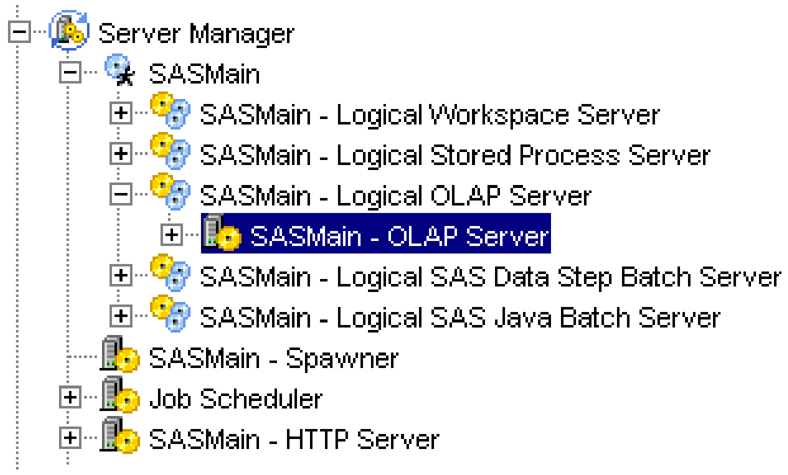
Display Detail Data for a Large Cube

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.

For information about increasing the heap size, see “SAS Services Application Heap Size” in *SAS Intelligence Platform: Web Application Administration Guide*.

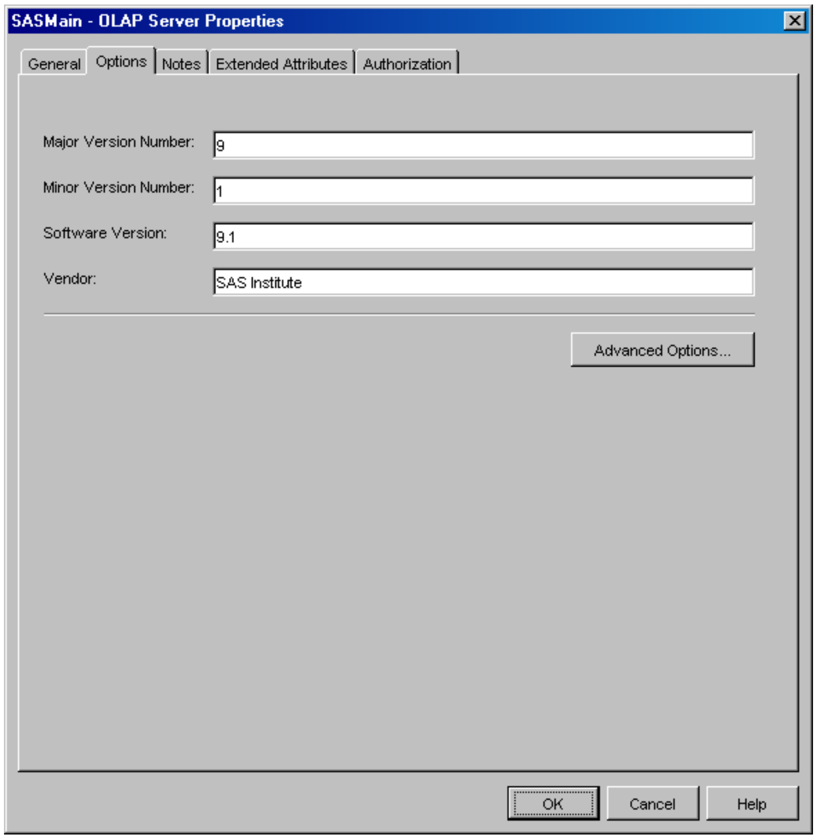
To increase the number of drill-through rows that your OLAP server can handle, you can change an OLAP server definition setting in Server Manager (in SAS Management Console) by completing the following steps. (The default number of drill–through rows that can be displayed by a query is 300,000 rows.)

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


Server Manager
SASMain
SASMain - Logical Workspace Server
SASMain - Logical Stored Process Server
SASMain - Logical OLAP Server
SASMain - OLAP Server
SASMain - Logical SAS Data Step Batch Server
SASMain - Logical SAS Java Batch Server
SASMain - Spawner
Job Scheduler
SASMain - HTTP Server

2 Select the server, and then select **File** ▶ **Properties** from the menu bar.

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



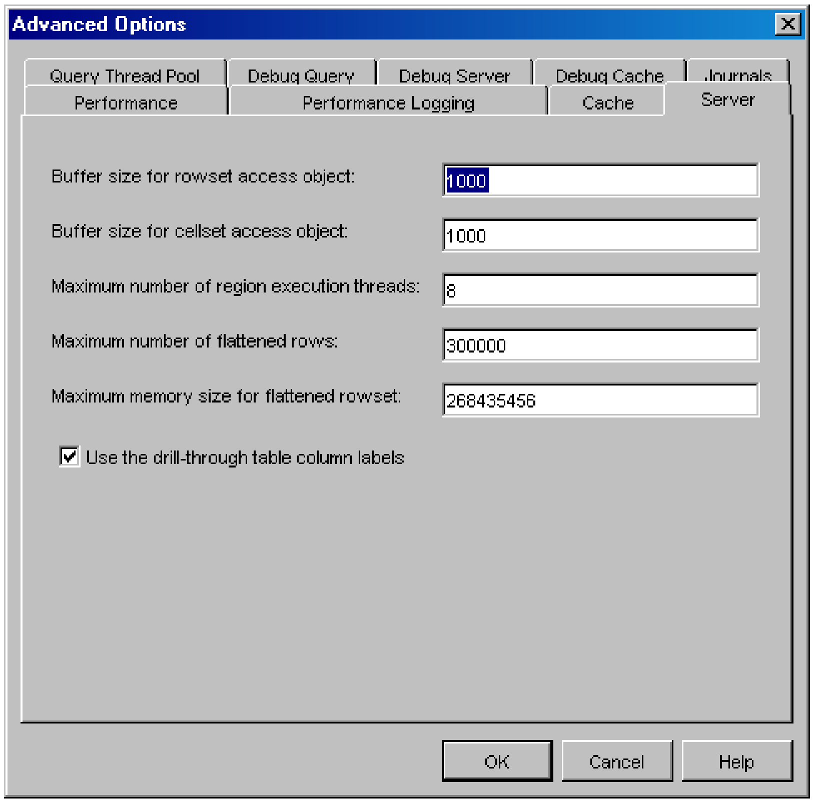
Major Version Number: 9

Minor Version Number: 1

Software Version: 9.1

Vendor: SAS Institute

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 region execution threads: 8

Maximum number of flattened rows: 300000

Maximum memory size for flattened rowset: 268435456

☑ Use the drill-through table column labels

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

APPENDIX 1 Recommended Reading

Recommended Reading

Here is the recommended reading list for this title:

- *SAS Intelligence Platform: System Administration Guide*
- *SAS Data Integration Studio: User's Guide*
- *SAS Language Reference: Concepts*
- *SAS Language Reference: Dictionary*
- *SAS Management Console: User's Guide*
- *SAS Metadata LIBNAME Engine: User's Guide*
- *SAS OLAP Server: Administrator's Guide*
- *SAS OLAP Server: MDX Guide*
- *SAS Scalable Performance Data Engine: Reference*

For a complete list of administration documentation for the SAS Intelligence Platform, see **http://support.sas.com/913administration.**

For a list of SAS documentation, see
http://support.sas.com/documentation/onlinedoc/sas9doc.html.

For a complete list of SAS publications, see the current *SAS Publishing Catalog*. To order the most current publications or to receive a free copy of the catalog, contact a SAS representative at

SAS Publishing Sales
SAS Campus Drive
Cary, NC 27513
Telephone: (800) 727-3228*
Fax: (919) 677-8166
E-mail: **sasbook@sas.com**
Web address: **support.sas.com/pubs**
* For other SAS Institute business, call (919) 677-8000.

Customers outside the United States should contact their local SAS office.

Glossary

administrative user
a special user of a metadata server who can create and delete user definitions and logins. An administrative user can also perform administrative tasks such as starting, stopping, pausing, and refreshing the metadata server. Unlike an unrestricted user, an administrative user does not have unrestricted access to the metadata. You are an administrative user if your user ID is listed in the adminUsers.txt file or if you connect to the metadata server using the same user ID that was used to start the metadata server.

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 set of computing resources that use the same authentication process. An individual uses the same user ID and password for all of the resources in a particular authentication domain. Authentication domains provide logical groupings for resources and logins in a metadata repository. For example, when an application needs to locate credentials that enable a particular user to access a particular server, the application searches the metadata for logins that are associated with the authentication domain in which the target server is registered.

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.

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

libref (library reference)
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 (online analytical processing)
a software technology that enables users to dynamically analyze data that is stored in multidimensional database (MDDB) tables.

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.

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 (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.

Index

Your Turn

If you have comments or suggestions about *SAS® 9.1.3 Intelligence Platform: Data Administration Guide*, please send them to us on a photocopy of this page, or send us electronic mail.

For comments about this book, please return the photocopy to

SAS Publishing
SAS Campus Drive
Cary, NC 27513
E-mail: **yourturn@sas.com**

For suggestions about the software, please return the photocopy to

SAS Institute Inc.
Technical Support Division
SAS Campus Drive
Cary, NC 27513
E-mail: **suggest@sas.com**