THE POWER TO KNOW.

SAS® Scalable Performance Data Server 5.3: Administrator's Guide

SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2016. *SAS® Scalable Performance Data Server 5.3: Administrator's Guide*. Cary, NC: SAS Institute Inc.

SAS® Scalable Performance Data Server 5.3: Administrator's Guide

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

October 2016

P1:spdsag

Contents

What's New in SAS Scalable Performance Data Server 5.3 . . . . *ix*

PART 1 Introduction 1

Chapter 1 • About This Book **3**
Overview 3
Audience 3
Syntax Conventions 3

PART 2 Installation 5

Chapter 2 • Pre-Installation Requirements **7**
Operating System Requirements and Tuning 7
Operating System Resource Configuration 7

Chapter 3 • Installing SAS Scalable Performance Data (SPD) Server on UNIX **9**
Pre-Installation Requirements 9
Installation Directories 10
Software Installation 10
Quick-Start Configuration 11
Testing the Quickstart Configuration 12
Potential Problems with Test Verification 12
Customizing the Default Configuration 13
Completing the Configuration 15
Notes for SPD Server Administrators 15
Troubleshooting 16
Renewing Your SPD Server License 17
Reinstalling and Upgrading SPD Server with a Hotfix 18

Chapter 4 • Installing SAS Scalable Performance Data (SPD) Server on Windows **19**
Pre-Installation Requirements 19
Installation Directories 20
Software Installation 20
Quick-Start Configuration 21
Testing the Quick-Start Configuration 22
Potential Problems with Test Verification 22
Customizing the Default Configuration 23
Completing the Configuration 25
Configuring SPD Server to Run as a Service 25
Notes for SPD Server Administrators 26
Troubleshooting 26
Renewing Your SPD Server License 27
Reinstalling and Upgrading SPD Server with a Hotfix 27

Chapter 5 • Migrating SPD Server to the Current Version **29**
Introduction 29
Overview of Compatibility between Earlier Servers and SPD Server 5.3 29
Converting SPD Server 4.x 32-Bit Windows Tables to SPD
Server 5.3 64-Bit Windows Tables 30
Enabling 32 Groups for SPD Server Users 31

PART 3 Configuration 33

Chapter 6 • Configuring Secure Sockets Communication **35**
Overview of Securing Client/Server Communication 35
Introduction to Transport Layer Security (TLS) 36
Understanding TLS Authentication for SPD Server 36
Overview of Configuring Server Authentication for SPD Server 36
Signed Versus Self-Signed Certificates 37
spdsserv.parm Options for TLS 37
Configuring SPD Server Clients to Perform Server Authentication 39
Examples: Creating and Implementing Self-Signed Certificates for SPD Server 40
Monitoring a Secure Sockets Connection 46
Troubleshooting Secure Sockets Problems 47

Chapter 7 • Managing Passwords and Users **51**
Overview of SPD Server Authentication 51
Understanding Native Authentication 52
Understanding Alternative Methods of Authentication 52
Configuring Nonnative Authentication in SPD Server 55
Creating SPD Server User Accounts 57
Anonymous User ID 57

Chapter 8 • Configuring Server Domains **59**
Overview 60
Syntax for the libnames.parm Parameter File 60
Domain Path Options 61
Path Substitution 61
Other Domain Options 62
Verification of the libnames.parm Parameter File 62
Dynamic Locking 63
Organizing Domains for Scalability 64
Domains and Data Spaces 66
Example libnames.parm Parameter File Configurations 68
Disk Storage 73
Dictionary 73

Chapter 9 • SQL Query Rewrite Facility **79**
Overview of the SQL Query Rewrite Facility 79
Configuring Storage Space for the SQL Query Rewrite Facility 79
SQL Query Rewrite Facility Options 80

Chapter 10 • Setting Server Operational Parameters **83**
Overview 84
Syntax for the spdsserv.parm Parameter File 84
Server Performance Levels 85
Examples of Performance Level Settings 85
Parameter File Validation 86

Dictionary . . . 87

Chapter 11 • Using SPD Server with an Internet Firewall . . . 115
Using SAS Scalable Performance Data (SPD) Server with an Internet Firewall . . . 115

Chapter 12 • Configuring and Administering SPD Server Using the SAS Management Console . . . 119
Overview of SAS Management Console . . . 119
Accessing SPD Server Services in SAS Management Console . . . 120
Connect to an SPD Server . . . 120
Password Manager . . . 122
ACL Manager . . . 126
Server Manager . . . 129
Process Profiler . . . 132
Proxy Manager . . . 134

PART 4 Security 137

Chapter 13 • Security Overview . . . 139
Security Overview . . . 139

Chapter 14 • ACL Security . . . 141
How ACLs Work to Control Access to SPD Server Resources . . . 141
Creating and Managing ACLs . . . 144
ACL Security Model . . . 144
ACL Examples . . . 148

Chapter 15 • Defining WHERE Constraints on Tables . . . 169
Table WHERE Constraints . . . 169

Chapter 16 • Symbolic Substitution . . . 173
Symbolic Substitution . . . 173

Chapter 17 • DICTIONARY.PWDB and DICTIONARY.ACLS . . . 177
DICTIONARY.PWDB and DICTIONARY.ACLS . . . 177

Chapter 18 • Audit File Facility . . . 181
Audit File Facility . . . 181

Chapter 19 • Securing SPD Server Against Unknown Client Attacks . . . 185
Securing SPD Server Against Unknown Client Attacks . . . 185

PART 5 Advanced Configuration Topics 187

Chapter 20 • Optimizing Server Performance . . . 189
SPD Server Performance and Usage Tips . . . 189
Symmetric Multiple Processor (SMP) Utilization . . . 189
File System Performance Concepts . . . 190
SPD Server Domains . . . 192
Optimizing Performance with Macro Variables . . . 193

Chapter 21 • Optimizing Disk Storage . . . 195
Introduction . . . 195
SPD Server Component File Types and Sizes . . . 195
Configuring Domain Disk Space . . . 196
How Table Partition Size Affects the Data Component . . . 198
Component File Pathnames . . . 198

Chapter 22 • Setting Up the Performance Server . . . 201
Overview of SPD Server Performance Server . . . 201
Starting SPD Server Performance Server . . . 202
Performance Server Log File . . . 203
Converting Your spdsperf Log into a Table . . . 203

Chapter 23 • Managing SPDSBASE Processes . . . 205
About SPD Server SPDSBASE Processes . . . 205
SPD Server User Proxy . . . 205
SPD Server SQL Proxy . . . 206
SPD Server SQL User Proxy . . . 206
Enhanced SPDSBASE Sharing . . . 206

PART 6 System Management 209

Chapter 24 • Maintaining SPD Server . . . 211
Maintaining Server Tables . . . 211
Maintaining Server Indexes . . . 211
Overview of SPD Server Maintenance Tools . . . 211

Chapter 25 • Password Database Utility . . . 213
Overview of the Password Database Utility . . . 213
Invoking the psmgr Utility . . . 214
Enabling 32 Groups for SPD Server Users . . . 216
Functional Listing of psmgr Commands . . . 216
psmgr Commands . . . 217

Chapter 26 • SPDO Procedure . . . 227
Overview: SPDO Procedure . . . 228
Concepts: SPDO Procedure . . . 229
Syntax: SPDO Procedure . . . 230
Examples: SPDO Procedure . . . 262

Chapter 27 • Backing Up and Restoring SPD Server Data . . . 269
Overview of the Backing Up and Restoring SPD Server Data . . . 270
Pros and Cons of Each Backup Approach . . . 270
Combining Backup Approaches . . . 271
Introduction to the SPD Server Backup and Restore Utilities . . . 272
SPD Server Backup Utility (spdsbkup) . . . 273
Using spdsbkup . . . 276
SPD Server Restore Utility (spdsrstr) . . . 279
Using spdsrstr . . . 280
Sample Usage Scenarios . . . 283

Chapter 28 • Index Utility . . . 289
Overview . . . 289
Invoking ixutil . . . 289

Syntax for the ixutil Utility 290
Examples 292

Chapter 29 • Domain List Utility 299
Overview 299
Invoking the spdsls Utility 299
Syntax for the spdsls Utility 300
Options for the spdsls Utility 301
Examples 302

Chapter 30 • Directory Cleanup Utility 305
Overview of the Directory Cleanup Utility 305
Wildcards and Pattern Matching with spdsclean 306
Files Used or Affected by spdsclean 306
Syntax for the spdsclean Utility 307
Options for the spdsclean Utility 308
Examples 309

Chapter 31 • Debugging Tools 313
Overview of SPD Server Debugging Tools 313
SPD Server LIBNAME Statement Debug Option 313
spdsserv.parm Debug Option 314

Recommended Reading 315
Glossary 317

What's New in SAS Scalable Performance Data Server 5.3

Overview

SAS Scalable Performance Data (SPD) Server 5.3 includes support for secure sockets communication, a new language driver, and documentation enhancements.

Secure Sockets Communication

SPD Server supports secure sockets communication by using Transport Layer Security (TLS), the successor to Secure Sockets Layer Security (SSL). TLS and SSL are cryptographic protocols that are designed to provide communication security. TLS and SSL provide network data privacy by encrypting client/server communication. In addition, TLS performs client and server authentication, and it uses message authentication codes to ensure data integrity. For more information, see Chapter 6, "Configuring Secure Sockets Communication," on page 35.

New Language Driver

The new language driver enables users to read and write SPD Server tables with the SAS DS2 Language and the SAS FedSQL Language, both of which were introduced with SAS 9.4. The driver is enabled in SPD Server SAS clients by specifying the LIBGEN=YES option in the SPD Server LIBNAME statement. Users submit DS2 language statements to the server by using the DS2 procedure. Users submit FedSQL language statements by using the FEDSQL procedure. For more information, see "Using the SAS DS2 and FedSQL Languages with SPD Server" in *SAS Scalable Performance Data Server: User's Guide*.

SAS Federation Server Support for SPD Server Tables

Beginning in February of 2017, SPD Server tables can be accessed for reading, writing, and update by SAS Federation Server. SAS Federation Server is a data server that provides scalable, threaded, multi-user, and standards-based data access technology in order to process and seamlessly integrate data from multiple data sources. The server acts as a hub that provides clients with data by accessing, managing, and sharing SAS data as well as several popular relational databases. SAS Federation Server enables powerful querying capabilities, as well as centralized data source management. With SAS Federation Server, you can efficiently unite data from many sources, without moving or copying the data. For more information about how SPD Server tables are accessed with SAS Federation Server, see *SAS Federation Server: Administrator's Guide*.

Documentation Enhancements

Beginning in SPD Server 5.3, the documentation for configuring and using SPD Server on Hadoop is contained it its own document: *SAS Scalable Performance Data Server: Processing Data in Hadoop*.

In addition, *SAS Scalable Performance Data Server: Administrator's Guide* and *SAS Scalable Performance Data Server: User's Guide* have been modified as follows:

- The administrator's guide has been reorganized to distinguish between installation, basic configuration, security, advanced configuration tasks, and system management tasks.

  The system management section includes reference information about SPD Server utilities. This includes information about the psmgr utility, which is used to create and maintain the SPD Server password database. See Chapter 25, "Password Database Utility," on page 213. It also includes information about the SPDO procedure, which serves as the operator interface for SPD Server. In previous server releases, information about the SPDO procedure was divided between the user's guide and the administrator's guide. See Chapter 26, "SPDO Procedure," on page 227.

- The user's guide has been reorganized into the following sections: Introduction, Getting Started, SPD Server SQL Processor, Optimizing SPD Server Queries, SPD Server Reference, and ODBC and JDBC Clients.

  The user's guide now provides reference as well as usage information about the SPD Server LIBNAME statement. See "Connecting to the Server" in *SAS Scalable Performance Data Server: User's Guide* and "SPD Server LIBNAME Statement" in *SAS Scalable Performance Data Server: User's Guide*.

  In addition, the user's guide now provides reference information about explicit pass-through statements and server-specific SQL statements. See "Explicit Pass-Through SQL Statements" in *SAS Scalable Performance Data Server: User's Guide* and "SPD Server SQL Statement Additions" in *SAS Scalable Performance Data Server: User's Guide*.

- *SAS Scalable Performance Data Server: Administrator's Guide* also has the following enhancements:

  - Chapter 22, "Setting Up the Performance Server," on page 201 provides modified instructions for starting the performance server. It also provides an example of how to load data from the performance log in to a table for further processing.

  - The documentation for the WORKPATH= libnames.parm parameter file option has been enhanced. See "WORKPATH= Parameter File Option" on page 77.

- The documentation for the BYINDEX, FMTDOMAIN=, FMTNAMENODE=, FMTNAMEPORT=, MAXWHTHREADS=, and WHERECOSTING spdsserv.parm parameter file options has been enhanced. See Chapter 10, "Setting Server Operational Parameters," on page 83.
- The security documentation has been enhanced. See Chapter 13, "Security Overview ," on page 139 and Chapter 14, "ACL Security," on page 141.
- A new topic describes the format of component file pathnames. See "Component File Pathnames" on page 198.
- The documentation for the domain list utility (spdsls) has new examples that show how to list related files for a specified component file. For more information, see Chapter 29, "Domain List Utility," on page 299.
- In January 2017, a new example that shows how to create and configure self-signed TLS certificates for both a UNIX and a Windows SPD Server was added to Chapter 6, "Configuring Secure Sockets Communication," on page 35.

Part 1

Introduction

Chapter 1
About This Book 3

Chapter 1 About This Book

Overview 3
Audience 3
Syntax Conventions 3

Overview

This book contains information necessary to install, configure, secure, and maintain a SAS Scalable Performance Server (SPD Server). For information about how the server operates, and about how to load, create, and manage tables on SPD Server, see *SAS Scalable Performance Data Server: User's Guide*.

Audience

The audience for this book is the SPD Server software installer, SPD Server administrator, and users with special privileges configured on the server. It is helpful if the SPD Server administrator is the SAS administrator, although that is not required.

Syntax Conventions

The parameter file documentation uses the following typographical and syntax conventions:

italicized text
: identifies parameter values that you supply. And example is *primary-path(s)*.

[ ]
: Square brackets identify optional arguments. Any argument that is not enclosed in square brackets is a required argument. Do not type the square brackets. An example is **`[NO]BYINDEX`**.

|
: A vertical bar indicates that you can choose one value from a group. Values separated by bars are mutually exclusive. An example is ***`n`*** | ***`n`*****`K`** | ***`n`*****`M`** | ***`n`*****`G`**.

The documentation for SPD Server utilities uses the following syntax conventions:

< >
: Angle brackets identify arguments whose value you supply. Do not type the angle brackets. An example is **`<dataset.column>`** or **`<physical-path>`**.

[ ]
: Square brackets identify optional arguments. Any argument that is not enclosed in square brackets is a required argument. Do not type the square brackets. An example is **`[-verbose]`**.

{ }
: Braces provide a method to distinguish required multi-word arguments. Do not type the braces. An example is **`{-f <fullfile> | -e <extfile>}`**.

|
: A vertical bar indicates that you can choose one value from a group. Values separated by bars are mutually exclusive.

…
: An ellipsis indicates that the argument or group of arguments that follow the ellipsis can be repeated any number of times. If the ellipsis and the following arguments are enclosed in square brackets, they are optional. An example is **`[table ...]`**.

Part 2

Installation

Chapter 2
Pre-Installation Requirements *7*

Chapter 3
Installing SAS Scalable Performance Data (SPD) Server on UNIX . . *9*

Chapter 4
Installing SAS Scalable Performance Data (SPD) Server on Windows *19*

Chapter 5
Migrating SPD Server to the Current Version *29*

Chapter 2

Pre-Installation Requirements

Operating System Requirements and Tuning 7
Operating System Resource Configuration 7

Operating System Requirements and Tuning

The operating-system-level requirements for SPD Server on a given platform are the same requirements needed for SAS 9.4. For complete information about platform requirements for SPD Server on SAS 9.4, see the information available in Usage Note 42197, via the online SAS Knowledge Base Samples and SAS Notes web page: http://support.sas.com/kb/42/197.html. The technical papers on this page are useful for troubleshooting system performance problems.

Some operating system owners provide white papers that help tune SAS and SPD server for best results on their platform. For example, IBM provides information at http://www.sas.com/en_us/partners/find-a-partner/alliance-partners/ibm.html#white-papers. You can find a list of SAS Alliance Partners such as IBM here: http://www.sas.com/en_us/partners/find-a-partner.html#alliance-partners.

Operating System Resource Configuration

Some operating systems limit the number of concurrent processes a user can own at one time. Some limit the number of concurrent files a user can have open at one time. If you are using an operating system that enforces any of these limits, use these guidelines to configure resources:

Number of Processes per Operating System User ID
: SPD Server requires eight concurrent processes. Furthermore, each user creates another process when the SPD Server client connects to the SPD Server host. Therefore, the maximum number of processes that SPD Server requires is the number of concurrent active SPD Server users plus eight.

Number of Open Files per Process
: During SPD Server queries, all tables that the query requires are fully opened, including the table metadata file, index files, and data partition files. Therefore, the maximum number of open files would be as follows:

```
SPD Server Max Open Files per Process =
[(1 + the maximum number of partitions in a queried SPD Server table) +
(2 * the maximum number of indexes in a queried SPD Server table)
```

For an SPD Server cluster table, the calculated number of open files per process for a single table is multiplied by the number of members in the cluster table.

For queries that involve more than one table, the maximum number of open files per process is the sum of the open files for each table (or cluster member) involved in the query.

A general practice for SPD Server resource management is to configure the number of SPD Server open files per process that are available to the user who owns the SPD Server executables to the system maximum value. If SPD Server reaches a system-imposed limit for the number of processes per user, or number of open files per user, the query fails and an error is printed to the SPD Server log.

Chapter 3

Installing SAS Scalable Performance Data (SPD) Server on UNIX

Pre-Installation Requirements 9

Installation Directories 10

Software Installation 10

Quick-Start Configuration 11

Testing the Quickstart Configuration 12

Potential Problems with Test Verification 12

Customizing the Default Configuration 13
- Logging 13
- Audit File Facility 14
- User Password and Parameter Files 14
- Accessing SPD Server through a Registered Port 14
- SNET Server 15

Completing the Configuration 15

Notes for SPD Server Administrators 15
- UNIX User IDs 15
- SPD Server Users 16

Troubleshooting 16
- Name Server Start-Up Failed 16
- SPD Server Host Start-Up Failed 16
- SAS LIBNAME Assignment Failed 17

Renewing Your SPD Server License 17

Reinstalling and Upgrading SPD Server with a Hotfix 18

Pre-Installation Requirements

The following are requirements for the UNIX user ID of the SPD Server installer:

- Use a UNIX user ID other than root to install and run your production SPD Server.
- The user ID needs appropriate access permissions to create the installation directory for SPD Server on the file system where you want to install the server software.

- The user ID should become the user ID of the SPD Server Administrator. In this way, the administrator has access to the installation directory and all SPD Server processes.
- If you want SPD Server clients to connect to SPD Server by using name services instead of by specifying port numbers at invocation, the user ID needs Write access to *both* the server and the client machine's **`/etc/inet/services`** or **`/etc/services`** files. Name services require you to define registered ports that use the services file appropriate to your machine.
- For more information, see "Notes for SPD Server Administrators" on page 15.

Install SPD Server in a location that is adequately mirrored and backed up to ensure reliability.

Installation Directories

The installation will create several subdirectories in your SPD Server host installation directory, **`<SASHome>/SASScalablePerformanceDataServer/release-number`**. This directory is referred to as InstallDir/ below.

Note: **`InstallDir/`** represents the root directory in which SPD Server is installed.

The **`InstallDir/bin/`** subdirectory contains SPD Server executable files.

The **`InstallDir/lib/`** subdirectory contains libraries that facilitate third-party access to SPD Server (other than via the SPD Server LIBNAME engine).

The **`InstallDir/samples/`** subdirectory contains sample start-up scripts and SAS programs that provide example uses of SPD Server.

The **`InstallDir/site/`** subdirectory is a storage directory for a user's site-specific customization of the sample SPD Server start-up and configuration files. The spdsclean utility script is created in this location as well.

Software Installation

Follow the instructions on your SAS software order to obtain your software: Use SAS Download Manager to download your order, and then use SAS Deployment Wizard to install your order.

While in SAS Deployment Wizard, be sure to select the following products for installation:

- SAS Scalable Performance Data Server
- SAS Scalable Performance Data Client
- SAS Management Console
- SAS Scalable Performance Data Server Plug-in for SAS Management Console
- SAS/SECURE Toolkit Libraries from RSA
- SAS/SECURE SSL.

During installation of SAS Scalable Performance Data Server, you will be prompted to configure a SAS Scalable Performance Data Server connection port, as well as a SAS Scalable Performance SNET Server connection port.

- The data server connection port connects the SPD Server SAS client engine and ODBC clients to the SPD Server name server.
- The SNET Server connection port connects a JDBC client to the SPD Server SNET Server. The SNET Server port is required in order to connect to SPD Server from a JDBC client.

The default SPD Server port assignments are 5400 and 5401, respectively. You can change the default port assignments to port numbers that are available on your system. If you are running within a firewall, see "Using SAS Scalable Performance Data (SPD) Server with an Internet Firewall" on page 115.

The SAS/SECURE components should be selected for SPD Server SAS client installations as well as the server installation.

Quick-Start Configuration

This section guides you through a quick-start configuration of SPD Server. The quick configuration enables you to validate and test your installation before performing more detailed configuration tasks.

After the SPD Server software is installed, navigate to the **`InstallDir/site`** subdirectory, and do the following to complete the quick configuration of your server:

1. Verify that the WORKPATH statement in your spdsserv.parm parameter file is valid. If the WORKPATH statement is incorrect, edit your spdsserv.parm parameter file.
2. Verify that the pathname for libname=tmp in the libnames.parm parameter file is valid. If the pathname is not correct, edit the libnames.parm parameter file.
3. Initialize the SPD Server password database by invoking the pwdb script and doing the following:
   a. At the **`Enter command>`** prompt, enter **`groupdef`**.
   b. At the **`Enter group name to define>`** prompt, enter **`admingrp`**.
   c. At the **`Enter command>`** prompt, enter **`add`** to add yourself as an administrator.
   d. Choose a name up to eight alphanumeric characters, and choose a temporary password of six to eight alphanumeric characters that uses at least one numeral.
   e. Assign the user an authorization level of 7 for maximum permissions, and then assign the user to the AdminGrp group. (For more information about authorization levels, see "Special Privilege" on page 139.)
   f. Press Enter to use default settings for the remaining prompts.
   g. Change the user's password to a permanent password by entering **`chgpass`** at the **`command>`** prompt, and follow the instructions.
   h. Enter **`quit`** at the **`Enter command>`** prompt to exit the script.
4. Start SPD Server by executing the rc.spds script.
5. Use the UNIX PS command to verify that SPD Server is running.

   ```
   ps -fe | grep port-number
   ```

   If SPD Server is running, you should see the following processes:

   - spdsserv
   - spdsbase

- spdssnet
- spdsnsrv

You should also see several spdslog processes.

If SPD Server is not running, review the SPD Server log at **`InstallDir/log/spdsserv*spdslog`** for any errors.

After you correct any start-up errors, end all running SPD Server processes. Then restart the software by using the rc.spds script.

6. Test the configuration. When all is well, review the default configuration and modify it to meet your preferences. See "Customizing the Default Configuration" on page 13.

Testing the Quickstart Configuration

Use SAS to connect to SPD Server and verify your installation.

In the SAS Program Editor, submit the following code. Be sure to change "InstallDir" references to a real directory path before submitting the requests.

```
%let spdshost=<hostname where SPD Server is running>;
%let spdsport=<your SPD Data Server port>;

libname test sasspds 'tmp'
  host="&spdshost"
  serv="&spdsport"
  user="anonymous";

libname testrl sasspds 'tmp'
  host="&spdshost"
  serv="&spdsport"
  user="anonymous"
  locking=yes;

/* Include InstallDir/samples/verify.sas" */
/* to verify dataset access to SPD Server.*/

%inc "<install_dir>/samples/verify.sas";

/* Include InstallDir/samples/verptsql.sas */
/* to verify SQL access to SPD Server.      */

%inc "<install_dir>/samples/verptsql.sas";
endsas;
```

Potential Problems with Test Verification

The verification program can fail with the following errors:

ERROR: The SASSPDS Engine cannot be found
: This error indicates that your SAS installation did not include the SAS Scalable Performance Data Server client. Review your SAS installation.

ERROR: Unable to connect to SPD Name Server
: This error indicates that SPD Server is not running, or that your SPDSHOST and SPDSPORT connection variable settings are not correctly configured. Verify your connection variables and ensure that your SPD Server is running on the specified host using the specified port.

ERROR: Protocol version mismatch. Proxy version is 5.3 while engine version is <version-number>
: This error indicates that you are trying to connect to an SPD Server host with an old client. For SPD Server 5.3, verify that you are running SAS 9.4 with the SAS Scalable Performance Data Server client.

Customizing the Default Configuration

This section provides an overview of the default configuration settings for basic activities such as logging, auditing, and parameter file locations, and describes how to change them. To modify the default settings, you will need to shut down SPD Server, make the changes, and then re-start SPD Server. You can shut down SPD Server by using the **`InstallDir/site/killrc`** script.

Logging

By default, SPD Server creates log files in the **`InstallDir/log`** directory. The files contain messages that are written to STDOUT or STDERR for the spdsnsrv (SPD Server name server), spdsserv (SPD Server host), and spdssnet (SPD Server SNET Server) processes. The content and location of SPD Server logs is controlled by the value of the rc.spds script variable LOGDIR= .

If you do not want to keep these logs, change the value of LOGDIR= to **`/dev/null`**. If you want to keep the logs in a location other than **`InstallDir/log`**, specify the new path value in LOGDIR=.

SPD Server log files can grow very large if there is considerable activity. The log files are configured to recycle at a given time each day and start a new logfile. The older log files can then be removed or archived.

The rc.spds script contains the following variables to control the log name and recycle time:

SPD Server host process:

DSRVFILE=spdsserv
: specifies the spdsserv process log file prefix. By default, the name spdsserv_mmddyyyy_hh:mm:ss.spdslog is generated. The values mmddyyyy and hh:mm:ss indicate the time at which the system created the log file.

DSRVTIME=00:00
: specifies a recycle time of midnight.

Name server process:

NSRVFILE=spdsnsrv
: specifies the spdsnsrv process log file prefix. By default, the name spdsnsrv_mmddyyyy_hh:mm:ss.spdslog is generated. The values mmddyyyy and hh:mm:ss indicate the time at which the system created the log file.

NSRVTIME=00:00
: specifies a recycle time of midnight.

SNET Server process:

SNSFILE=spdsnet
: specifies the prefix of the spdssnet process log file. By default, the name spdssnet_mmddyyyy_hh:mm:ss.spdslog is generated. The values mmddyyyy and hh:mm:ss indicate the time at which the system created the log file.

SNSTIME=00:00
: specifies the time of midnight.

To disable automatic log filename generation and cycling, change those settings to empty pointers.

See the comments in the rc.spds script for information to set the log files to recycle at a different time of day or interval from the start time.

Audit File Facility

The audit file facility is not enabled by default. Modify the following variables in the rc.spds script to enable auditing:

AUDDIR=
: specifies the directory for the audit log files.

AUDFILE=
: specifies the prefix for audit log files.

AUDFILESQL=
: specifies the prefix for SQL audit log files.

AUDTIME=
: specifies the time of day (HH:MM) to cycle the audit log file.

The AUDDIR= and AUDFILE= variables enable proxy audit file creation. The AUDDIR= and AUDFILESQL= variables enable SQL audit file creation. The AUDTIME= variable enables automatic audit file cycling at the specified time of day. For more information about SPD Server auditing, see "Audit File Facility" on page 181.

User Password and Parameter Files

The rc.spds script looks for the spdsserv.parm parameter file, which contains server processing parameters. The script also looks for the password database, which stores information about SPD Server users, in the `InstallDir/site` directory. If you do not want to keep the files in this location, change the paths of the PARMDIR= and ACLDIR= variables, respectively.

Accessing SPD Server through a Registered Port

To enable access to SPD Server through a registered port (name service), add the following service to your `/etc/inet/services` or `/etc/services` file (if this service is not already present):

```
spdsname 5400/tcp # SPDS Name Service
```

This service defines the port number for the name server process. Make sure that this port number matches the port number that you used when you installed SPD Server. If you are running SAS on an existing SPD Server installation, this service name is probably already defined. You can either define another service name for the SAS client to use (for example, *sp53name*), or you can directly include the SPD Server port number in your SAS statements.

SNET Server

The default SPD Server installation assumes that you want the SNET Server (spdssnet process) for accessing SPD Server data via a JDBC client to run. To disable start-up of the SNET server, specify the -NOSNET option when executing the rc.spds script to start SPD Server. Or modify the rc.spds script to set SNET=NO.

Completing the Configuration

Once the basic configuration of SPD Server is the way you want it, you can expand the configuration to add users, configure secure client/server communication, configure data access, define data security, and optimize server processing and performance settings.

- For information to secure client/server communication, see Chapter 6, “Configuring Secure Sockets Communication,” on page 35.
- For information to add users, see Chapter 7, “Managing Passwords and Users,” on page 51 and Chapter 25, “Password Database Utility,” on page 213.
- For information to configure data access, see Chapter 8, “Configuring Server Domains,” on page 59. Also see Chapter 21, “Optimizing Disk Storage,” on page 195.
- For information to define data security, see Chapter 13, “Security Overview ,” on page 139.
- For information to optimize server processing and performance settings, see Chapter 10, “Setting Server Operational Parameters,” on page 83.

Notes for SPD Server Administrators

The SPD Server administrator performs the configuration and maintenance functions for the server. The following sections contain guidelines for administrators.

UNIX User IDs

The SPD Server administrator requires a UNIX login ID on the machine where SPD Server will be installed and administered. Other server users do not need UNIX login IDs. Other users’ access to server data resources is controlled via SPD Server user IDs in the server’s password database.

Administer your SPD Server environment by using the same UNIX user ID that was used to install SPD Server on the host machine. The common user ID minimizes potential problems with file ownership and system access permissions on the server machine.

Regardless of how the server run-time environment is configured, server processes always run using the UNIX user ID that started the SPD Server session. That UNIX user ID owns all of the files that the server process creates. The UNIX user ID is governed by UNIX file access permissions. Remember this when you start server processes and run server administration utilities. Otherwise, it is possible to create files that have ownership and permissions that deny access to required server processes. If you perform all SPD Server installation and administration tasks from the same UNIX user ID, subsequent use of the server is much easier.

SPD Server Users

The SPD Server administrator uses the SPD Server psmgr utility to register users. SPD Server uses its own layer of access controls that overlay UNIX access permissions.

Each server user is given their own SPD Server user ID and password. The user ID and password are needed to complete the LIBNAME connection to the server. All resources that a user creates are owned by the user. A server user can access only resources that he or she created, and resources that another server user grants him or her access to via server Access Control Lists (ACLs). There also exists an Anonymous user account that any SPD Server user can access with no password, and where all resources that are created by the Anonymous user are accessible to any other server user. For more information, see “Anonymous User ID” on page 57.

Troubleshooting

Key information for SPD Server troubleshooting can be found in the name server (spdsnsrv*) and host process (spdsserv*) log files in `InstallDir/log` subdirectory. Those two log files enable you to reconstruct SAS interactions with server components. Entries in the log files are time-stamped for reference. You should be able to correlate activities between the two logs by using the time-stamp information. The logs are formatted as plain text files.

Name Server Start-Up Failed

Check the name server log file. The log should contain information about the problem. Some common things to look for include the following:

- The -LICENSEFILE file specification is not valid.
- -LICENSEFILE specifies a file with invalid contents.
- The name server port is in use by another process.

Determine whether another name server process is already running on the same node by issuing the following command:

```
ps -ef | grep -i spdsnsrv
```

SPD Server Host Start-Up Failed

Check the SPD Server host log file for information. Some common things to look for include the following:

- The -NAMESERVER node name is incorrect.
- -NAMESERVERPORT specifies the wrong port number if the SPD Server name server is running with a nonstandard port assignment.
- The -PARMFILE file specification is invalid, or the specified file does not exist.
- The -LIBNAMEFILE file specification is invalid, or the specified file does not exist.
- The contents of the specified -LIBNAMEFILE does not conform to expected syntax. Check the SPD Server host log file for messages about invalid entries.
- The -ACLDIR option was omitted from the command line.
- The -ACLDIR option specifies an invalid directory path for the SPD Server password file, or the specified directory path does not contain a valid SPD Server password file.

SAS LIBNAME Assignment Failed

If the SAS LIBNAME assignment fails, first check the error messages from the SPD Server LIBNAME engine through the SAS log output. In most circumstances, you can diagnose the reason for the failure from these messages. Some common problems include the following:

ERROR: The SASSPDS engine cannot be found.
This error indicates that your SAS installation did not include the SAS Scalable Performance Data Server client. Review your SAS installation.

ERROR: Unable to connect to SPD Name Server.
This error indicates that SAS SPD Server is not running, or your LIBNAME HOST= or SERV= values are not correct. Verify your LIBNAME statement and that SPD Server is running on the correct host with the correct port.

ERROR: Protocol version mismatch. Proxy version is 5.3 while engine version is <version>.
This error indicates that you are trying to connect to SPD Server with an old client. Verify that you are running the latest version of the SAS Scalable Performance Data Server client.

ERROR: SPD Server has rejected login from user <username>
This error indicates you are trying to connect to SPD Server with an invalid user ID or password. Verify your LIBNAME statement.

ERROR: Password has expired and a new password not given.
This error indicates your password has expired and you need to use the NEWPASSWORD LIBNAME option to update your password.

Renewing Your SPD Server License

When you receive SPD Server, licensing information is pre-initialized. When you renew the license, you receive a new license to replace your existing license. To use the new license file, stop SPD Server, edit the LICFILE= setting in your rc.spds script to reference the name of the new license file, and then re-start SPD Server.

Reinstalling and Upgrading SPD Server with a Hotfix

If you need to reinstall SPD Server or install a hotfix, the installation process will not replace any customized SPD Server start-up scripts, parameter files, or utility scripts files that have been modified in the **`InstallDir/site`** directory. Nor will it affect the password database created by psmgr utility or pwdb script. Any custom modifications that were made to installation files are retained when you reinstall SPD Server or install a hotfix.

Chapter 4

Installing SAS Scalable Performance Data (SPD) Server on Windows

Pre-Installation Requirements **19**

Installation Directories **20**

Software Installation **20**

Quick-Start Configuration **21**

Testing the Quick-Start Configuration **22**

Potential Problems with Test Verification **22**

Customizing the Default Configuration **23**
Logging 23
Audit File Facility 23
User Password and Parameter Files 24
Accessing SPD Server through a Registered Port 24

Completing the Configuration **25**

Configuring SPD Server to Run as a Service **25**

Notes for SPD Server Administrators **26**
SPD Server Users 26

Troubleshooting **26**
Name Server Start-Up Failed 26
SPD Server Host Start-Up Failed 26
SAS LIBNAME Assignment Failed 27

Renewing Your SPD Server License **27**

Reinstalling and Upgrading SPD Server with a Hotfix **27**

Pre-Installation Requirements

If you want SPD Server clients to connect to the SPD Server host using name services instead of specifying port numbers at invocation, the SPD Server installer ID needs Write access to the server machine's `...\etc\services` directory. Name services require you to define registered ports that use the services file appropriate to your machine. The installer ID also needs Write access to the services files on the clients, in the path `...\etc\services`.

Installation Directories

The installation will create several subdirectories in your SPD Server host installation directory, **<*SASHome*>\SASScalablePerformanceDataServer\*release-number***. This directory is referred to as InstallDir\ below.

Note: **InstallDir\** represents the root directory in which SPD Server is installed.

The **InstallDir\bin\** subdirectory contains the SPD Server executable files.

The **InstallDir\lib\** subdirectory contains libraries that facilitate third-party access to SPD Server (other than via the SPD Server LIBNAME engine).

The **InstallDir\samples\** subdirectory contains sample start-up scripts and SAS programs that provide example uses of SPD Server.

The **InstallDir\site\** subdirectory is a storage directory for a user's site-specific customization of the sample SPD Server start-up and configuration files.

Software Installation

Follow the instructions on your SAS software order to obtain your software: Use SAS Download Manager to download your order, and then use SAS Deployment Wizard to install your order.

While in SAS Deployment Wizard, be sure to select the following products for installation:

- SAS Scalable Performance Data Server
- SAS Scalable Performance Data Client
- SAS Management Console
- SAS Scalable Performance Data Server Plug-in for SAS Management Console
- SAS/SECURE Toolkit Libraries from RSA
- SAS/SECURE SSL.

During installation of SAS Scalable Performance Data Server, you will be prompted to configure a SAS Scalable Performance Data Server connection port, as well as a SAS Scalable Performance SNET Server connection port.

- The data server connection port connects the SPD Server SAS client engine to the SPD Server name server.
- The SNET Server connection connects a JDBC client to the SPD Server SNET Server. The SNET Server port is required in order to connect to SPD Server from a JDBC client.

The default SPD Server port assignments are 5400 and 5401, respectively. You can change the default port assignments to port numbers that are available on your system. If you are running within a firewall, see "Using SAS Scalable Performance Data (SPD) Server with an Internet Firewall" on page 115 for more detailed information.

The SAS/SECURE components should be selected for SPD Server SAS client installations as well as the server installation.

Quick-Start Configuration

This section guides you through a quick-start configuration of SPD Server to enable you to validate and test your installation before performing more detailed configuration tasks.

After you install SPD Server, navigate to the `\site` subdirectory (relative to the directory where you installed SPD Server), and do the following to complete the quick configuration of your server:

1. Verify that the WORKPATH statement in your spdsserv.parm parameter file is valid. If the WORKPATH statement is incorrect, edit your spdsserv.parm parameter file.
2. Verify that the pathname for libname=tmp in the libnames.parm parameter file is valid. If the pathname is not correct, edit the libnames.parm parameter file.
3. Initialize the SPD Server password database:
   a. Click the Windows **Start** button and select **All Programs** ⇨ **SAS** ⇨ **SAS SPD Server 5.3** ⇨ **Account Manager**. The SPD Server Account Manager is launched in a command window.
   b. At the **`Enter command>`** prompt, enter **`groupdef`**.
   c. At the **`Enter group name to define>`** prompt, enter **`admingrp`** .
   d. To add yourself as an administrator, enter **`add`** at the **`Enter command>`** prompt.
   e. Choose a user name of up to eight alphanumeric characters, and choose a temporary password of six to eight alphanumeric characters that uses at least one numeral.
   f. Assign yourself an authorization level of 7 for maximum permissions, and then assign yourself to the AdminGrp group. (For more information about authorization levels, see "Special Privilege" on page 139). Press Enter to use default settings for the remaining prompts.
   g. To change your password to a permanent password, enter **`chgpass`** at the **`command>`** prompt and follow the instructions.
   h. To exit the password database utility, enter *quit* at the **`Enter command>`** prompt.
4. Start the SPD Server name server. Click the Windows **Start** button and select **All Programs** ⇨ **SAS** ⇨ **SAS SPD 5.3 Name Server**.
5. Start SPD Server. Click the Windows **Start** button and select **All Programs** ⇨ **SAS** ⇨ **SAS SPD 5.3 Data Server**.
6. Use the Windows Task Manager to verify that SPD Server is running. You should see the following processes running:
   - spdsserv.exe
   - spdsbase.exe
   - spdsnsrv.exe
   - spdslog.exe

   If SPD Server is not running, review the SPD Server log at **`InstallDir\log\spdsserv*spdslog`** for any errors.

After you correct any start-up errors, end all running SPD Server processes using the Windows Task Manager, and then restart your server.

7. Test the configuration. When all is well, review the default configuration and modify it to meet your preferences. See “Customizing the Default Configuration” on page 23.

Testing the Quick-Start Configuration

Use SAS to connect to SPD Server and to verify your installation.

In the SAS Program Editor, submit and run the following:

```
%let spdshost=<hostname where SPD Server is running>;
%let spdsport=<your SPD Data Server port>;

libname test sasspds 'tmp'
  host="&spdshost"
  serv="&spdsport"
  user="anonymous";

libname testrl sasspds 'tmp'
  host="&spdshost"
  serv="&spdsport"
  user="anonymous"
  locking=yes;


/* Include InstallDir\samples\verify.sas" */
/* to verify dataset access to SPD Server.*/

%inc "<install_dir>\samples\verify.sas";

/* Include InstallDir\samples\verptsql.sas */
/* to verify SQL access to SPD Server.       */

%inc "<install_dir>\samples\verptsql.sas";
endsas;
```

Potential Problems with Test Verification

The verification programs can fail with the following errors:

ERROR: The SASSPDS Engine cannot be found
: This error indicates that your SAS installation did not include the SAS Scalable Performance Data Server client. Review your SAS installation.

ERROR: Unable to connect to SPD Name Server
: This error indicates that SPD Server is not running, or that your SPDSHOST and SPDSPORT connection variable settings are not correctly configured. Verify your

connection variables, and ensure that your SPD Server is running on the specified host using the specified port.

ERROR: Protocol version mismatch. Proxy version is 5.3 while engine version is <version-number>
: This error indicates that you are trying to connect to an SPD Server host with an old client. For SPD Server 5.3, verify that you are running SAS 9.4 with the SAS Scalable Performance Data Server client.

Customizing the Default Configuration

This section provides an overview of the default configuration settings for basic activities such as logging, auditing, and parameter file locations, and describes how to change them. To modify the default settings, you will need to shut down SPD Server, make the changes, and then re-start SPD server. Shut down SPD Server by using the Windows Task Manager.

Logging

By default, SPD Server creates log files in the **`InstallDir\log`** directory. The files contain messages that are written to STDOUT or STDERR for the spdsnsrv (SPD Server name server), spdsserv (SPD Server host), and spdssnet (SPD Server SNET Server) processes. The content and location of SPD Server logs is controlled by the **`InstallDir\site\*.bat`** files for each process.

SPD Server log files can grow very large if there is considerable activity. The log files can be configured to recycle at a given time each day and start a new logfile. The older log files can then be removed or archived.

To alter the name, location, or cycle time to generate a new log file, modify the following start-up parameters for the spdsserv, spdsnsrv, or spdssnet processes in their respective batch files:

`-logfile` ***`fileSpec`***
: specifies that the logger process automatically creates a server log file. *fileSpec* specifies a partial pathname or filename that is used to generate the complete log file path.

: For example, if you specify *fileSpec* as **`c:\logs\spdsnsrv`**, the name **`c:\logs\spdsnsrv_mmddyyyy_hh:mm:ss.spdslog`** is generated. The values **`mmddyyyy`** and **`hh:mm:ss`** indicate the time at which the system created the log file.

`-logtime hh:mm`
: specifies the time of day at which to cycle a new generation of the name server log file. At this time each day, the previous log file is closed and a new log file is opened. For example, the default value, **`-logtime 00:00`**, cycles the log at midnight.

Audit File Facility

The audit file facility is not enabled by default. To enable SPD Server auditing, include the following start-up parameters when you invoke the **`Installdir\site\spdsserv.bat`** file:

-auditfile *fileSpec*
The AUDITFILE argument enables proxy audit logging for the server, and enables automatic audit log file creation by the audit process. The parameter *fileSpec* specifies a path or filename that is used to generate the complete audit file path.

For example, if you specify *fileSpec* as **\audit\spds**, the generated name will be **\audit\spds_mmddyyyy_hh:mm:ss.spdsaudit**, where **mmddyyyy** is the system date when the log file was created.

-sqlauditfile *fileSpec*
The SQLAUDITFILE argument enables SQL audit logging for the server, and enables automatic audit log file creation by the audit process. The parameter *fileSpec* specifies a path or filename that is used to generate the complete SQL audit file path. For example, if you specify *fileSpec* as **\audit\spdssql**, the generated name will be **\audit\spdssql_mmddyyyy_hh:mm:ss.spdsaudit**, where **mmddyyyy** is the system date when the log file was created.

-audittime *hh:mm*
The AUDITTIME argument specifies the time of day to cycle a new generation of the audit log file or SQL audit log file. At this time each day, the previous log file is closed and a new log file is opened. For example, **-audittime 00:00** cycles the logs at midnight.

For more information about SPD Server auditing, see "Audit File Facility" on page 181.

User Password and Parameter Files

The spdsserv.bat file looks for the spdsserv.parm parameter file, which contains server processing parameters; the libnames.parm parameter file, which defines SPD Server domains; and the SPD Server password database, which stores information about SPD Server users, in the **InstallDir\site** directory. If you do not want to keep the files in this location, modify the following arguments in the spdsserv.bat file:

-parmfile *fileSpec*
The PARMFILE argument specifies an explicit file path for the spdsserv.parm parameter file. This file is mandatory and contains any SPD Server options. If this option is omitted, the SPD Server host assumes that a parameter file named spdsserv.parm is in the process's current working directory.

-libnamefile *fileSpec*
The LIBNAMEFILE argument specifies the name of the file that contains the domain definitions that the SPD Server host supports.

-acldir *fileSpec*
The ACLDIR argument specifies the name of the directory for the password database.

.

Accessing SPD Server through a Registered Port

To enable access to SPD Server through a registered port (name service), add the following service to your **\etc\inet\services** or **\etc\services** file (if this service is not already present):

```
spdsname 5400\tcp # SPDS Name Service
```

This service defines the port number for the SPD Server name server process. Make sure that this port number matches the port number that you used when you installed SPD

Server. If you are running SAS on an existing SPD Server installation, this service name is probably already defined. You can either define another service name for the SAS client to use (for example, *sp53name*) or you can directly include the SPD Server port number in your SAS statements.

Completing the Configuration

Once the basic configuration of SPD Server is the way you want it, you can expand the configuration to add users, configure data access, configure secure client/server communication, define data security, and optimize server processing and performance settings.

- For information to secure client/server communication, see Chapter 6, "Configuring Secure Sockets Communication," on page 35.
- For information to add users, see Chapter 7, "Managing Passwords and Users," on page 51 andChapter 25, "Password Database Utility," on page 213.
- For information to configure data access, see Chapter 8, "Configuring Server Domains," on page 59. Also see Chapter 21, "Optimizing Disk Storage," on page 195.
- For information to define data security, see Chapter 13, "Security Overview ," on page 139.
- For information to optimize server processing and performance settings, see Chapter 10, "Setting Server Operational Parameters," on page 83.

Configuring SPD Server to Run as a Service

The SPD Server installation configures SPD Server to run under Windows Services by default. This enables SPD Server to automatically be started and stopped when users start and stop Windows. It also enables you to verify SPD Server services via Windows Services. To open the Windows Services window, select **Start** ⇨ **Control Panel** ⇨ **Administrative Tools** ⇨ **Services**. The main panel of Windows Services contains a sortable list of Windows services. Scroll down the **Services** list to find entries for the SPD Server 5.3 services.

Most users prefer to use Windows Services. If you do not want to take advantage of Windows Services, you can disable use of Windows Services for SPD Server as follows.

1. In the Window Services window, scroll down the **Services** list to find the entry for SPD Server 5.3 Name Server.
2. Right-click the name server service in the list, and select **Properties**. The Properties window appears.
3. Select **Manual** from the **Start-up type** list. This setting configures the name server for manual start up. Click **OK** to apply the changes and close the window.
4. Repeat this process to change the **Start-up type** setting for the SPD Server data server and the SPD Server SNET Server. At this point, your name server, data server, and SNET Server services are configured to be started manually.

Notes for SPD Server Administrators

The SPD Server administrator performs maintenance and configuration functions for the server. The following sections contain guidelines for administrators.

SPD Server Users

The SPD Server administrator needs to be familiar with the SPD Server psmgr utility. The server uses its own layer of access controls that overlay Windows access permissions. Server processes run in the context of a Windows user ID, and that user owns all of the resulting SPD Server file resources that are created.

Each server user is given their own SPD Server user ID and password. The user ID and password are needed to connect to SPD Server. All resources that a user creates are owned by the user. A server user can access only resources that the user created and resources that another server user grants them access to via server Access Control Lists (ACLs). There also exists an Anonymous user account that any server user can access with no password, and where all resources that are created by the Anonymous user are accessible to any other server user.For more information, see “Anonymous User ID” on page 57.

Troubleshooting

Key information for SPD Server troubleshooting can be found in the name server (spdsnsrv*) and host process (spdsserv*) log files. Those two log files enable you to reconstruct SAS interactions with server components. Entries in the log files are time-stamped for reference. You should be able to correlate activities between the two logs by using the time-stamp information. The logs are formatted as plain text files.

Name Server Start-Up Failed

Check the name server log file. The log should contain information about the problem. Some common things to look for include the following:

- The -LICENSEFILE file specification is not valid
- -LICENSEFILE specifies a file with invalid contents.
- The name server port is in use by another process.

SPD Server Host Start-Up Failed

Check the SPD Server host log file for information. Some common things to look for include the following:

- The -NAMESERVER node name is incorrect.
- -NAMESERVERPORT specifies the wrong port number if the SPD Server name server is running with a nonstandard port assignment.
- The -PARMFILE file specification is invalid, or the specified file does not exist.
- The -LIBNAMEFILE file specification is invalid, or the specified file does not exist.

- The contents of the specified -LIBNAMEFILE does not conform to expected syntax. Check the SPD Server host log file for messages about invalid entries.
- The -ACLDIR option was omitted from the command line.
- The -ACLDIR option specifies an invalid directory path for the SPD Server password file, or the specified directory path does not contain a valid SPD Server password file.

SAS LIBNAME Assignment Failed

If the SAS LIBNAME assignment fails, first check the error messages from the SAS LIBNAME engine through the SAS log output. In most circumstances, you can diagnose the reason for the failure from these messages. Some common problems include the following:

ERROR: The SASSPDS engine cannot be found.
This error indicates your SAS installation did not include the SAS Scalable Performance Data Server client. Review your SAS installation.

ERROR: Unable to connect to SPD Name Server.
This error indicates that SAS SPD Server is not running, or your LIBNAME HOST= or SERV= values are not correct. Verify your LIBNAME statement and that SPD Server is running on the correct host with the correct port.

ERROR: Protocol version mismatch. Proxy version is 5.3 while engine version is <version>.
This error indicates that you are trying to connect to SPD Server with an old client. Verify that you are running the latest version of the SAS Scalable Performance Data Server client.

ERROR: SPD Server has rejected login from user <username>
This error indicates you are trying to connect to SPD Server with an invalid user ID or password. Verify your LIBNAME statement.

ERROR: Password has expired and a new password not given.
This error indicates your password has expired and you need to use the NEWPASSWORD LIBNAME option to update your password.

Renewing Your SPD Server License

When you receive SPD Server, licensing information is pre-initialized. When you renew the license, you receive a new license to replace your existing license. To use the new license file, stop SPD Server, edit the -licensefile setting in your spdsnsrv.bat file to reference the name of the new license file, and then re-start SPD Server.

Reinstalling and Upgrading SPD Server with a Hotfix

If you need to reinstall SPD Server or install a hotfix, the installation process will not replace any customized SPD Server start-up scripts, parameter files, or utility scripts files that have been modified in the `InstallDir\site` directory. Nor will it affect the password database created by psmgr utility or pwdb script. Any custom modifications that were made to installation files are retained when you reinstall SPD Server or install a hotfix.

Chapter 5 Migrating SPD Server to the Current Version

Introduction **29**

Overview of Compatibility between Earlier Servers and SPD Server 5.3 . . . . **29**

Converting SPD Server 4.x 32-Bit Windows Tables to SPD Server 5.3 64-Bit Windows Tables **30**

Convert SPD Server 4.x Tables to SAS Tables and Then to SPD Server 5.3 Tables 30

Convert Directly from SPD Server 4.4x and 4.5x Tables to SPD Server 5.3 Tables 31

Using PROC COPY with SPD Server Cluster Tables 31

Enabling 32 Groups for SPD Server Users **31**

Migrating SPD Server 4.x Password Tables to SPD Server 5.x Password Tables . . 32

The psmgr Utility GROUPMEM Updates 32

Introduction

This chapter describes migration requirements and options for customers who are migrating from an earlier version of SPD Server to SPD Server 5.3.

Overview of Compatibility between Earlier Servers and SPD Server 5.3

SPD Server 5.3 can read earlier SPD Server 5.x tables and 4.x tables. If SPD Server 5.3 modifies an earlier 5.1 or 4.x table, that table is no longer usable by a version older than SPD Server 5.2. SPD Server 5.2 introduced table modifications in support of Hadoop. Any tables updated by a 5.2 or 5.3 server will include these metadata modifications. Table data is not affected by the format upgrade.

SPD Server 5.3 cannot read 32-bit Windows tables. They must be converted as described in "Converting SPD Server 4.x 32-Bit Windows Tables to SPD Server 5.3 64-Bit Windows Tables" on page 30.

The SASSPDS 5.3 engine can connect to any earlier 5.x or 4.x SPD Server. The 5.3 client supports all features of the earlier server. However, the earlier server does not have access to 5.3 client features, such as strong encryption for data at rest.

Converting SPD Server 4.*x* 32-Bit Windows Tables to SPD Server 5.3 64-Bit Windows Tables

Tables that were created with SPD Server 4.*x* running in Windows 32-bit mode are not compatible with SPD Server 5.3 tables in 64-bit mode. SPD Server 5.3 for 64-bit Windows does not recognize SPD Server 4.x components. If you submit a PROC CONTENTS command to SPD Server 5.3 on a Windows 64-bit server, SPD Server 5.3 does not recognize SPD Server 4.*x* tables, views, or clusters, even if they exist in a valid SPD Server domain. There are two ways that you can convert SPD Server 4.*x* tables from a 32-bit Windows environment to SPD Server 5.3 tables for use in a 64-bit Windows environment:

Convert SPD Server 4.x Tables to SAS Tables and Then to SPD Server 5.3 Tables

You can convert SPD Server 4.*x* tables into SAS 9.2 or SAS 9.3 tables by making a LIBNAME connection to your SPD Server 4.*x* server, and then making a LIBNAME connection to SAS. After both LIBNAME connections are established, you can use PROC COPY to copy the tables from your SPD Server domain to your SAS domain.

```
LIBNAME spd44 sasspds "<SPD-Server-4.x-domain-name>"
  host=localhost
  serv=<SPD-Server-4.x-server-port>
  user=<user-ID> ;

LIBNAME SAS "<path-to-SAS9.2-or-SAS9.3-directory>";
PROC COPY in=spd44 out=sas;
```

After you copy your SPD Server 4.*x* tables into SAS 9.2 or SAS 9.3 tables and store them in a SAS location, you can install SAS 9.4 and SPD Server 5.3 on your Windows 64-bit platform. Next, you use the 64-bit Windows environment to create a SAS 9.4 LIBNAME connection to the directory location where you stored your copied SPD Server 4.*x* tables, and create a SAS 9.4 LIBNAME connection to your SPD Server 5.3 domain. Now you can use the SAS 9.4 PROC COPY command to restore the former SPD Server 4.*x* tables to your SPD Server 5.3 domain in SPD Server 5.3 format.

```
LIBNAME spd53 sasspds "<SPD-Server-5.3-domain-name>"
  host=localhost
  serv=<SPD-Server-5.3-server-port>
  user=<user-ID> ;

LIBNAME SAS94 "<path-to-SAS9.4-directory>";

PROC COPY in=SAS94 out=spd53;
```

SPD Server 5.3 will be able to recognize and use the copied tables in the new SAS 9.4 and SPD Server 5.3 domain space.

Convert Directly from SPD Server 4.4x and 4.5x Tables to SPD Server 5.3 Tables

To convert directly from SPD Server 4.4*x* and 4.5*x* tables to SPD Server 5.3 tables, use SAS 9.4 with the SPD Server 4.4*x* or 4.5*x* server and an SPD Server 5.3 server. You must be able to operate both versions of SPD Server in your computing environment in order to use this conversion approach.

First, you make two LIBNAME connections using SAS 9.4. You make the first LIBNAME connection to your SPD Server 4.4*x* or 4.5*x* server. Then you make the second LIBNAME connection to your SPD Server 5.3 server. After you establish a LIBNAME connection between the SPD Server 4.4*x* or 4.5*x* host and the SPD Server 5.3 host, you can use a PROC COPY statement to copy your server tables directly from your 4.4*x* or 4.5*x* host to your SPD Server 5.3 host.

```
LIBNAME spd44 sasspds "<SPD-Server-4.4x-domain-name>"
  host=localhost
  serv=<SPD-Server-4.4.x-server-port>
  user=<user-ID> ;

LIBNAME spd53 sasspds "<SPD-Server-5.3-domain-name>"
  host=localhost
  serv=<SPD-Server-5.3-server-port>
  user=<user-ID> ;

PROC COPY in=spd44 out=spd53;
```

Using PROC COPY with SPD Server Cluster Tables

If you use the PROC COPY command to move tables from one SPD Server domain to another SPD Server domain or SAS location, and if your source domain contains SPD Server cluster tables, the output file that is produced by the PROC COPY command will be either a SAS table or an SPD Server table. The output file will not be a clustered table.

To migrate formerly clustered tables from SPD Server 4.*x* to SPD Server 5.3, you must first undo the table cluster in the source (SPD Server 4.*x*) location. Then copy the component tables to the destination (SPD Server 5.3) location. You can use PROC COPY to copy the unclustered files from the source domain to the destination domain. After you successfully copy your tables to the destination domain, use SPD Server 5.3 CLUSTER TABLE commands to re-create the clustered table structures.

Enabling 32 Groups for SPD Server Users

SPD Server versions 4.*x* and earlier limited SPD Server users to membership in five or fewer SPD Server user groups. SPD Server 5.1 and later support user membership in up to 32 groups. The default user limitation remains at five groups per SPD Server user ID, which suits the majority of SPD Server users. But you can grant users the ability to belong to as many as 32 groups. Users requiring the larger group membership ceiling need to convert their SPD Server group password data sets to SPD Server 5.3 password data sets.

Migrating SPD Server 4.x Password Tables to SPD Server 5.x Password Tables

The SPD Server psmgr utility manages both SPD Server 4.*x* and SPD Server 5.*x* password tables. The psmgr utility independently recognizes both types of table formats and handles all transactions appropriately.

To convert an SPD Server 4.*x* password table to an SPD Server 5.3 password table, use the psmgr EXPORT and IMPORT commands as follows:

1. Back up old password table data sets.
2. Start the SPD Server psmgr utility using your previous SPD Server installation.
3. Export your SPD Server password table to a file.
4. Start your new installation of the SPD Server psmgr utility with a new password table. Include the **`-ver 5`** switch in the command:

   ```
   psmgr <target-directory-name> -ver 5
   ```

5. In your new installation of SPD Server, import the file from the previous installation into the new password table.
6. Verify that the SPD Server start-up scripts point to the newly created password table data sets.

The psmgr Utility GROUPMEM Updates

The SPD Server 5.x psmgr utility provides greater flexibility in managing group memberships than SPD Server 4.x. For example, in SPD Server 4.*x*, if the user specifies fewer than five groups by using the GROUPMEM command, the utility prompts the user to supply additional group names until five group names are provided.

In SPD Server 5.*x*, you have more control over the group list:

- Specify an exclamation point (!) as a GROUPMEM group name list argument to terminate the group list without further interactive user prompting.
- Specify a hyphen (-) as a group list argument to serve as a place holder for a group name that cannot be changed.
- Specify an empty quoted argument (" ") as a group list argument to clear the group in that position.

See documentation for the "GROUPMEM Command" on page 223 for an example of how the new arguments are used.

Part 3

Configuration

Chapter 6
Configuring Secure Sockets Communication *35*

Chapter 7
Managing Passwords and Users *51*

Chapter 8
Configuring Server Domains *59*

Chapter 9
SQL Query Rewrite Facility *79*

Chapter 10
Setting Server Operational Parameters *83*

Chapter 11
Using SPD Server with an Internet Firewall *115*

Chapter 12
Configuring and Administering SPD Server Using the SAS Management Console *119*

Chapter 6
Configuring Secure Sockets Communication

Overview of Securing Client/Server Communication **35**

Introduction to Transport Layer Security (TLS) **36**

Understanding TLS Authentication for SPD Server **36**

Overview of Configuring Server Authentication for SPD Server **36**

Signed Versus Self-Signed Certificates **37**

spdsserv.parm Options for TLS **37**
- Server Authentication Options 37
- Client Authentication Options 39

Configuring SPD Server Clients to Perform Server Authentication **39**

Examples: Creating and Implementing Self-Signed Certificates for SPD Server . **40**
- Overview 40
- Creating the Self-Signed Certificates 40
- Configuring Your UNIX SPD Server and Client to Use the Self-Signed Server Certificates 44
- Configuring Your Windows SPD Server and Client to Use the Self-Signed Server Certificates 44

Monitoring a Secure Sockets Connection **46**

Troubleshooting Secure Sockets Problems **47**
- Overview 47
- Configuring SPDSSERV Logging 47
- Configuring SPDSBASE Logging 48
- Configuring SAS Logging 49

Overview of Securing Client/Server Communication

Beginning with SPD Server 5.3, SPD Server provides secure sockets communication by using Transport Layer Security, the successor to Secure Sockets Layer Security. The secure sockets communication is provided for SAS clients and SAS Federation Server clients only.

Introduction to Transport Layer Security (TLS)

TLS and its predecessor, Secure Sockets Layer (SSL), are cryptographic protocols that are designed to provide over-the-wire communication security. TLS and SSL provide network data privacy, data integrity, and authentication.

TLS uses X.509 certificates and hence asymmetric cryptography to verify the party with whom they are communicating. As a consequence of choosing X.509 certificates, certificate authorities and a public key infrastructure are necessary to verify the relationship between a certificate and its owner, as well as to generate, sign, and administer the validity of certificates.

In addition to providing encryption services, TLS performs client and server authentication, and it uses message authentication codes to ensure data integrity.

Understanding TLS Authentication for SPD Server

TLS authentication can be configured at two levels. At a minimum, TLS specifies that the client should perform server authentication by verifying the server's certificate. This basic configuration is achieved by identifying the location of the certificates and the TLS public key and private key on the server with TLS options. Client authentication, which specifies that the server should also verify the client's certificate, can also be configured, but it is optional.

For SPD Server, the server authentication options affect socket connections for the SPDSSERV process and the SPDSBASE user proxy. The SPDSSERV process authenticates the user ID and password specified in the SPD Server LIBNAME statement. The SPDSBASE user proxy plans and executes SAS DATA step requests made through the SASSPDS LIBNAME engine.

The client authentication options affect socket communications made from the SASSPDS LIBNAME engine to an SPDSBASE process, or communication from one SPDSBASE process to another SPDSBASE process. These processes plan and execute SQL queries made using implicit and explicit pass-through to SPD Server. For more information about SPDSBASE processes, see Chapter 23, "Managing SPDSBASE Processes," on page 205.

To ensure secure sockets communication for all requests, configure both server and client authentication options.

Overview of Configuring Server Authentication for SPD Server

The components that you need to configure secure sockets communication on SPD Server are delivered when you install SAS/SECURE Toolkit Libraries from RSA and SAS/SECURE SSL software. These components are delivered as part of the SPD Server installation.

The TLS configuration process is different for UNIX and Windows SPD Servers. To take advantage of TLS on a UNIX SPD Server, you must set up digital certificates; add the certificates to the trusted CA bundle that is configured for you by the SAS Deployment Manager; and configure the locations of the trusted CA bundle and TLS public and private keys on SPD Server. These items are configured by adding TLS options to the server's spdsserv.parm parameter file.

For Windows, SPD Server uses the SChannel library that comes with the Windows operating system. You must decide whether you want to request a digital certificate from the Microsoft Certificate Authority or request a digital certificate from a certificate authority that is not Microsoft. Once you have decided which certificates to use, you must ensure that the server certificate is in the server host's Windows Certificate Store. You must also configure the certificates with SSL options in the Windows server's spdsserv.parm server parameter file.

The process for setting up digital certificates is described in detail for both UNIX and Windows SAS servers in *Encryption in SAS*. See "Part 2. Installing and Configuring TLS and Certificates". Some of the configuration options that are described in Part 1 of *Encryption in SAS* do not apply to SPD Server. Use the information about SSL options for SPD Server in this chapter to configure secure sockets communication for SPD Server and its clients.

Signed Versus Self-Signed Certificates

Secure sockets communication can be configured using a signed certificate or a self-signed certificate. A signed certificate is an authorized certificate that is issued by a trustworthy certificate authority. A self-signed certificate is signed by the same entity whose identity it certifies. Both a signed and self-signed certificate will encrypt data communication. Using a signed certificate authority tells a customer the server information has been verified by a trusted store.

Self-signed certificates are useful for testing secure communication. They are adequate for production use in cases where encryption of the data is required, but authorization of access to the network is not required. In many installations, SPD Server is installed on an internal company network where only encryption is needed. For these cases, a self-signed certificate can be used to satisfy secure requirements.

This chapter includes examples of how to create and implement self-signed certificates to secure communication between an SPD Server and its clients. The examples use the OpenSSL cryptography toolkit. For more information, see "Examples: Creating and Implementing Self-Signed Certificates for SPD Server" on page 40.

spdsserv.parm Options for TLS

The following options are supported in the spdsserv.parm server parameter file to configure secure sockets communication on SPD Server. The options are dependent on the host environment. The server authentication options must be set to enable basic secure sockets communication.

Server Authentication Options

The following option enables secure sockets communication on both UNIX and Windows SPD Server hosts.

SSLSECURE= NO | PREFERRED | YES
: Specifies how TLS is used by SPD Server. The default value is NO, which specifies that client connections are not secured. To enable secure sockets communication, specify YES or PREFERRED. YES specifies that the server requires a secure client connection. A client that is not configured for TLS cannot connect to SPD Server. PREFERRED specifies that a secure connection is made if the client has TLS configured. Clients that are not configured for TLS can connect, but these connections are not secure. For more information, see "SSLSECURE= Parameter File Option" on page 111.

UNIX:

These options configure the certificates, public key, and private key on a UNIX SPD Server host:

SSLALLOWUNXDS
: Specifies whether TLS or UNIX domain sockets are used to secure data communication between clients that are on the same host as SPD Server. The default value, NOSSLALLOWUNXDS, specifies to use TLS for both local and remote client/server data communications. For more information, see "SSLALLOWUNXDS Parameter File Option" on page 103.

SSLCALISTLOC=
: Specifies the location of the public certificate or certificates for trusted certificate authorities (CA). This option is required to enable TLS.For more information, see "SSLCALISTLOC= Parameter File Option" on page 104.

SSLCERTLOC=
: Specifies the location of the digital certificate for the machine's public key. This option is required to enable TLS. For more information, see "SSLCERTLOC= Parameter File Option" on page 105.

SSLPKCS12LOC=
: (Optional) Specifies the location of the PKCS #12 encoding package file. When SSLPKCS12LOC= is set, SSLCERTLOC= and SSLPVTKEYLOC= are ignored. For more information, see "SSLPKCS12LOC= Parameter File Option" on page 108.

SSLPKCS12PASS=
: Used with SSLPKCS12LOC, specifies the password that TLS requires for decrypting the private key. For more information, see "SSLPKCS12PASS= Parameter File Option" on page 109.

SSLPVTKEYLOC=
: Specifies the location of the private key that corresponds to the digital certificate. This option is required to enable TLS. For more information, see "SSLPVTKEYLOC= Parameter File Option" on page 109.

SSLPVTKEYPASS=
: Used with SSLPVTKEYLOC, specifies the password that TLS requires for decrypting the private key. For more information, see "SSLPVTKEYPASS= Parameter File Option" on page 110.

Windows:

These options configure certificates for a Windows SPD Server host:

SSLCERTISS=
: Specifies the name of the issuer of the digital certificate that TLS should use. For more information, see "SSLCERTISS= Parameter File Option" on page 105.

SSLCERTSERIAL=
: Used with SSLCERTISS=, specifies the serial number of the digital certificate that TLS should use. For more information, see "SSLCERTSERIAL= Parameter File Option" on page 106.

SSLCERTSUBJ=
: Specifies the subject name of the digital certificate that TLS should use. Use SSLCERTISS= and SSLCERTSERIAL= or SSLCERTSUBJ=. Do not specify both. For more information, see "SSLCERTSUBJ= Parameter File Option" on page 106.

Client Authentication Options

SSLCLIENTAUTH | NOSSLCLIENTAUTH.
: Specifies whether the server should verify the client's certificate in addition to the server's certificate. The default is NOSSLCLIENTAUTH. For more information, see "SSLCLIENTAUTH Parameter File Option" on page 107.

SSLCRLCHECK
: Specifies to check a Certificate Revocation List (CRL) when a digital certificate is validated. The default value is NOSSLCRLCHECK. For more information, see "SSLCRLCHECK Parameter File Option" on page 107.

SSLCRLLOC=
: Specifies the location of a Certificate Revocation List (CRL). Used in conjunction with SSLCRLCHECK. For more information, see "SSLCRLLOC= Parameter File Option" on page 108.

SSLREQCERT=
: Specifies the protocol for exchanging digital certificates at your site. The valid values are ALLOW, DEMAND, NEVER, or TRY. Used in conjunction with SSLCLIENTAUTH. For more information, see "SSLREQCERT= Parameter File Option" on page 110.

Configuring SPD Server Clients to Perform Server Authentication

Configuring an SPD Server SAS client for secure sockets communication is a two-step process:

1. The client computer must be made aware of the public certificate or trusted CA bundle. A SAS client that runs on a UNIX computer can specify the path to the public certificate or trusted CA bundle by setting the SSLCALISTLOC= system option in the SAS session. For more information about this system option, see *Encryption in SAS*. A Windows client must add the CA to the client computer's Certificate Store.

2. Secure sockets communication must be enabled for SPD Server in the SAS session. To enable secure sockets communication for SPD Server, set the SPDSRSSL= macro variable before assigning a SASSPDS LIBNAME statement. This macro variable has two settings: YES and NO. NO is the default value. YES enables a client to make a secure sockets connection to an SPD Server that is configured with SSLSECURE=YES or SSLSECURE=PREFERRED. Regardless of the setting of SPDSRSSL=, the client will successfully connect to an older server (SPD Server 5.2

or earlier) with non-secure sockets. For more information, see "SPDSRSSL Macro Variable" in *SAS Scalable Performance Data Server: User's Guide*.

A SAS Federation Server client sets the SPDSRSSL= environment variable to enable a secure connection to SPD Server. For more information about configuring secure sockets communication for SAS Federation Server, see the *SAS Federation Server: Administrator's Guide*.

Examples: Creating and Implementing Self-Signed Certificates for SPD Server

Overview

These examples use OpenSSL on the Linux platform to create self-signed server certificates for secure sockets communication between an SPD Server and an SPD Server SAS client. OpenSSL enables you to create self-signed server certificates for any UNIX or Windows configuration. This example does not describe OpenSSL. For information about OpenSSL, see https://www.openssl.org/docs/.

A self-signed server certificate includes three pieces: a Certificate of Authority (CA), a server certificate, and a server private key file. The CA is referred to as the client certificate. The CA enables the client to initiate secure sockets communication with the server. The server accepts the communication and uses SSL options to validate the client certificate. The server then uses SSL options to locate the server certificates. A CA can be used on any client on any host. The server certificate and key file are host-specific.

This example creates a single client certificate. Then it shows how to create self-signed server certificates for a UNIX SPD Server host and a Windows SPD Server host that are signed with the client certificate. Finally, it shows how to implement the client and server certificates in their respective host environments. The host-specific server certificates are created on the same Linux host as the client certificate.

The certificates in these examples are created for the following organization:

Company information:
Name: SSL Testing

Country: US

State: North Carolina

Locality: Cary

Organization: ABCD

Common Name: Bob Smith

Creating the Self-Signed Certificates

Client Certificate

Follow these steps to create the client certificate. To follow along, modify the information for your SPD Server installation. This client certificate can be used for both a Windows client and a UNIX client

1. On the Linux system, create working directory “myssl” to create your certificates. The working directory will contain the client and server certificates and enable you to modify the default openssl.cnf configuration file. Change your directory to your myssl working directory for all further work.

2. Make the following OpenSSL subdirectories and the flat file database files to keep track of signed certificates:

   ```
   >mkdir certs  crl newcerts private
   >touch index.txt
   >echo 1000 > serial
   ```

3. Locate where OpenSSL is installed on your computer. The location is /etc/pki/tls for this example:

   ```
   >openssl version -d
   OPENSSLDIR: "/etc/pki/tls"
   ```

4. Copy the OpenSSL configuration file to your myssl working directory and modify the configuration file for your environment.

   ```
   > cp /etc/pki/tls/openssl.cnf .
   ```

   Edit the openssl.cnf file as follows:

   - Modify the line containing “Where everything is kept” to be the full path to your myssl working directory.
   - Modify the line containing “default_md = sha1” to “default_md = sha256”.

5. You are now ready to create certificates in your myssl working directory. First, generate a private key for your Certificate of Authority. The name of the private key file for this example is “private/TestingCA.key.pem”.

   ```
   >openssl genrsa -aes256 -out private/TestingCA.key.pem 4096
   ```

   You will be prompted to enter and then verify a pass phrase. Enter a secret password and save it for later use.

   ```
   Enter pass phrase for private/TestingCA.key.pem:
   ```

6. Next, execute the following command to create the client certificate with your private key file. The name of the client certificate is “certs/TestingCA.pem”. This will be the “root” CA.

   ```
   > openssl req -config openssl.cnf -key private/TestingCA.key.pem -new -x509
   -days 7300 -sha256 -extensions v3_ca -out certs/TestingCA.pem
   ```

   You will be asked to enter the following information. Substitute information for your configuration. For the pass phrase, enter your secret password from step 5.

   TIP Continue to save this password. You will need it later.

   ```
      Enter pass phrase for private/TestingCA.key.pem:
      Country Name (2 letter code) [XX]:US
      State or Province Name (full name) []:North Carolina
      Locality Name (eg, city) [Default City]:Cary
      Organization Name (eg, company) [Default Company Ltd]:SSL Testing
      Organizational Unit Name (eg, section) []:ABCD
      Common Name (eg, your name or your server's hostname) []:Bob Smith
      Email Address []: <cr>
   ```

   You have finished creating the client certificate. You should have the following client certificate in your myssl working directory:

   ```
   certs/TestingCA.pem
   ```

You can look at this “root” CA using the following command:

```
> openssl x509 -noout -text -in certs/TestingCA.pem
```

Server Certificates for a UNIX SPD Server

After creating the client certificate, follow these steps to create a server certificate and server key file for a UNIX SPD Server.

Assume the following SPD Server server installation:
Host name: laxbox.unx.sslt.com

1. Using OpenSSL in the myssl working directory, submit a certificate request to generate a new key for the server certificate. Name the server key file “private/TestingServerLax.key.pem”. Name the request file “TestingServerLax.req”.

   ```
   >openssl req -config openssl.cnf -new -newkey rsa:1024
   -keyout private/TestingServerLax.key.pem -out TestingServerLax.req -nodes
   ```

   You will be asked to enter the following information. Substitute information for your configuration.

   ```
      Country Name (2 letter code) [XX]:US
      State or Province Name (full name) []:North Carolina
      Locality Name (eg, city) [Default City]:Cary
      Organization Name (eg, company) [Default Company Ltd]:SSL Testing
      Organizational Unit Name (eg, section) []:ABCD
      Common Name (eg, your name or your server's hostname) []:laxbox.unx.sslt.com
      Email Address []: <cr>
      Please enter the following 'extra' attributes
      A challenge password []: <cr>
      An optional company name []:<cr>
   ```

2. Create (sign) the new server certificate with the “root” CA and the certificate request.

   ```
   >openssl ca -config openssl.cnf -cert certs/TestingCA.pem
   -keyfile private/TestingCA.key.pem -in TestingServerLax.req
   -out certs/TestingServerLax.pem
   ```

   You will be asked to enter a pass phrase. Enter the secret password that you used to create your client certificate.

   ```
   Enter pass phrase for private/TestingCA.key.pem:
   ```

   If everything is in order, you will be prompted as follows. Enter Y at each prompt to sign and commit the certificate.

   ```
   Sign the certificate? [y/n]:y
   1 out of 1 certificate requests certified, commit? [y/n]y
   Write out database with 1 new entries
   Data Base Updated
   ```

3. Verify the server certificate as follows:

   ```
   > openssl verify -verbose -CAfile certs/TestingCA.pem certs/TestingServerLax.pem
   ```

You have finished creating the self-signed certificates. You should have the following server certificate and private server key file in your myssl working directory:

```
certs/TestingServerLax.pem
private/TestingServerLax.key.pem
```

Note: Your server certificate can be used only if your server is running on laxbox.unx.sslt.com. You will need additional server certificates that use the same "root" CA for any other server hosts in your configuration. Repeat the steps above for any additional UNIX SPD Server hosts in your configuration.

Server Certificates for a Windows SPD Server

The process for creating self-signed certificates for Windows is the same as it is for UNIX with one difference. Unlike a server certificate for UNIX, a server certificate for Windows must contain the server certificate and server private key file in the same file. This file is called a PKCS#12 file. The file extension for PKCS#12 files is .p12 or .pfx. You can use OpenSSL to convert a UNIX server certificate and key file into a PKCS#12 file.

Follow these steps to create the self-signed server certificates for the Windows SPD Server.

Assume the following SPD Server installation:
Host name: winbox.win.sslt.com

1. Using OpenSSL in the myssl working directory, submit a certificate request to generate a new key for the Windows server certificate. Name the server key file "private/TestingServerWin.key.pem". Name the request file "TestingServerWin.req".

   ```
   >openssl req -config openssl.cnf -new -newkey rsa:1024
   -keyout private/TestingServerWin.key.pem -out TestingServerWin.req -nodes
   ```

   You will be asked to enter the following information. Substitute information for your configuration.

   ```
      Country Name (2 letter code) [XX]:US
      State or Province Name (full name) []:North Carolina
      Locality Name (eg, city) [Default City]:Cary
      Organization Name (eg, company) [Default Company Ltd]:SSL Testing
      Organizational Unit Name (eg, section) []:ABCD
      Common Name (eg, your name or your server's hostname) []:winbox.win.sslt.com
      Email Address []: <cr>
      Please enter the following 'extra' attributes
      A challenge password []: <cr>
      An optional company name []:<cr>
   ```

2. Create (sign) the new server certificate with the "root" CA and the certificate request.

   ```
   >openssl ca -config openssl.cnf -cert certs/TestingCA.pem
   -keyfile private/TestingCA.key.pem -in TestingServerWin.req
   -out certs/TestingServerWin.pem
   ```

   You will be asked to enter a pass phrase. Enter the secret password that you used to create your client certificate.

   ```
   Enter pass phrase for private/TestingCA.key.pem:
   ```

   If everything is in order, you will be prompted as follows. Enter Y at each prompt to sign and commit the certificate.

   ```
   Sign the certificate? [y/n]:y
   1 out of 1 certificate requests certified, commit? [y/n]y
   Write out database with 1 new entries
   Data Base Updated
   ```

3. Verify the server certificate as follows:

   ```
   > openssl verify -verbose -CAfile certs/TestingCA.pem certs/TestingServerWin.pem
   ```

Note: Your server certificate can only be used if your server is running on winbox.win.sslt.com. You will need additional server certificates that use the same “root” CA for any other server hosts in your configuration. Repeat the steps above for any additional Windows server hosts in your configuration.

Use the following OpenSSL command to create a .p12 file that contains the Windows server certificate and key file. The command creates a .p12 file named “mywin.p12”.

```
>openssl pkcs12 -export -in certs/TestingServerWin.pem
-inkey private/TestingServerWin.key.pem -export -out mywin.p12
```

You will be asked to enter and verify an export password as follows. Enter any secret password and save it for later use.

```
Enter Export Password:
```

Configuring Your UNIX SPD Server and Client to Use the Self-Signed Server Certificates

Configuring a UNIX Server Certificate

To configure a UNIX SPD Server to use your self-signed certificates, set the following TLS options in the spdsserv.parm server parameter file:

```
    SSLSECURE="YES";
    SSLCALISTLOC="<full-path-to-myssl>/certs/TestingCA.pem";
    SSLCERTLOC="<full-path-to-myssl>/certs/TestingServerLax.pem";
    SSLPVTKEYLOC="<full-path-to-myssl>/private/TestingServerLax.key.pem";
```

SSLSECURE=YES instructs the server to accept secure connections only. SSLCALISTLOC= specifies the path to the client certificate. SSLCERTLOC= specifies the path to the server certificate. SSLPVTKEYLOC= specifies the path to server’s private key file.

Configuring a UNIX Client Certificate

To configure an SPD Server SAS client that runs on UNIX to make a secure connection, set the following options in the SAS session. Set the options before you assign an SASSPDS LIBNAME statement:

```
options set=SSLCALISTLOC="<full-path-to-myssl>/certs/TestingCA.pem";
%let SPDSRSSL=YES;
```

The SSLCALISTLOC= system option specifies the path to the client certificate. The SPDSRSSL= macro variable requests that a secure connection be made.

Configuring Your Windows SPD Server and Client to Use the Self-Signed Server Certificates

Windows servers and clients use the Windows Certificate Store to locate the server and client certificates.

Configuring a Windows Server Certificate

This example uses the Microsoft Management Console (MMC) on a Windows 10 computer to register the server certificate.

1. Copy the mywin.p12 file to a location that is accessible to your Windows SPD Server.

2. Add the mywin.p12 file to the Certificate Store as a private certificate as follows:
   a. Run “mmc.exe” to open the console.
   b. Select **File ⇨ Add/Remove Snap-in**.
   c. Under **Available snap-ins**, highlight **Certificates**, and click **Add**.
   d. Select **Computer account**. Click **Next**.
   e. Select **Local computer**. Click **Finish**.
   f. In the Add or Remove Snap-ins window, click **OK**.
   g. In the Console window, select **Certificates (Local Computer)** in the **Console Root** panel.
   h. Under **Logical Store Name**, right-click **Personal ⇨ All Tasks ⇨ Import**.
   i. In the Certificate Import wizard, click **Next**.
   j. In the File to Import window, enter the full path to your mywin.p12 file in **File Name**. Or click **Browse** to locate the file.Then, click **Next**.
   k. You will be prompted for a password. Enter your export password. Check **Include all extended properties**. Then, click **Next**.
   l. In the Certificate Store, the following options should be highlighted:
      - **Place all certificates in the following store**
      - **certificate store: Personal**

      Click **Next**.
   m. Verify the information in Completing the Certificate Import Wizard Settings. Click **Finish**.
   n. A dialog box titled “The import was successful” indicates that you have successfully added the server PKCS#12 file. Click **OK**.
   o. Double-click first **Personal**, then **Certificates** under **Logical Store Name** to see your certificate. The name should be “winbox.win.sslt.com”.
   p. Close the console. You do not have to select **Save the console settings**. You have finished adding your server certificate to the Certificate Store.

To configure your Windows SPD Server to use the server certificate, set the following TLS options in the spdsserv.parm server parameter file:

```
SSLSECURE="YES";
SSLCERTSUBJ="winbox.win.sslt.com"
```

SSLSECURE=YES instructs the server to accept secure connections only. SSLCERTSUBJ= is set to the name of your personal server certificate (winbox.winsslt.com).

Configuring a Windows Client Certificate

Follow these steps to add the client certificate to the Certificate Store.

1. Copy your certs/TestingCA.pem file to a location that is accessible to your Windows client computer.
2. Add the client certificate to your Trusted Root Certificate Authorities using the MMC as follows:
   a. Run “mmc.exe” on your PC to open the console.

b. Select **File** ⇨ **Add/Remove Snap-in**.

c. Under **Available snap-ins**, highlight **Certificates**. Then, click **Add**.

d. Select **Computer account**. Click **Next**.

e. Select **Local computer**. Click **Finish**.

f. In the Add or Remove Snapins window, click **OK**.

g. Back on the Console window, select **Certificates (Local Computer)** in the **Console Root** panel.

h. Under **Logical Store Name**, double-click **Trusted Root Certification Authorities**.

i. Under **Object Type**, right-click **Certificates** ⇨ **All Tasks** ⇨ **Import**.

j. In the Certificate Import wizard, click **Next**.

k. In the File to Import window, enter the full path to where you copied TestingCA.pem in **File Name**. Or, click **Browse** to locate the file. Then, click **Next**.

l. In the Certificate Store, the following options should be highlighted:

   - **Place all certificates in the following store**
   - **certificate store: Trusted Root Certification Authorities**

   Click **Next**.

m. Verify the information in Completing the Certificate Import Wizard Settings. Click **Finish**.

n. A dialog box titled The import was successful indicates that you have successfully added the client certificate. Click **OK**.

o. Double-click first **Trusted Root Certification**, and then **Certificates** under **Logical Store Name** to see your certificate. The name should be the common name of the certificate specified in step Step 6 on page 41 when you created your certificate.

p. Close the console. You do not have to select **Save the console settings**. You have finished adding the self-signed client certificate to the Certificate Store.

To configure the SAS session on a Windows computer to make a secure connection to SPD Server, set the SPDSRSSL macro variable. Set the macro variable before you assign an SASSPDS LIBNAME statement.

```
%let SPDSRSSL=YES;
```

Monitoring a Secure Sockets Connection

If you have successfully configured your client and server to run secure, a message similar to the following is written to the SAS log after you issue a LIBNAME statement that specifies the SASSPDS engine:

```
? libname foo sasspds 'test' host='laxbox.unx.sslt.com' serv='5400' user='anonymous';
NOTE: This is a SPD 5.3  Engine
      executing SAS (r) 9.4 (TS1M3) on the Linux platform.
NOTE: User anonymous(ACL Group ) connected to Secure SPD(LAX) 5.3 server at
```

```
      10.24.7.79.
NOTE: Libref FOO was successfully assigned as follows:
      Engine:        SASSPDS
      Physical Name::16121/data/testing
```

Troubleshooting Secure Sockets Problems

Overview

SPD Server supports use of the SAS logging facility for troubleshooting problems with secure sockets. The SAS logging facility is a framework that categorizes and filters log messages in SAS server and programming environments. It also writes log messages to various output devices. The SAS logging facility provides diagnostic information beyond that provided by SPD Server logging. We recommend that you reserve use of the logging facility for diagnosing problems, because the logging facility generates a lot of information.

The SAS logging facility is activated by creating an XML configuration that specifies the desired loggers and then by referencing this configuration file at server or program start-up. SPD Server supports the App.tk.tcp logger and App.tk.eam.ssl logger for monitoring secure sockets activity on the client and server. It also supports the Encryption loggers described in *Encryption in SAS*.

There are two scenarios for troubleshooting communication between the SASSPDS client and SPD Server:

- communication between the SASSPDS client and the SPDSSERV process to establish the LIBNAME connection.
- communication between the SASSPDS client and the SPDSBASE process after the LIBNAME connection is established.

To enable diagnostic logging for the first scenario, you must configure SPDSSERV logging and SAS logging. To enable diagnostic logging for the second scenario, you must configure SPDSBASE logging and SAS logging. This is accomplished by creating separate XML configuration files for the SPDSSERV process, SPDSBASE processes, and the SAS session. These log files use the same loggers, but specify a different FileNamePattern to distinguish the log files that they create. The logging facility enables messages to be filtered on the following sensitivity levels: TRACE, DEBUG, INFO, WARN, ERROR, and FATAL. To monitor secure sockets activity, specify the loggers with a TRACE setting.

Configuring SPDSSERV Logging

1. Create a logging configuration file named logcfgspdsserv.xml. In the configuration file, specify the App.tk.tcp and App.tk.eam.ssl loggers. Specify the SPDSSERV process in the FileNamePattern.

   Here is an example of a logconfig.xml file that specifies the App.tk.tcp and App.tk.eam.ssl loggers and specifies a FileNamePattern for the SPDSSERV process:

```
<?xml version="1.0"?>
<logging:configuration xmlns:logging="http://www.sas.com/xml/logging/1.0/">
   <!-
      Configure a FileAppender instance for our server log.
   ->
   <appender name="RootAppender" class="FileAppender">
      <!-
         This is where we want the SPD Server log files to be created and
         their template name.  The name will be spdsbase_log4sas_YYYY-MM-DD_<PID>.log
      ->
      <param name="FileNamePattern" value="/u/foobar/spds53/spdsserv_log4sas_%d_%S{jobid}.log"/>
      <param name="Append" value="false"/>
      <param name="ImmediateFlush" value="true"/>
      <!-
         We want to use this as the message layout
      ->
    <layout>
        <param name="conversionpattern" value="%r %d{HH:mm:ss.SSS} thr[%t] %15.15F:%-6L - %m"/>
    </layout>
   </appender>
   <!-
      Logging is enabled for events emitted in the "App.tk.tcp" TKITCP extension.
      Logging levels available are Fatal, Error, Warn, Info, Debug, and Trace.
      The logging level enabled is Trace.
   ->
   <logger name="App.tk.tcp">
      <level value="trace"/>
   </logger>
   <logger name="App.tk.eam.ssl">
       <level value="trace">
   </logger>

   <root>
      <appender-ref ref="RootAppender"/>
   </root>
</logging:configuration>
```

2. Modify the SPD Server start-up script to add the path to the logcfgspdsserv.xml file as a parameter to the SPDSSERV process as follows. For UNIX, modify the site/rc.spds script. For Windows, modify the site\spdsserv.bat file. The LOGCONFIGLOC option specifies the path to the configuration file.

   `spdsserv -logconfigloc <full-path>/logcfgspdsserv.xml --`

   Note: The LOGCONFIGLOC option must be the first parameter in the start-up script. The option value must include the double hyphens (--).

A log file will be created for the SPDSSERV process in directory **`/u/foobar/spds53`**. The log file will be named spdsserv_log4sas_YYYY-MM-DD_<PID>.log.

Configuring SPDSBASE Logging

1. Create a logging configuration file named logcfgspdsbase.xml. In the configuration file, specify the App.tk.tcp and App.tk.eam.ssl loggers and a FileNamePattern for the SPDSBASE process. The file can have the same content as the logcfgspdsserv.xml file, except for the FileNamePattern.

   Here is an example of how you might specify the FileNamePattern:

   ```
   param name="FileNamePattern" value="/u/foobar/spds53/spdsbase_log4sas_%d_%S{jobid}.log"
   ```

2. Export the path to the configuration file with the SPDSLOG4SASCONFIGLOC= environment variable. For a UNIX server, modify the site/rc.spds script to add the SPDSLOG4SASCONFIGLOC= environment variable. For a Windows server,

modify the site\spdsserv.bat file. Specify the SPDSLOG4SASCONFIGLOC= environment variable as follows:

```
export SPDSLOG4SASCONFIGLOC=<full-path>/logcfgspdsbase.xml
```

A log file will be created for each SPDSBASE process in directory **`/u/foobar/spds53`**. The log file will be named spdsbase_log4sas_YYYY-MM-DD_<PID>.log.

Configuring SAS Logging

1. Create a logging configuration file named logcfgsas.xml. In the configuration file, specify the App.tk.tcp and App.tk.eam.ssl loggers and a FileNamePattern for the SAS session. The file can have the same content as the logcfgspdsserv.xml file, except for the FileNamePattern.

   Here is an example of how you might specify the FileNamePattern:

   ```
   param name="FileNamePattern" value="/u/foobar/spds53/sas_log4sas_%d_%S{jobid}.log"
   ```

2. Specify the -LOGCONFIGLOC option in the SAS start-up command as follows:

   ```
   -logconfigloc <full-path>/logcfgsas.xml
   ```

A log file will be created for the SAS process in directory **`/u/foobar/spds53`**. The log file will be named sas_log4sas_YYYY-MM-DD_<PID>.log.

For more information about the SAS logging facility, see *SAS Logging: Configuration and Programming Reference*.

Chapter 7

Managing Passwords and Users

Overview of SPD Server Authentication **51**

Understanding Native Authentication **52**
Overview 52

Understanding Alternative Methods of Authentication **52**
Overview 52
How SPD Server Performs Alternative Authentication 53
Understanding LDAP Authentication 53
Understanding SAS Metadata Server Authentication 54

Configuring Nonnative Authentication in SPD Server **55**
Overview 55
Direct LDAP Configuration Options 55
SAS Metadata Server Authentication Options 56
Configuring LDAP Authentication through SAS Metadata Server 57

Creating SPD Server User Accounts **57**

Anonymous User ID **57**

Overview of SPD Server Authentication

SPD Server supports native authentication through the SPD Server password database and nonnative authentication through LDAP (Lightweight Directory Access Protocol) or the SAS Metadata Server. SAS Metadata Server authentication provides improved integration with other SAS products, and can be configured to use the authentication provider of choice (such as Active Directory or LDAP).

When nonnative authentication is used, SPD Server users still need to be registered in the password database. This password database stores information about user access rights, groups, and group memberships. It also provides the default authentication and password management services.

Psmgr is an SPD Server command-line utility that you can use to modify the password database.

Understanding Native Authentication

Overview

Native authentication refers to an internal SPD Server user-authentication process that accesses the SPD Server password database. This form of authentication is configured by using the psmgr utility.

By default, whenever SPD Server needs to authenticate a user's identity, it accesses the password database. This database contains relevant information about SPD Server users, such as the following:

- SPD Server user ID
- SPD Server password
- ACL groups
- ACL group memberships
- Special user privileges
- Optional performance level
- Date and time of last login
- Number of consecutive failed login attempts
- Account expiration information.

In the native authentication configuration, SPD Server validates the user by passing the user name from the SASSPDS LIBNAME statement to the password database. This database contains all information pertaining to the user, including permissions and account management information. After validation, SPD Server allows access to the tables, groups, and rights that are appropriate for each user.

To use native SPD Server authentication only, do not include any authentication configuration options in the spdsserv.parm parameter file.

Understanding Alternative Methods of Authentication

Overview

An enterprise computing environment usually contains a collection of business software solutions that are sold as a platform, applied across an organization, and further customized to specific areas. Platform services come with their own authentication processes and password management systems. When SPD Server is added to such an environment, it brings its own authentication process and password management system. This means that the administrator has to maintain two sets of user IDs: one for SPD Server and another for the platform services. In addition, disparate user ID and password management tools can have incongruous requirements. For example, SPD Server password lengths must be 6–8 characters. But the password length requirements for a given enterprise platform can range anywhere from 6 characters to 64 or more characters, allowing for stronger passwords.

One solution is to leave all SPD Server authentication to the enterprise platform. By integrating SPD Server user IDs and passwords with the framework of the platform's authenticator, the SPD Server administrator can maintain only one set of user IDs and passwords.

Configuring an alternative form of authentication involves adding appropriate options to the spdsserv.parm parameter file.

How SPD Server Performs Alternative Authentication

SPD Server performs alternative authentication in the following way:

1. The SPD Server user issues a SASSPDS LIBNAME statement from SAS.
2. The SPD Server client passes the request to the SPD Server host, including user name and password information provided in the body of the LIBNAME statement.
3. SPD Server forwards the request to the alternative authenticator. The alternative authenticator (for example, LDAP or SAS Metadata Server) verifies whether the user account is in good standing. It also ensures that all authentication policies (such as a password expiration date) are in force.
4. If the alternative authenticator finds the user account is in good standing, the SPD Server host is signaled. The SPD Server host then looks up the user in the SPD Server password database to apply attributes specific to SPD Server (such as access rights, groups, and group memberships) to the user’s permissions.
5. Following authentication, SPD Server allows the user access and rights as indicated by the information contained in the password database.

Understanding LDAP Authentication

To use direct LDAP authentication, an LDAP server that performs LDAP authentication must be running on the SPD Server host. When you use this direct LDAP authentication, the operating system handles password maintenance. LDAP authentication has the added benefit of operating-system-level security and convenience.

On SPD Server startup, an LDAP configuration in the spdsserv.parm parameter file signals the SPD Server host to use LDAP authentication. SPD Server sends the LDAP server a DN (Distinguished Name), which consists of the user ID. LDAP begins the process of validation and setting access accordingly. LDAP authentication levels can range from **anonymous authentication**, which gives the least amount of access to information, to **administrator authentication**, which gives a user complete access. After LDAP settings are accessed, SPD Server grants user access according to the protocols set in the password database.

When you use an LDAP server to perform SPD Server user authentication, keep the following facts in mind:

- You still need to register users with the psmgr utility. A password database record is required for each SPD Server user. SPD Server uses the password database to perform user access control tasks and other tasks that are not related to user password authentication.
- Users that connect to SPD Server must have corresponding logon information in the LDAP server. The LDAP server user ID and the SPD Server user ID formats are the same. The logon password format is the host-operating-system format.

- You must enter an initial password in the psmgr utility when you are adding a new user. This password is never used. It simply enables you to add the new user. The user will connect with his LDAP password. The user is not required to use the NEWPASSWD= or CHANGEPASS=YES LIBNAME option to use the LDAP password.
- Some LDAP server products might require users to enter host logon information. In these cases, confirm with your LDAP server administrator that the host logon information exists in the LDAP database.
- If you are using LDAP user authentication and a user connection specifies the NEWPASSWORD= LIBNAME option, the user password is not changed. Follow the operating system procedures to change a user password. Be sure to check with your LDAP server administrator to ensure that the LDAP database records the password changes. The same process information applies to other nonnative authenticators as well.

Understanding SAS Metadata Server Authentication

In order to use SAS Metadata Server Authentication, you must create metadata definitions for each SPD Server user on the SAS Metadata Server, if they do not already exist. You create user definitions in SAS Management Console. Once stored on the SAS Metadata Server, the user definitions can be used by other enterprise applications. SAS Metadata Server uses the authentication provider that is specified in its configuration to perform the authentication. In this case, SPD Server passes the authentication request to the back-end authenticator via the SAS Metadata Server.

On SPD Server start-up, SAS Metadata Server configuration options in the spdsserv.parm parameter file signal the SPD Server host to authenticate users through SAS Metadata Server. SPD Server passes the user name and password from the SASSPDS LIBNAME statement to the SAS Metadata Server for validation. After the SAS Metadata Server validates the user account, SPD Server then accesses its internal password database file to determine other attributes belonging to the user.

When you use SAS Metadata Server to perform SPD Server user authentication, an entry is still required for each server user in the password database. The SPD Server password database is managed by the psmgr utility. Each user entry in the database provides non-authentication information, such as SPD Server group memberships, user performance levels, ACL privileges, and so on.

The benefits of using SAS Metadata Server authentication include the ability to use longer passwords than supported by the native SPD Server authentication. SPD Server has a native password length limit of 8 characters. The password length limit when using a non-native authentication via SAS Metadata Server is defined by the back-end authenticator. This often provides access to longer and more secure passwords. SAS Metadata Server also provides better support for using LDAP as a back-end authentication provider than direct LDAP authentication. The newer SAS Metadata Server versions provide higher levels of encryption, better integration, support, and documentation.

Configuring Nonnative Authentication in SPD Server

Overview

To use a nonnative authentication provider, you include options that identify the nonnative provider in the spdsserv.parm parameter file. Only one authenticator should be configured in the spdsserv.parm parameter file. For example, include options that configure either direct authentication through LDAP or options that configure authentication through SAS Metadata Server. SAS Metadata Server provides a choice of external authenticators, including LDAP. Do not include options that configure both. Doing so can cause undesirable results.

Most server parameters can be changed and refreshed while the SPD Server host is running. However, the collection of server parameter options that begin with LDAP* and META* should not be changed or modified while SPD Server is running.

CAUTION:

Changing LDAP* and META* property settings without first shutting down SPD Server can cause unpredictable results. To modify LDAP* and META* options in the spdsserv.parm parameter file, you must first shut down SPD Server, make your parameter file configuration changes, and then restart SPD Server.

Direct LDAP Configuration Options

To configure direct LDAP authentication for SPD Server:

- Register users in the SPD Server password database with the psmgr utility.
- Make sure that the SPD Server user IDs match the user's LDAP user IDs. The passwords are not required to match. The LDAP password will be used.
- Include LDAP parameter options in the spdsserv.parm parameter file.

The LDAP parameter options for the spdsserv.parm parameter file are as follows:

LDAP
: Turns on LDAP authentication. If the LDAP parameter is found during start-up, SPD Server creates a context for LDAP authentication. The default setting is NOLDAP.

LDAPSERVER=
: Specify a valid IP address of the host machine for the LDAP server. This address is usually the same as the IP address of the SPD Server host. The default value is the IP address of the SPD Server host.

LDAPPORT=
: Specify the TCP/IP port that is used to communicate with the LDAP server. This value is usually the default LOCAL_HOST value, or port 389. Valid values are in the range 0–65,536. The default setting is the LDAP_PORT value.

LDAPBINDMETH=
: Controls how SPD Server clients are authenticated by the LDAP server. If this parameter is found in the SPD Server parameter file, LDAPBINDMETH= is a character string whose value must be LDAP_AUTH_SASL. The default setting is null.

LDAPBINDDN=
: Specify the distinguished name (DN) of the LDAP database entry, or the location in the LDAP Server database where the client information is stored. LDAPBINDDN is an LDAP term. LDAPBINDDN is a combination of the user ID and the network domain in which the user operates. The form of this string is `ID= , rdn1=RDN1, rdn2=RDN2, ..`, where ID is the identifier for the relative distinguished name (RDN) of a user ID that exists in the LDAP server database. The default value of the DN is `uid= , dc=DOM1, dc=DOM2, dc=DOM3`. The default value of the LDAPBINDDN parameter is null.

If no distinguished name is specified in the SPD Server server parameter file, SPD Server uses the LDAP Server host's domain name to generate values for DOM1, DOM2, and DOM3. The SPD Server user ID becomes the value for the user ID. The resulting value becomes the default user location for LDAP database members.

For example, suppose the LDAP host machine is sunhost.unx.sun.com, and the user ID is sunjws. The resulting default DN is `uid=sunjws, dc=unx, dc=sun, dc=com`. The distinguished name is used to locate the user sunjws. Then the sunjws user password is compared to the password that is stored in the LDAP database. If SPD Server users are located in a specific location in your LDAP database, be sure to specify that location using LDAPBINDDN.

See the LDAP Server administrator for your site if you need more information about LDAP parameters for your server parameter file. To use the default value for any LDAP parameter, omit the parameter specification from the server parameter file. Undeclared parameters automatically assume default values.

Note: Entering the LDAP_HOST value for LDAPSERVER can cause SPD Server to fail during start-up.

SAS Metadata Server Authentication Options

Specify one or more of the following options in the spdsserv.parm parameter file to configure SAS Metadata Server authentication:

METAAUTHENTIC
: specifies that SPD Server should use the SAS Metadata Server to perform user authentication. The back-end authentication provider is whatever the SAS Metadata Server was configured with. NOMETAAUTHENTIC is the default setting.

Note: Use METAAUTHENTIC or the other META* options. When any other META* options are specified, use of METAUTHENTIC is assumed.

METASERVER=
: Specify the name of the host that the SAS Metadata Server is running on. If SAS Metadata Server authentication is configured but METASERVER= is not specified, then SPD Server assigns a default value of METASERVER=`localhost`.

METAPORT=
: Specify the port number of the SAS Metadata Server. If SAS Metadata Server authentication is configured but no METAPORT= value is specified, then SPD Server assigns a default value of METAPORT=8561.

METADOMAIN=
: Specify the metadata authentication domain where the back-end authenticator resides. If no domain is specified, none is configured. Consult the SAS Metadata Server administrator for the name of the appropriate metadata authentication domain.

Configuring LDAP Authentication through SAS Metadata Server

If you want to use LDAP as the default authenticator for SAS Metadata Server, SPD Server can send user IDs through SAS Metadata Server to LDAP for validation. In this scenario, the SPD Server user IDs must exist in LDAP, and the METAAUTHENTIC option must be specified in the spdsserv.parm parameter file. (You can also specify values for the METASERVER=, METAPORT=, and METADOMAIN= server parameter options as needed.)

The SPD Server host passes the SPD Server user ID to SAS Metadata Server for authentication. SAS Metadata Server routes the authentication requests to LDAP. LDAP then performs the authentication and passes the results via SAS Metadata Server back to the SPD Server host.

Both LDAP and the password database require user IDs and passwords to operate. Some administrative planning is required: User IDs for LDAP and the password database accounts must be exact matches. However, the passwords for the LDAP and password database user IDs do not have to match. Only the LDAP password is used to perform authentication. The password database is not part of authentication transactions. The password database password is used only when SPD Server administrators add a new user to the password database.

For example, in the following LIBNAME statement, the password **`ripsnert9`** for the user **`coretest`** must exist in the authentication provider configured in SAS Metadata Server, but not in the SPD Server password database.

```
libname region_data sasspds "spds_domain"
  host="s658d01.unx.sas.com"
  service="14567"
  user="coretest"
  password="ripsnert9";
```

Creating SPD Server User Accounts

SPD Server user accounts are created by using the psmgr utility. Psmgr is a command-line utility that you can use to modify the password database.For information about this utility, see Chapter 25, “Password Database Utility,” on page 213.

SAS Management Console provides an alternate way of modifying the password database. The SPD Server Password Manager in SAS Management Console provides a graphical user interface for the same functionality provided by the psmgr utility. For more information, seeChapter 12, “Configuring and Administering SPD Server Using the SAS Management Console ,” on page 119.

Anonymous User ID

SPD Server has a general ID that is called Anonymous. Any person can connect to the server by using the eight-character user ID **`anonymou`** and no password. (There is an eight-character limit for user IDs.) All resources that are created by the Anonymous user are accessible to any other SPD Server user.

If you want to prevent Anonymous user access at your site, use the psmgr utility to add a user named Anonymou to the password database and keep the password secret.

Chapter 8

Configuring Server Domains

Overview **60**

Syntax for the libnames.parm Parameter File **60**

Domain Path Options **61**

Path Substitution **61**

Other Domain Options **62**

Verification of the libnames.parm Parameter File **62**

Dynamic Locking **63**
Overview of SPD Server Locking 63
Benefits of Using Dynamic Locking 63
How Dynamic Locking Works 63
Comparison of Dynamic Locking and Record-Level Locking 64

Organizing Domains for Scalability **64**
Overview of Organizing Domains 64
Data Tablespace 65
Index Tablespace 65
Metadata Tablespace 65
SPD Server Workspaces 66

Domains and Data Spaces **66**
Overview of Domains and Data Spaces 66
Permanent Tablespace 67
Semi-Permanent Tablespace 68
Temporary Tablespace 68

Example libnames.parm Parameter File Configurations **68**
Example 1: Minimum Configuration for Domain 69
Example 2: Specify Domain Paths for Data, Index, and Workspace Tables 69
Example 3: Query-Rewrite Domain Configuration 69
Example 4: Multiple Domain Types and Paths Configuration 70

Disk Storage **73**

Dictionary **73**
BACKUP= Parameter File Option 73
DATAPATH= Parameter File Option 73
DYNLOCK= Parameter File Option 74
INDEXPATH= Parameter File Option 74
LIBACLINHERIT= Parameter File Option 75
METAPATH= Parameter File Option 75
OWNER= Parameter File Option 76

PATHNAME= Parameter File Option 77
WORKPATH= Parameter File Option 77

Overview

When SPD Server starts, it reads the information stored in the libnames.parm parameter file. The libnames.parm parameter file establishes the names and file storage locations of SPD Server domains during the server session. A domain is an SPD Server resource that enables you to access and manipulate tables, views, and so on. Administrators use the libnames.parm parameter file as a central tool to control domain resources and user access to the domains.

Syntax for the libnames.parm Parameter File

To define a domain in the libnames.parm parameter file, you must specify a domain name and the path that points to the directory in which data files are stored. The domain specification syntax also supports domain path options and other domain options. Here is an example:

```
libname=domain-name
   pathname=primary-path
   <domain-path-options>
   <other-domain-options>;
```

At a minimum, you must specify a domain name and a PATHNAME= to create a domain. The domain name can be up to eight characters. The first character must be a letter or an underscore. Subsequent characters can be letters, numbers, or underscores. Blanks and special characters (except the underscore) are not allowed. Each domain name must be unique.

The PATHNAME= specification specifies the storage path for SPD Server component files that the domain creates. Different domains should never share the same path. As a best practice, do not use directory names with underscores in the PATHNAME= specification.

Here are some simple examples of defining domains:

```
libname=spds123 pathname=c:\data\spds123;

libname=123spds pathname=c:\data\123spds;
```

Once domains are defined, users can connect to the domains by submitting the following code to SPD Server from the SAS client. The code connects to the first domain in the previous examples:

```
libname mylib sasspds 'spds123' server=d8488.5400 user='myid' password='secret';
```

Administrators can use the options described in “Domain Path Options” on page 61 to enhance computational performance by specifying separate paths for SPD Server component files and work files. Administrators can use other domain options to control access to a domain. For more information, see “Other Domain Options” on page 62.

Domain Path Options

SPD Server uses four types of component files to store a server table. In addition, it supports partitioning of the component files. By default, all the files are stored in PATHNAME=, which is considered the primary path. You can specify dedicated paths for component files by specifying the following options in the domain specification:

DATAPATH=
: specifies a list of paths that contain data files that are associated with the domain.

INDEXPATH=
: specifies a list of paths that contain index files that are associated with the domain.

METAPATH=
: specifies a list of paths that are allocated to contain overflow metadata if the designated metadata space that is allocated in the PATHNAME= option statement becomes full.

WORKPATH=
: specifies a list of paths that contain temporary work tables and intermediate files that are associated with the domain.

  Note: Use this option with caution. When set, it overrides the WORKPATH= setting in the spdsserv.parm parameter file and does not apply only to the domain.

ROPTIONS=
: contains the optional path locations within the libnames.parm parameter file.

Use the following syntax to specify optional paths in the libnames.parm parameter file:

```
libname=domain-name
   pathname=primary-path
   roptions="datapath=('data-path_1' 'data-path_2' ... 'data-path_n')
   indexpath=('index-path_1' 'index-path_2' ... 'data-path_n')
   workpath=('work-path_1' 'work-path_2' ... 'work-path_n')
   metapath=('meta-path_1' 'meta-path_2' ... 'meta-path_n')";
```

Path Substitution

You can use the keyword @HOSTNAME@ in your primary path or ROPTIONS= paths to substitute the short host name in the path. Path substitution is useful when SPD Server is installed on a shared file system that is accessible by multiple hosts, in which each host is running its own image of SPD Server. Path substitution enables each host to share the same libnames.parm parameter file, but substitutes the unique pathnames that each host requires for data that is private to that host. Use the following syntax:

```
libname=hostspec pathname=/data/@HOSTNAME@;
```

Other Domain Options

When you define a domain, you can use the following attributes to control the accessibility of resources in the domain:

OWNER=
: specifies an owner for the domain. The owner is the only user who can create and use resources in the domain. The owner can define ACL security to grant other users access to the domain. A domain that does not have an owner can be accessed by any server user. For more information, see "OWNER= Parameter File Option" on page 76. Also see Chapter 14, "ACL Security," on page 141.

LIBACLINHERIT=
: (Optional) used in conjunction with OWNER=, extends ACL security that is defined to control access to the domain to the owner's resources. For more information, see "LIBACLINHERIT= Parameter File Option" on page 75.

DYNLOCK=
: specifies whether dynamic locking is enabled. For more information, see "DYNLOCK= Parameter File Option" on page 74. Also see "Dynamic Locking" on page 63.

BACKUP=
: controls whether the objects in the domain can be backed up or restored with SPD Server utilities. For more information, see "BACKUP= Parameter File Option" on page 73. Also see Chapter 27, "Backing Up and Restoring SPD Server Data," on page 269.

Use the following syntax to specify other domain options in the libnames.parm parameter file:

```
libname=domain-name
   pathname=primary-path
   <domain-path-options>
   <owner=userID>
   <libaclinherit=yes|no>
   <dynlock=yes|no>
   <backup=yes|no>;
```

Verification of the libnames.parm Parameter File

The libnames.parm parameter file is verified on system start-up and when the system is refreshed. The validation includes syntax checking and path validation. Syntax error messages are displayed in the SPD Server log.

Path validation verifies that the specified path exists. Path validation errors are reported in the SPD Server log.

If errors are detected in the libnames.parm parameter file during SPD Server start-up, the session is not launched. If errors are detected in the libnames.parm parameter file during a refresh, the refresh is canceled and the previous definitions are used.

Dynamic Locking

Overview of SPD Server Locking

SPD Server uses member-level locking to protect data integrity. Member-level locking obtains an exclusive lock on a table on behalf of the user who is updating the table. When a member-level lock is held, only the owner of the lock can access the table. Attempts by others to open the table for Write access fail with a member lock failure error. The server supports the LOCKING= SPD Server LIBNAME statement option to enable users to override the default locking behavior for a domain. The LOCKING= LIBNAME statement option invokes record-level locking. For more information about SPD Server record-level locking, see "LOCKING= LIBNAME Statement Option" in *SAS Scalable Performance Data Server: User's Guide*.

Dynamic locking is an alternative to both member-level locking and record-level locking. Dynamic locking is an SPD Server feature that gives multiple users concurrent Read and Write access to tables. It provides concurrent access by queueing requests. That is, a Write request waits on getting a lock on the table. The lock will be granted when there are no other Read or Write locks on the table. Instead of getting a member lock failure when a table is already locked, any subsequent Write requests wait until any existing locks are released.

Dynamic locking is different from SPD Server record-level locking. In record-level locking, all clients share the same record-level locking proxy process. Clients that use dynamic locking connect to a separate SPD user proxy process for each connection in the domain.

Dynamic locking is enabled or disabled at the domain level in the libnames.parm parameter file. All tables that are stored within the domain are subject to the setting of the dynamic locking feature.

Benefits of Using Dynamic Locking

SPD Server uses the dynamic locking feature to alleviate some of the problems and limitations that occur with record-level locking. The dynamic locking method of using separate proxy processes instead of a single, record-level proxy distributes resource allocations. This process decreases the probability of a single-proxy process reaching resource limits. Dynamic locking also removes a single, record-level locking point of failure for the record-level proxy. If there is a failure in an SPD Server user proxy when dynamic locking is being used, only the client that is connected to that proxy is affected. If there is a failure in an SPD Server record-level proxy, then all client connections are affected.

How Dynamic Locking Works

When SPD Server proxy processes receive concurrent Update, Append, Insert, and Delete commands, the commands are sequentially queued and then executed in order of arrival. Only one Update operation is performed on a table at any one time. Read requests can be executed at any point during an Update operation. Read requests get the most recent information that is available in the table, based on the last physical update to disk.

Dynamic locking is not a replacement for using record-level locking when the user requires SAS record-level integrity across multiple clients. Reading a record using dynamic locking does not guarantee that the record will not change before a subsequent read or update is executed. If a client needs a true record-level lock, then the record-level locking protocol should be used.

Note: A domain that has dynamic locking enabled cannot also use record-level locking.

Comparison of Dynamic Locking and Record-Level Locking

The following lists summarize the highlights of the two locking features for easy comparison:

Dynamic locking:

- Locks the table being updated.
- Queues subsequent write requests, instead of failing the requests.
- Each server connection gets its own user proxy.
- A failure in one user proxy does not affect requests in the other user proxies.
- It's possible that two users proxies will update the same table record at the same time, and overwrite each other's updates.
- Metadata updates will get an error because they require an exclusive table lock.
- The feature must be enabled and disabled in the libnames.parm parameter file.

Record-level locking:

- Locks the record being updated while concurrently allowing access to other parts of a table.
- Requests to update a locked record fail.
- All user access to all domains that use record-level locking go through one user proxy: the locking proxy.
- Integrity of record updates is guaranteed.
- If there is a failure in the user proxy, all users are affected.
- Metadata updates will get an error because they require an exclusive table lock.
- The feature can be enabled or disabled in the SASSPDS LIBNAME statement.

Organizing Domains for Scalability

Overview of Organizing Domains

SPD Server performance is based on scalable I/O. To exploit scalable I/O, you can use the libnames.parm parameter file to optimize how SPD Server stores files. "Domain Path Options" on page 61 describes how to specify named paths for the data components of server tables (data tables, index tables, and metadata tables), and how to specify paths for temporary intermediate calculation tables. Domains can specify the system paths that are associated with each tablespace component. However, you must allocate the correct amount of disk space and I/O redundancy to the various paths.

This section provides functional information about the tablespaces that are defined by the DATAPATH=, INDEXPATH=, WORKPATH=, and METAPATH= options. Use this information to determine the best sizing, I/O, and redundancy requirements to optimize performance and scalability for domain paths.

Data Tablespace

When you define a domain, data tables are stored in the space that is defined in the PATHNAME= specification, unless you specify the DATAPATH= option. The PATHNAME= space contains metadata tables for a domain, but it can also contain data tables. As the size and complexity of a domain increase, so do the benefits of organizing data tables into their own DATAPATH= space.

Organizing your data tablespace significantly impacts I/O scalability. The disk space that is allocated to data tables stores permanent warehouse tables that users will access. This disk space should support scalable I/O because it facilitates both parallel processing and real-time multi-user access to the data. In a large warehouse, this disk space probably has the greatest proportion of Read and Write I/O.

Typically, you load and refresh tables in the data tablespace using batch processes during evenings or off-peak hours. You can restrict access to data tablespace to Read-Only access for all users except administrators who perform the load and refresh processes.

To ensure reliability, organize data tablespace into RAID 1+0 or RAID-5 disk configurations. For large warehouses, consider a RAID-5 configuration with a second storage array to mirror the data.

Index Tablespace

When you define a domain, index tables are stored in the space that is defined in the PATHNAME= specification, unless you specify the INDEXPATH= option. The PATHNAME= space contains metadata tables for a domain, but it can also contain index tables. As the size and complexity of a domain increase, so do the benefits of organizing index tables into their own INDEXPATH= space.

Index space typically does not require the high-level scalability that data space, temporary tablespace, or workspace needs for I/O performance. When a process is using an index, the Read access pattern is different from a parallel I/O Read access pattern of data, or multiple user Read access patterns against data.

Typically, you configure index space as a large striped file system across a large number of disks and I/O channels. A typical configuration such as RAID 1+0 or RAID 5 supports some redundancy to ensure the availability of index space.

Metadata Tablespace

When you define a domain, metadata tables are stored in the space that is specified in the PATHNAME= parameter. If the space configured in PATHNAME= is full, SPD Server stores overflow metadata for existing tables in the space that is specified in the METAPATH= option, if it is specified. The PATHNAME= and METAPATH= spaces contain metadata tables for a domain.

Compared to the other space categories, metadata space is relatively small and usually does not require scalability. If compressed data in a given warehouse uses 10 terabytes of disk space, then there are approximately 10 gigabytes of metadata. When you are setting up metadata space, plan to allot 20 gigabytes of metadata space for every 10 terabytes of physical data disk space. When new data paths are added to expand a server, you should add more metadata space within the primary path of the server. Even though the metadata requires only a small amount of space, the disk space must be expandable and mirrored. You also need to back up the metadata.

The metadata for a table becomes larger when rows in the table are marked as deleted. Bitmaps are stored in the metadata that is used to filter the deleted rows. The space required depends on the number of rows that were deleted and on their distribution within the table.

SPD Server Workspaces

You reserve a space for intermediate calculations and temporary files in statements that are in the body of the spdsserv.parm parameter file. The workspace that you configure in spdsserv.parm is shared by all SPD Server users.

Some users have data needs that might be constrained by using the common intermediate calculation and file space that is reserved for all users. Use the libnames.parm parameter file to create and reserve a workspace that is specifically associated with a single domain and its approved users. Doing so can improve both security and performance. As the size and complexity of a domain increase, so do the benefits of organizing temporary and intermediate tables into their own workspace, defined by the WORKPATH= option.

A workspace is an area on disk that SPD Server uses to store required files when the available CPU memory cannot contain the entire set of calculations. When sufficient memory is not available, some utility files are written to disk. Workspaces are important to scalability. Tasks such as large sorts, index creation, parallel group-by operations, and SQL joins can require dedicated workspace to store temporary utility files.

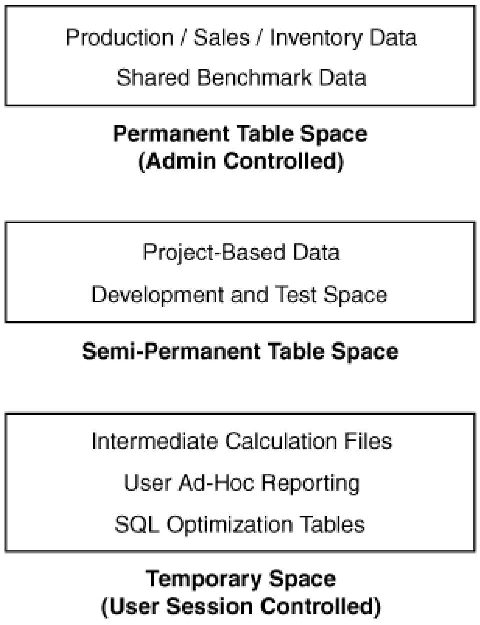
You typically configure a workspace as part of a large striped file system that spans as many disks and I/O channels as possible. Workspace I/O can critically impact the performance behavior of an SPD Server host.

Workspace on disk is typically a RAID 0 configuration or a hardware-redundant RAID design. RAID 0 configurations are risky because if the RAID 0 disk goes down, the system is also affected; any process that was running at the time of failure is also likely to be affected.

Domains and Data Spaces

Overview of Domains and Data Spaces

You can configure SPD Server to meet different data requirements. If you need different types of domain space, you can define domains in the libnames.parm parameter file to configure spaces that balance processing speed, space, and growth needs with data security requirements. Typically, SPD Server users use most or all of the types of tablespaces. The type of queries and reports that the user makes can indicate the types of data space that the user needs. The following figure shows the three basic types of domain space.


Production / Sales / Inventory Data
Shared Benchmark Data
Permanent Table Space
(Admin Controlled)
Project-Based Data
Development and Test Space
Semi-Permanent Table Space
Intermediate Calculation Files
User Ad-Hoc Reporting
SQL Optimization Tables
Temporary Space
(User Session Controlled)

Permanent Tablespace

In SPD Server, large production, inventory, and sales data storage areas work best using permanent tablespace. A rolling 5-year sales data table organized by division and company is an example of an SPD Server structure that is most suitable for permanently allocated space on the enterprise computers. If you have this type of table, large quantities of production, inventory, or sales data can be updated on a day-to-day, or even shift-to-shift, basis. These data repositories require permanent, secure processing space that can be accessed only by a select group of users. When you allocate permanent space for the data, you ensure that disk space that is required for combining and manipulating large amounts of data from multiple large warehouse tables is always available.

For example, an organization might call such a tightly controlled, permanently defined area the production data space. Data analysts in organizations typically manipulate production-type data to produce smaller, more focused reports. Analyst reports often benchmark specific areas of performance or interest. Regular analyst reports are frequently distributed across the organization. The distributed analyst reports (although not as critical as the production, inventory, or sales data) should also use permanently defined data spaces that are separate from the permanent tablespaces devoted to production reporting. In this situation, permanent tablespace should be accessible to a specific group (such as analysts) of regular SPD Server users.

You can use the libnames.parm parameter file to configure paths that map to an area of reserved disk space on a host computer. This disk space is a safe place to store permanent tables, with limited user access. To reserve permanent tablespace, use the DATAPATH=, INDEXPATH=, and OWNER= options to specify unique, appropriately sized disk areas for data tables and index tables. The OWNER= options configures ownership and access. You must ensure that the paths specified for a domain have access to sufficient disk space.

You can grant user access to permanent tablespaces by using individual user account access privileges, or by establishing an ACL group of approved users through the owner of the domain. Permanent tablespace is created by default.

Semi-Permanent Tablespace

Organizations often have short-term data mining projects that rely on production, inventory, or sales data. Sometimes the organization changes how the data is processed, or augments the production, inventory, or sales data with additional information. These projects should be conducted in a test data space, isolated from the permanent space that is dedicated to critical production, inventory, or sales data. This design lets development trials be conducted without the risk of corrupting mission-critical data.

For example, the test data space that is used for a month-long development project could be considered a semi-permanent space. You need to grant access to an area where data can safely exist, isolated from production, sales, or inventory data, for a period of time that is longer than a single SPD Server user session. The test environment should persist long enough for works-in-progress to mature to production-quality. After the project is completed, the data, metadata, and work tables that are associated with the development phase should be cleaned up and deleted from the test environment.

SPD Server administrators and users can configure semi-permanent tablespace. However, administrators should allocate semi-permanent spaces using the libnames.parm parameter file.

Temporary Tablespace

Managers in an organization often ask analysts to query data warehouses for various types of information. Such ad hoc requests might be as important as standard reports, but ad hoc reporting has different data space needs. Ad hoc reports usually have a lower frequency of repetition and a broader query scope than standard daily reporting does. Ad hoc reports are usually suitable for temporary tablespace. The life span of temporary tablespaces begin and end with the user's SPD Server sessions.

You can use temporary tablespace for more than ad hoc user reporting. Even data warehouse queries and reports that use permanent tablespace use intermediate tables and calculation metadata to process queries. For example, the SPD Server SQL optimization process requires significant temporary tablespace while it heuristically finds the most efficient SQL strategy to resolve a query. Intermediate server tables and calculation metadata are usually deleted when the job terminates.

Any report might require temporary tablespace for intermediate calculation tables. SPD Server users can configure temporary tablespace by setting the TEMP=YES LIBNAME option in the LIBNAME statement that connects their SAS session to the server. All tables in temporary tablespace are lost at the end of the SPD Server user session, when temporary tablespace is automatically deleted.

Example libnames.parm Parameter File Configurations

The following examples show the range of domains that you can define in the libnames.parm parameter file. The examples begin with the simplest form and increase in complexity.

Example 1: Minimum Configuration for Domain

This example contains the syntax that is required for the simplest form of a domain configuration. This statement defines the domain SKULIST. All tables that are associated with the SKULIST domain (table data, metadata, index data, and intermediate data) reside in the single directory that is referenced in the path specification **`c:\data \skulist`**.

```
libname=skulist pathname=c:\data\skulist;
```

Example 2: Specify Domain Paths for Data, Index, and Workspace Tables

This example contains the syntax that is required to define a domain with separate paths allocated for the data tables, index tables, and intermediate data files. The domain metadata continues to be stored in the location specified by the PATHNAME= specification.

```
libname=skulist pathname=/metadata/skulist
   roptions="
    datapath=('/data01/skulist'
              '/data02/skulist'
              '/data03/skulist'
              '/data04/skulist'
              '/data05/skulist'
              '/data06/skulist')
    indexpath=('/idx01/skulist'
               '/idx02/skulist'
               '/idx03/skulist'
               '/idx04/skulist')
    workpath=('/work01/skulist'
              '/work02/skulist'
              '/work03/skulist'
              '/work04/skulist')";
```

This example uses the options DATAPATH=, INDEXPATH=, and WORKPATH=. You can achieve optimal performance with this configuration when each domain path resides on a separate disk or on network components that can take advantage of parallelism.

The INDEXPATH= option takes advantage of multiple file systems. Starting in SPD Server 5.1, index components can take advantage of the RANDOMPLACEDPF feature. You can configure smaller disk partitions for index space with the RANDOMPLACEDPF feature, which benefits backup and recovery operations.

The WORKPATH= specified for the SKULIST domain enables users to override a default workpath that is specified in the spdsserv.parm parameter file.

Example 3: Query-Rewrite Domain Configuration

This example shows how to use temporary tables to configure a domain for optimal performance when you are using the SPD Server SQL query rewrite facility.

The SQL query rewrite facility finds the most processor-efficient method to evaluate SQL statements. The SQL query rewrite facility uses numerous temporary tables that are distributed across a parallelized environment to rapidly evaluate and process the SQL statements.

At the end of the SPD Server session, temporary tables are automatically deleted. Some SPD Server users might use the QRW domain for its temporary tablespace, even if they are not submitting code for an SQL query rewrite job.

This example defines a query rewrite domain named QRW that uses distributed temporary tables. To use SPD Server QRW:

- Define a domain for the query rewrite operations in the libnames.parm parameter file. This example names the query rewrite domain QRW.
- In the spdsserv.parm parameter file, include the TMPDOMAIN=option that references the QRW domain that is defined in the libnames.parm parameter file.

The following libnames.parm parameter file defines a domain for the query rewrite tables:

```
libname=qrw pathname=/metadata/qrw
   roptions="
    datapath=('/data01/qrw'
              '/data02/qrw'
              '/data03/qrw'
              '/data04/qrw'
              '/data05/qrw'
              '/data06/qrw'
              '/data07/qrw'
              '/data08/qrw'
              '/data09/qrw')
    indexpath=('/idx01/qrw'
               '/idx02/qrw'
               '/idx03/qrw'
               '/idx04/qrw'
               '/idx05/qrw')";
```

In the spdsserv.parm parameter file, the TMPDOMAIN=QRW option references the domain for query rewrite tables.

```
tmpdomain=qrw;
```

Example 4: Multiple Domain Types and Paths Configuration

This example uses a combination of libnames.parm parameter file settings, spdsserv.parm parameter file settings, and user-issued SAS code that is submitted to SPD Server to define multiple domains that store the following items:

- permanent production tables
- permanent to semi-permanent user tables
- temporary tables for intermediate calculations

In this environment, users can access information from permanent production-type tables, manipulate the information, and save and delete the results in a semi-permanent user space. At the same time, they can use temporary tables with sufficient disk space to perform large or optimized intermediate table calculations. The code specifies data and index paths to take advantage of RAID-configured disk arrays.

This libnames.parm parameter file defines the domain PROD, which contains permanent production and historical data tables.

```
libname=prod pathname=/metadata/prod
  roptions="
    datapath=('/data01/prod'
              '/data02/prod'
              '/data03/prod'
              '/data04/prod'
              '/data05/prod'
              '/data06/prod'
              '/data07/prod'
              '/data08/prod'
              '/data09/prod')
    indexpath=('/idx01/prod'
               '/idx02/prod'
               '/idx03/prod'
               '/idx04/prod'
               '/idx05/prod')";
```

Additional libnames.parm parameter file code defines the domain USERTBLS, which contains semi-permanent tables for user projects. SPD Server users can save and delete content in USERTBLS.

```
libname=usertbls pathname=/metadata/usertbls
   roptions="
    datapath=('/data01/usertbls'
              '/data02/usertbls'
              '/data03/usertbls'
              '/data04/usertbls'
              '/data05/usertbls'
              '/data06/usertbls'
              '/data07/usertbls'
              '/data08/usertbls'
              '/data09/usertbls')
    indexpath=('/idx01/usertbls'
               '/idx02/usertbls'
               '/idx03/usertbls'
               '/idx04/usertbls'
               '/idx05/usertbls')";
```

Finally, more libnames.parm parameter file code defines the domain named SPDTEMP, which contains temporary tablespace that is automatically deleted at the end of the session.

```
libname=spdtemp pathname=/metadata/spdtemp
   roptions="
    datapath=('/data01/spdtemp'
              '/data02/spdtemp'
              '/data03/spdtemp'
              '/data04/spdtemp'
              '/data05/spdtemp'
              '/data06/spdtemp')
    indexpath=('/idx01/spdtemp'
               '/idx02/spdtemp'
               '/idx03/spdtemp'
               '/idx04/spdtemp')";
```

The spdsserv.parm parameter file code uses the TMPDOMAIN=SPDTEMP option to reference the domain that was defined for temporary tables. This code also uses the WORKPATH= option to identify an array of RAID-enabled disk paths for temporary work tables and temporary intermediate files.

```
SORTSIZE=128M;
INDEX_SORTSIZE=128M;
GRPBYROWCACHE=128M;
BINBUFSIZE=32K;
INDEX_MAXMEMORY=8M;
NOCOREFILE;
SEQIOBUFMIN=64K;
RANIOBUFMIN=4K;
MAXWHTHREADS=8;
WHERECOSTING;
RANDOMPLACEDPF;
MINPARTSIZE=128M;
TMPDOMAIN=SPDTEMP;
WORKPATH="('/work1/spdswork'
           '/work2/spdswork'
           '/work3/spdswork'
           '/work4/spdswork'
           '/work5/spdswork')";
```

The following SAS code is submitted to SPD Server by the user. The code connects to the PROD, USERTBLS, and SPDTEMP domains, and configures SPDTEMP as a temporary domain space. Tables in the SPDTEMP domain are automatically deleted at the end of the session.

```
libname prod sasspds "prod"
   server=hostname.hostport
   user="user-id"
   password="password"
   ip=yes;

libname usertbls sasspds "usertbls"
   server=hostname.hostport
   user="user-id"
   password="password"
   ip=yes;

libname spdtemp sasspds "spdtemp"
   server=hostname.hostport
   user="user-id"
   password="password"
   ip=yes
   temp=yes;
```

Disk Storage

How you configure SPD Server disk storage is important, whether you have many SPD Server users or a just a few large-scale users. To effectively configure SPD Server disk storage for your installation, you need to understand the four types of component files that SPD Server creates, the relative sizes of these files, and when these files are created. For more information, see Chapter 21, “Optimizing Disk Storage,” on page 195.

Dictionary

BACKUP= Parameter File Option

Controls whether the objects in the domain can be backed up or restored using the SPD Server backup and restore utilities.

Valid in:	libnames.parm parameter file
Default:	NO
Restriction:	Users cannot create tables with the DS2 and FedSQL languages in domains that have BACKUP=YES.

Syntax

BACKUP= NO | YES

Arguments

NO
disables backup and restore utilities. This is the default setting.

YES
enables use of SPD Server backup and restore utilities. For more information, see Chapter 27, “Backing Up and Restoring SPD Server Data,” on page 269.

Details

Users of the DS2 and FedSQL languages will get the following error if they try to create a table in a BACKUP=YES domain:

```
ERROR: Creation of aligned tables in BACKUP=YES domains not supported
```

DATAPATH= Parameter File Option

Specifies a list of paths that contain SPD Server data tables that are associated with the domain.

Valid in:	libnames.parm parameter file
Default:	The primary path specified in PATHNAME=

Requirement: Use ROPTIONS= to specify DATAPATH=

Syntax

DATAPATH=*path(s)*

Argument

path(s)
a complete pathname in single or double quotation marks within parentheses. Separate multiple paths with spaces. Here is an example:

```
datapath=('/data1/spds123'
           '/data2/spds123'
           '/data3/spds123'
           '/data4/spds123')
```

DYNLOCK= Parameter File Option

Specifies whether dynamic locking is enabled.

Valid in: libnames.parm parameter file

Default: NO

Syntax

DYNLOCK= NO | YES

Arguments

NO
disables dynamic locking. This is the default setting.

YES
enables dynamic locking. For more information about dynamic locking, see "Dynamic Locking" on page 63..

Details

If you set DYNLOCK=YES, you cannot use the LOCKING= LIBNAME statement option. Use of dynamic record locking precludes use of record-level locking.

INDEXPATH= Parameter File Option

Specifies a list of paths that contain SPD Server index tables that are associated with the domain.

Valid in: libnames.parm parameter file

Requirement: Use ROPTIONS= to specify INDEXPATH=

Syntax

INDEXPATH=*path(s)*

Argument

path(s)
: a complete pathname in single or double quotation marks within parentheses. Separate multiple paths with spaces. Here is an example:

```
indexpath=('/idx1/spds123'
           '/idx2/spds123'
           '/idx3/spds123'
           '/idx4/spds123')
```

LIBACLINHERIT= Parameter File Option

Extends the user and group permissions defined in the domain ACL to resources created by the domain owner.

Valid in:	libnames.parm parameter file
Default:	NO
Requirement:	OWNER= must be specified.
See:	Chapter 14, "ACL Security," on page 141

Syntax

LIBACLINHERIT= NO | YES

Argument

NO
: Specifies to use normal ACL rules for resource access.

YES
: Extends the permissions in the domain ACL to resources created by the domain owner. This allows users to access the owner's resources without the owner having to create separate user or group ACLs for those resources. For more information, see "Domain ACLs for LIBACLINHERIT=YES Domains" on page 146.

See Also

"OWNER= Parameter File Option" on page 76

METAPATH= Parameter File Option

Specifies a list of paths that are allocated to contain overflow SPD Server metadata if the designated metadata space that is allocated in the PATHNAME= option becomes full.

Valid in:	libnames.parm parameter file
Requirement:	Use ROPTIONS= to specify METAPATH=

Syntax

METAPATH=*path(s)*

Argument

path(s)

a complete pathname in single or double quotation marks within parentheses. Separate multiple paths with spaces. The additional metapaths provide a buffer space that can be used for Update and Append operations to existing server tables. Here is an example:

```
metapath=('/meta1/spdsmgr/meta'
          '/meta2/spdsmgr/meta')
```

Details

When the primary metadata space that is defined by the PATHNAME= option is full, new tables cannot be added to the domain. Put the primary path on a file system that is expandable and mirrored. As a conservative estimate for space, plan for 20 GB of metadata for every terabyte of compressed physical data.

OWNER= Parameter File Option

Specifies the owner of a domain.

Valid in:	libnames.parm parameter file
Interaction:	(Optional) Used in conjunction with LIBACLINHERIT=.
See:	Chapter 14, “ACL Security,” on page 141

Syntax

OWNER=*userID*

Argument

userID

the server user ID of the owner of a domain. Here is an example:

```
owner=admin
```

Details

The user specified as OWNER= controls access to the domain. The domain owner can create a domain ACL to provide access to the domain to other SPD Server users. When a domain is specified with an OWNER=, only the owner can use the TEMP=YES LIBNAME statement option with the domain.

The domain ACL controls Read and Write access to the domain. It does not grant access to the resources in the domain. Once the domain ACL is established, normal SPD Server ACL access rules are used to grant or deny access to the resources in the domain. That is, the creator of a resource is the resource owner.

The OWNER= can use the domain ACL to grant the following access levels:

- Read access to allow a user or group to connect to the domain and read server tables to which that user or group has been given access.
- Write access to allow a user or group to create new tables in the domain.
- Control access to allow a user or group to modify the domain ACL.

The domain ACL does not grant access to resources created by the owner, unless the domain definition specifies OWNER= and LIBACLINHERIT=YES. For more information, see "LIBACLINHERIT= Parameter File Option" on page 75.

See Also

"LIBACLINHERIT= Parameter File Option" on page 75

PATHNAME= Parameter File Option

Specifies the primary path in which to store data partitions for an SPD Server table.

Valid in: libnames.parm parameter file

Syntax

PATHNAME=*primary-path(s)*

Argument

primary-path
the complete pathname to a directory. Here is an example:

```
pathname=/disk1/primarypath
```

Note: The pathname must be unique for each domain.

WORKPATH= Parameter File Option

Specifies a list of paths that contain temporary SPD Server work tables and temporary intermediate files that are associated with the declared domain.

Valid in: libnames.parm parameter file

Requirement: Use ROPTIONS= to specify WORKPATH=

Syntax

WORKPATH=*path(s)*

Argument

path(s)
a complete pathname in single or double quotation marks within parentheses. It separates multiple paths with spaces. Here is an example:

```
workpath=('/work1/spds123'
             '/work2/spds123'
             '/work3/spds123'
             '/work4/spds123')
```

Details

The WORKPATH= setting in the libnames.parm parameter file does not apply only to the domain. The WORKPATH= setting in the libnames.parm parameter file overrides the WORKPATH= value in the spdsserv.parm parameter file to become the new global setting for the server.

The WORKPATH= setting is activated when a connection is made to the domain. The last server connection that specifies a domain that has a WORKPATH= setting sets the global working path for all other connections. Consult the server log to see which WORKPATH= is in use.

Chapter 9
SQL Query Rewrite Facility

Overview of the SQL Query Rewrite Facility 79

Configuring Storage Space for the SQL Query Rewrite Facility 79

SQL Query Rewrite Facility Options 80
_QRWENABLE Option 80
_QRW Option 81

Overview of the SQL Query Rewrite Facility

The SQL query rewrite facility in SPD Server examines SQL queries in order to optimize processing performance. Some SQL queries contain SQL statements and subqueries that can be more rapidly evaluated in a separate space, as opposed to sequentially evaluating large blocks of SQL statements. When SPD Server detects and processes SQL statements or subqueries in a separate space, intermediate result tables are produced. The original SQL query is dynamically rewritten, using intermediate results tables to replace the SQL code that was evaluated separately. The result is a dynamic process that evaluates and processes SQL queries more efficiently.

Inserting the derived intermediate data into the original SQL query does not change the quantitative results. Rather, the processing that is required to calculate them is expedited. The SQL query rewrite facility does not change the content of the query's answer row set. However, the order of the rows in the query answer set might vary if you compare the optimized query answer set with the query answer set that SPD Server generates when the SQL query rewrite facility is disabled.

Configuring Storage Space for the SQL Query Rewrite Facility

The SQL query rewrite facility uses intermediate results tables to expedite original SQL queries. Administrators must provide adequate space in order for the intermediate results tables to be generated and stored. The intermediate results tables are formatted as server tables. Optional indexes are permitted.

Intermediate results tables are stored in a common SPD Server domain. One dedicated domain is sufficient to provide adequate intermediate results table storage for many concurrent SPD Server users. The SQL query rewrite facility uses the sasspds engine's TEMP=YES LIBNAME option when it accesses the SPD Server domain for intermediate result tables. The TEMP=YES option creates a processing environment in which multiple users can concurrently create tables without name or resource contention issues. The TEMP=YES option also automatically cleans up the contents of the working storage area when users close their SPD Server session in a normal fashion.

SPD Server administrators and users can specify the domain for SQL query rewrite facility intermediate results storage. Administrators can specify the domain by setting the TMPDOMAIN= parameter in the spdsserv.parm parameter file, as follows:

```
TMPDOMAIN=Domain-Name;
```

Domain-Name is the name of a domain that is defined in the libnames.parm parameter file that is associated with SPD Server.

Users can specify the location by submitting TMPDOMAIN= as a SQL pass-through RESET option. For example, if a user wanted to use the domain EMATMP for SQL Rewrite Facility intermediate results, he could submit the following statement in his SQL job stream:

```
execute(reset tmpdomain=ematmp) by sasspds;
```

The EMATMP domain takes precedence over the TMPDOMAIN= setting that was configured in the spdsserv.parm parameter file. Any domain that a user submits as an SQL Query Rewrite storage location must be defined in the libnames.parm parameter file of the SPD server that runs the pass-through SQL code.

Reassigning the location of the SQL query rewrite facility intermediate results storage does not affect the TEMP=YES environment setting that permits concurrent access to tables in the domain by multiple SPD Server users. Multiple SPD Server users can specify and share remapped TMPDOMAIN= locations without experiencing table handling or contention issues.

Note: If the SPD Server parameter TMPDOMAIN= is not configured and the SQL query rewrite is enabled, the query rewrite fails with the following error:

```
SPDS_ERROR: Error materializing RWE context.
```

SQL Query Rewrite Facility Options

The SQL query rewrite facility is enabled by default in SPD Server. When an SPD Server user submits SQL statements that contain subexpressions that SPD Server can handle more efficiently by using the SQL query rewrite facility, the software optimizes the SQL query. The following RESET options provide control over the SQL query Rewrite Facility.

_QRWENABLE Option

Use the _QRWENABLE reset option to disable the SQL query rewrite facility. You might use this option if you suspect that the SQL query rewrite facility is not enhancing the performance of the SQL query execution. The SQL query rewrite facility is enabled by default.

Examples:

Disable SQL query rewrite facility:

```
execute(reset no_qrwenable) by sasspds; /* Disable query rewrite */
execute(reset _qrwenable=0) by sasspds; /* Another way to disable */
```

Re-enable SQL query rewrite facility:

```
execute(reset _qrwenable) by sasspds; /* Re-enable query rewrite */
execute(reset _qrwenable=1) by sasspds; /* Another way to enable */
```

_QRW Option

Use the _QRW reset option to enable diagnostic debugging and tracing outputs from the SQL query rewrite facility in the log. The diagnostic debugging option is disabled by default.

Examples:

Enable diagnostic debugging function:

```
execute(reset _qrw) by sasspds; /* Enable diagnostics */
execute(reset _qrw=1) by sasspds; /* Another way to enable */
```

Disable diagnostic debugging function:

```
execute(reset no_qrw) by sasspds; /* Disable diagnostics */
execute(reset _qrw=0) by sasspds; /* Another way to disable */
```

Chapter 10
Setting Server Operational Parameters

Overview 84
Syntax for the spdsserv.parm Parameter File 84
Server Performance Levels 85
Examples of Performance Level Settings 85
Parameter File Validation 86
Dictionary 87
BINBUFSIZE= Parameter File Option 87
BYINDEX Parameter File Option 87
COREFILE Parameter File Option 88
FMTDOMAIN= Parameter File Option 88
FMTNAMENODE= Parameter File Option 89
FMTNAMEPORT= Parameter File Option 90
GRPBYROWCACHE= Parameter File Option 90
IDLE_TIMEOUT= Parameter File Option 91
INDEX_MAXMEMORY= Parameter File Option 91
INDEX_SORTSIZE= Parameter File Option 92
LDAP Parameter File Option 92
LDAPBINDDN= Parameter File Option 93
LDAPBINDMETH= Parameter File Option 93
LDAPPORT= Parameter File Option 94
LDAPSERVER= Parameter File Option 94
MAXGENNUM= Parameter File Option 94
MAXMEMORYSIZE= Parameter File Option 95
MAXSEGRATIO= Parameter File Option 95
MAXSORTTHREADS= Parameter File Option 96
MAXWHTHREADS= Parameter File Option 96
METAAUTHENTIC Parameter File Option 97
METADOMAIN= Parameter File Option 98
METAPORT= Parameter File Option 98
METASERVER= Parameter File Option 98
MINPARTSIZE= Parameter File Option 99
MINPORTNO= and MAXPORTNO= Parameter File Options 99
RANDOMPLACEDPF Parameter File Option 100
RANIOBUFMIN= Parameter File Option 100
REVNUMUPDATE Parameter File Option 101
SEQIOBUFMIN= Parameter File Option 101
SHRUSRPRXY Parameter File Option 102
SORTSIZE= Parameter File Option 102
SQLAUDLEN= Parameter File Option 103

SQLOPTS= Parameter File Option 103
SSLALLOWUNXDS Parameter File Option 103
SSLCALISTLOC= Parameter File Option 104
SSLCERTISS= Parameter File Option 105
SSLCERTLOC= Parameter File Option 105
SSLCERTSERIAL= Parameter File Option 106
SSLCERTSUBJ= Parameter File Option 106
SSLCLIENTAUTH Parameter File Option 107
SSLCRLCHECK Parameter File Option 107
SSLCRLLOC= Parameter File Option 108
SSLPKCS12LOC= Parameter File Option 108
SSLPKCS12PASS= Parameter File Option 109
SSLPVTKEYLOC= Parameter File Option 109
SSLPVTKEYPASS= Parameter File Option 110
SSLREQCERT= Parameter File Option 110
SSLSECURE= Parameter File Option 111
STARSIZE= Parameter File Option 111
TMPDOMAIN= Parameter File Option 112
WHAUDLEN= Parameter File Option 112
WHEREAUDIT Parameter File Option 113
WHERECOSTING Parameter File Option 113
WORKPATH= Parameter File Option 114

Overview

When SPD Server starts, it reads information stored in the spdsserv.parm parameter file. The spdsserv.parm parameter file contains options that control server processing behavior and use of server resources. The server looks for the spdsserv.parm parameter file in the start-up directory, unless you created the file in a different location or with another name and specified it using the -PARMFILE start-up option.

For information about spddserv.parm options for Hadoop, see *SAS Scalable Performance Data Server: Processing Data in Hadoop*.

Syntax for the spdsserv.parm Parameter File

The spdsserv.parm parameter file options that can take multiple values are expressed as Name = Value pairs:

```
Name=Value;
```

Options that are Boolean (yes or no) have this form:

```
NAME | NONAME
```

Option keywords are not case sensitive. In Name = Value pairs, the value is option dependent. Any value that is a memory size is stated in bytes. These values can support a "K" suffix to specify kilobytes, an "M" suffix to specify megabytes, or a "G" suffix to specify gigabytes.

Comments are allowed in the spdsserver.parm parameter file. For an example, see the default spdsserv.parm parameter file at **`InstallDir/samples/spdsserv.parm`**.

Server Performance Levels

You can specify different performance levels based on the performance class of the user. You can also specify a performance level based on whether the user is assigned the SPD Server locking proxy by the LOCKING=YES LIBNAME option. Every user in the SPD Server password database is assigned a performance class attribute. For more information about how the performance classes are assigned, see Chapter 25, “Password Database Utility,” on page 213. Each performance class is associated with a keyword in the spdsserv.parm parameter file that controls how user resources are allocated.

Users in the password database are designated as either low-, medium-, or high-performance users. The spdsserv.parm parameter file supports MED, HIGH, and LOCKING keywords to make the performance-level parameter assignments:

- Users with a low designation get the unlabeled values that are specified in the spdsserv.parm parameter file.
- Users with a medium designation get the unlabeled values plus any values redefined in the MED section of the spdsserv.parm parameter file.
- Users with a high designation get the unlabeled values plus any values redefined in the HIGH section of the spdsserv.parm parameter file.
- LOCKING=YES is a special user class that is activated by a LOCKING=YES LIBNAME assignment for the user. LOCKING=YES users get the unlabeled values plus any values redefined in the LOCKING section of the spdsserv.parm parameter file.

Examples of Performance Level Settings

Here is an example of how the low-, medium-, and high-performance levels are set. User Bob belongs to the low-performance class. User Tom belongs to the medium-performance class. User Mary belongs to the high-performance class. If you wanted to assign different server values to low, medium, and high users, you would define the following entities in the spdsserv.parm parameter file:

```
SORTSIZE=256M;
BINBUFSIZE=32K;
MAXWHTHREADS=4;
WORKPATH="/var/tmp";
---MED---
SORTSIZE=512M;
MAXWHTHREADS=6;
---HIGH---
SORTSIZE=1G;
MAXWHTHREADS=6;
WORKPATH="/var/hightmp";
```

Here’s what happens in the spdsserv.parm parameter file:

- Low users receive the unlabeled values for SORTSIZE=, BINBUFSIZE=, MAXWHTHREADS=, and WORKPATH=.

- Medium users get the unlabeled values, with the exceptions of the SORTSIZE= setting, which has a value of 512 MB, and the MAXWHTHREADS= setting, which has a value of 6 GB.
- High users get the unlabeled values, with the following exceptions: the SORTSIZE= setting, which has a value of 1 GB; the MAXWHTHREADS= setting, which has a value of 6 GB; and the WORKPATH= setting, which has a value of **`/var/hightmp`**.

All initial values for each performance class are inherited from the unlabeled settings. However, settings from the labeled classes are not inherited. Users who are set to HIGH do not inherit any parameter settings from the MED level. Therefore, you must specify MAXWHTHREADS=6 for the HIGH and MED levels.

The LOCKING=YES setting is a special user case. All users share the same SPD Server locking proxy. For LOCKING=YES, all users get the unlabeled values in the spdsserv.parm parameter file, regardless of their performance class. You can use the following code to override the parameter values for all users of the locking proxy.

```
SORTSIZE=256M;
BINBUFSIZE=32K;
MAXWHTHREADS=4;
WORKPATH="/var/tmp";
---MED---
SORTSIZE=512M;
MAXWHTHREADS=6;
---HIGH---
SORTSIZE=1G;
MAXWHTHREADS=8;
WORKPATH="/var/hightmp";
---LOCKING---
MAXWHTHREADS=2;
```

In this example, all LOCKING=YES users inherit the unlabeled parameter values, with the exception of the initial MAXWHTHREADS=4 setting, which changes to MAXWHTHREADS=2.

Parameter File Validation

The spdsserv.parm parameter file is validated at system start-up and when refreshed by the PROC SPDO REFRESH PARMS statement. For more information about the REFRESH PARMS statement, see Chapter 26, “SPDO Procedure,” on page 227. Any errors in the spdsserv.parm parameter file at start-up cause the SPD Server program start to fail. Errors are reported in the SPD Server log file.

Errors that occur during a spdsserv.parm parameter file refresh will cause the refresh to fail, and the server to maintain the most recent parameter file settings. PROC SPDO refresh parameter errors are reported in the SAS log and in the SPD Server log file.

Dictionary

BINBUFSIZE= Parameter File Option

Specifies the amount of memory to allocate for each bin buffer during a sort operation.

Valid in:	spdsserv.parm parameter file

Syntax

BINBUFSIZE= *n* | *n*K | *n*M | *n*G;

Required Argument

n

specifies a number that indicates the amount of memory in terms of bytes (1), kilobytes (1,024 bytes), megabytes (1,048,576 bytes), or gigabytes (1,073,741,824 bytes).

Note If you specify a value that is smaller than the record length of the spill bin, a bin buffer large enough to hold one record is created automatically.

Details

During the sorting process, SPD Server writes blocks of sorted rows (called spill bins) to disk. The final step of the process reads the contents of the spill bins to perform final row ordering. BINBUFSIZE= specifies the amount of memory that is allocated to each spill bin during final row ordering. The spill bins use the memory buffer to read rows back into memory during interleaving.

The number of spill bins depends on the size of the table, the amount of memory specified on SORTSIZE=, and the number of threads that SPD Server uses to perform sorting. For example, if you sort a 10-GB table using two concurrent threads, and SORTSIZE=2 GB, the SORTSIZE= value is divided between the two concurrent threads. Each thread reads 1 GB of row data from the table into memory. In this case, each 1-GB block of row data makes up a spill bin. The rows in the spill bin are sorted and then written to disk. After all of the rows in the table have been sorted and written to disk, the sorting process reads the spill bins back into memory for final processing. In the example, a total of 10 spill bins and 10 buffer areas interleave the sorted rows.

BYINDEX Parameter File Option

Specifies to use an index for BY sorts.

Valid in:	spdsserv.parm parameter file
Default:	NOBYINDEX
Note:	When NOBYINDEX is set, other SPD Server options that affect index use for BY sorts cannot enable index use.

Syntax

[NO]**BYINDEX**;

Optional Arguments

BYINDEX;
enables index use for BY sorts.

NOBYINDEX;
disables index use for BY sorts.

Details

The NOBYINDEX server parameter takes precedence over other SPD Server options that affect index use for BY sorts. That is, when NOBYINDEX is set, users cannot enable BY sorts to use an index. When BYINDEX is set, users can suppress index use for BY sorts with the SPDSNBIX macro variable, the SPDSNIDX macro variable, or the NOINDEX= table option.

COREFILE Parameter File Option

Specifies whether the LIBNAME proxy creates a core file when an unexpected process trap occurs.

Valid in:	spdsserv.parm parameter file
Default:	NOCOREFILE

Syntax

[NO]**COREFILE**;

Optional Arguments

COREFILE
enables core file creation.

NOCOREFILE
disables core file creation.

FMTDOMAIN= Parameter File Option

Specifies the domain where user-defined formats are stored.

Valid in:	spdsserv.parm parameter file
Default:	FORMATS

Syntax

FMTDOMAIN= *domain-name*;

Required Argument

domain-name
specifies the name of an SPD Server domain.

Requirement	A LIBNAME= specification for the domain must exist in the libnames.parm parameter file. This domain cannot be in Hadoop.

Details

The server can only have one domain where all user-defined formats are stored. The FMTDOMAIN= option defines this domain. If the FMTDOMAIN= option is not specified in the spdsserv.parm parameter file, the server looks for formats in a domain named FORMATS.

Note: The FORMATS domain is not pre-defined in the libnames.parm parameter file. You must add a LIBNAME= specification for the FORMATS domain to the libnames.parm parameter file.

Specify the FMTDOMAIN= option in the spdsserv.parm parameter file if you want to use a domain other than FORMATS as your formats domain.

FMTNAMENODE= Parameter File Option

Specifies the server on which the user-defined formats are stored.

Valid in:	spdsserv.parm parameter file
Default:	The host name of the computer on which SPD Server is running.
Interaction:	Use FMTNAMEPORT= with FMTNAMENODE=.

Syntax

FMTNAMENODE=*host-name* ;

Required Argument

host-name
specifies the name of the host where the user-defined formats are stored.

Details

Two SPD Server installations can share formats. Specify the FMTNAMENODE= option in the spdsserv.parm parameter file if you want to use a server other than the current one to store your formats.

Example

In the following example, the user-defined formats are stored on a host named d8488.

```
FMTNAMENODE=d8488;
```

FMTNAMEPORT= Parameter File Option

Specifies the port number of the server on which the user-defined formats are stored.

Valid in:	spdsserv.parm parameter file
Default:	The port number where SPD Server is running.
Interaction:	Use FMTNAMENODE= with FMTNAMEPORT=.

Syntax

FMTNAMEPORT=*port-number*;

Required Argument

port-number
specifies a port number.

Details

Two SPD Server installations can share formats. Specify the FMTNAMEPORT= option in the spdsserv.parm parameter file if you want to use a server other than the current one to store your formats.

Example

In the following example, the port number of the server on which the user-defined formats are stored is 5400.

```
FMTNAMEPORT=5400;
```

GRPBYROWCACHE= Parameter File Option

Allocates memory for caching groups during parallel group aggregations.

Valid in:	spdsserv.parm parameter file
Default:	2 MB per thread
Interaction:	Used in conjunction with MAXWHTHREADS=. See "MAXWHTHREADS= Parameter File Option" on page 96.

Syntax

GRPBYROWCACHE=*n* | *n*K | *n*M | *n*G;

Required Argument

n
specifies the amount of memory to use for the memory cache in terms of bytes (1), kilobytes (1,024 bytes), megabytes (1,048,576 bytes), or gigabytes (1,073,741,824 bytes).

Details

The parallel group SELECT statement uses multiple threads up to the MAXWHTHREADS= limit to perform parallel group aggregations. The threads equally share the memory that is specified on GRPBYROWCACHE= to cache groups in memory. Each thread receives 1/MAXWHTHREADS= of the cache.

When a thread accumulates enough distinct groups to fill its cache, the groups are moved to secondary bins. At the completion of the parallel BY-group processing, the parallel group aggregations in memory and in secondary bins are merged to produce the final sorted results. If you omit the GRPBYROWCACHE= option, the default value is a 2-MB cache per thread. You can improve aggregation performance with large numbers of groups by increasing the default value. However, you can potentially allocate more memory than is needed for caching, which diminishes the resources that are available for processing by the excess amount of assigned memory.

IDLE_TIMEOUT= Parameter File Option

Specifies the interval of idle time that lapses before the SPD Server client process automatically terminates the client connection.

Valid in:	spdsserv.parm parameter file
Default:	0 (zero)

Syntax

IDLE_TIMEOUT=*n*;

Required Argument

n
: specifies a number that indicates the number of seconds the client connection can be idle. When IDLE_TIMEOUT= is greater than 0, the option is enabled. Otherwise, SPD Server does not enable idle timeouts.

INDEX_MAXMEMORY= Parameter File Option

Specifies the amount of memory that is allocated for each open index.

Valid in:	spdsserv.parm parameter file

Syntax

INDEX_MAXMEMORY=*n* | *n*K | *n*M | *n*G;

Required Argument

n
: specifies a number that indicates the amount of memory in terms of bytes (1), kilobytes (1,024 bytes), megabytes (1,048,576 bytes), or gigabytes (1,073,741,824 bytes).

Details

This option affects Read operations on server tables.

INDEX_SORTSIZE= Parameter File Option

Specifies the amount of memory that is allocated for creating asynchronous (parallel) sort indexes or appends.

Valid in:	spdsserv.parm parameter file

Syntax

INDEX_SORTSIZE=*n* | *n*K | *n*M | *n*G;

Required Argument

n
specifies a number that indicates the amount of memory in terms of bytes (1), kilobytes (1,024 bytes), megabytes (1,048,576 bytes), or gigabytes (1,073,741,824 bytes).

Details

The specified value is divided by the number (*n*) of indexes that are to be created or appended in parallel; each index receives 1/*n*th of the allocated memory.

LDAP Parameter File Option

Specifies whether LDAP user authentication is used for SPD Server.

Valid in:	spdsserv.parm parameter file
Default:	NOLDAP
Interaction:	Use with LDAPBINDDN=, LDAPBINDMETH=, LDAPPORT=, and LDAPSERVER=.

Syntax

[NO]**LDAP**;

Optional Arguments

LDAP
enables LDAP user authentication.on SPD Server.

NOLDAP
disables LDAP user authentication for SPD Server.

Details

There are four possible configurations for direct LDAP authentication:

- **Configuration 1:** LDAP Server that is running on an SPD Server host. For this configuration, assume that all other LDAP settings use the default configuration.

Specify only the LDAP option in the spdsserv.parm parameter file. User authentication is performed by the LDAP server, which is running on the port LOCAL_HOST on the SPD Server host.

- **Configuration 2:** LDAP Server that is running on an SPD Server Host using a port other than LOCAL_HOST. For this configuration, assume that all other LDAP settings use the default configuration. Specify the LDAP and LDAPPORT= options in the spdsserv.parm parameter file.
- **Configuration 3:** LDAP Server and SPD Server host are running on different machines. Specify the LDAPSERVER= option in addition to the LDAP option in the spdsserv.parm parameter file. If the LDAP Server uses a port other than LOCAL_HOST, specify LDAPPORT= as well.
- **Configuration 4:** SPD Server user IDs and passwords are not in their default location in the LDAP database. SPD Server expects the LDAP Server to store information about SPD Server users at the location that corresponds to ou=people, dc=domain, dc=company, dc=com in its database. If your LDAP database stores the information in a different location, specify the LDAPBINDDN= option in addition to LDAP and any other LDAP configuration options that you have set.

LDAPBINDDN= Parameter File Option

Specifies the relative distinguished name (RDN) or the location in the LDAP server database where information for the connecting SPD Server client is stored.

Valid in:	spdsserv.parm parameter file
Requirement:	The LDAP parameter file option must also be set.

Syntax

LDAPBINDDN=*LDAP_server_binddn_string*;

Required Argument

LDAP_server_binddn_string
You can obtain RDN strings from the LDAP server administrator when you are configuring the SPD Server to use LDAP authentication.

LDAPBINDMETH= Parameter File Option

Specifies the LDAP authentication security level.

Valid in:	spdsserv.parm parameter file
Default:	LDAP_AUTH_SASL
Requirement:	The LDAP parameter file option must also be set.

Syntax

LDAPBINDMETH=*LDAP-bind-method-string*;

Required Argument

LDAP-bind-method-string
By default, when LDAP user authentication is configured, the Simple Authentication and Security Layer (SASL) performs LDAP_AUTH_SASL and SPD Server user authentication using Digest-MD5. This parameter enables you to specify a different LDAP bind method.

LDAPPORT= Parameter File Option

Specifies the TCP/IP port that is used to communicate with the LDAP server.

Valid in:	spdsserv.parm parameter file
Default:	LOCAL_HOST or port 389
Requirement:	The LDAP parameter file option must also be set.

Syntax

LDAPPORT=*LDAP-port-number*;

Required Argument

LDAP-port-number
specifies the TCP/IP port number or port name that is used to communicate with the LDAP server.

LDAPSERVER= Parameter File Option

Specifies the network IP address or the host machine for the LDAP server.

Valid in:	spdsserv.parm parameter file
Requirement:	The LDAP parameter file option must also be set.
Interaction:	This value is usually the same as the IP address of the SPD Server host, which is the default value.

Syntax

LDAPSERVER= *host*

Required Argument

host
specifies the host name or network IP address of the computer to use as the LDAP server.

MAXGENNUM= Parameter File Option

Specifies the maximum number of member tables that can be created in a cluster table.

Valid in:	spdsserv.parm parameter file
Default:	512

Syntax

MAXGENNUM=*n*;

Required Argument

n
specifies a number indicating the maximum number of member tables.

MAXMEMORYSIZE= Parameter File Option

Specifies the maximum amount of memory that can be used by an SPDSBASE process.

Valid in:	spdsserv.parm parameter file
Default:	Unlimited
Interaction:	All server parameter memory size options are constrained by MAXMEMORYSIZE=.

Syntax

MAXMEMORYSIZE=*n* | *n*K | *n*M | *n*G;

Required Argument

n
specifies a number that indicates the amount of memory in terms of bytes (1), kilobytes (1,024 bytes), megabytes (1,048,576 bytes), or gigabytes (1,073,741,824 bytes).

Example

The following statement specifies that a maximum of 4 GB of memory can be dedicated to an SPDSBASE process.

```
MAXMEMORYSIZE=4G;
```

MAXSEGRATIO= Parameter File Option

Controls segment candidate pre-evaluation for WHERE clause predicates with a hybrid index.

Valid in:	spdsserv.parm parameter file
Default:	75

Syntax

MAXSEGRATIO=*n*;

Required Argument

n

specifies a number indicating the percentage of possible segments from the total number of segments that can be candidate segments before pre-evaluation is not performed. If you omit this value, the default value is 75%.

Details

The WHERE clause planner pre-evaluates the segment candidates for the predicate. Only the segment candidates are searched to resolve the WHERE clause. Some queries can benefit from not performing pre-evaluation, based on the ratio of the number of segments that contain candidates to the total number of segments in the file. If the percentage of possible segments exceeds MAXSEGRATIO=, pre-evaluation is not performed, and all of the segments are searched to resolve the WHERE clause.

MAXSORTTHREADS= Parameter File Option

Specifies the maximum number of parallel threads that can be launched for a parallel sort operation.

Valid in:	spdsserv.parm parameter file
Default:	If you omit MAXSORTTHREADS=, SPD Server assigns the value of MAXWHTHREADS= to MAXSORTTHREADS=.

Syntax

MAXSORTTHREADS=*n*;

Required Argument

n

specifies a number indicating the maximum number of threads that can be launched.

Details

Threading for sorting data in parallel is a resource-intensive process that behaves differently from threaded processing. Use caution when you assign a value for MAXSORTTHREADS=. If a parallel sort uses one thread for every CPU on the server, the sort job might starve other jobs of resources. For better performance during parallel sort operations, configure values for SORTSIZE= (in MB) and MAXSORTTHREADS= (in number of threads) so that the ratio of SORTSIZE= to MAXSORTTHREADS= is between 256 MB per thread and 1 GB per thread.

Use MAXSORTTHREADS= with MAXWHTHREADS= to balance your system load. Parallel sorting can be a resource-intensive process, and parallel WHERE processing tends to be more I/O intensive. In most cases, parallel WHERE processing tasks require more threads than parallel sorting tasks.

MAXWHTHREADS= Parameter File Option

Specifies the maximum number of parallel threads that can be launched for a WHERE clause evaluation.

Valid in:	spdsserv.parm parameter file

Default:	the number of cores, divided by two
Interaction:	Use of MAXWHTHREADS= should be balanced with MAXSORTTHREADS=, if MAXSORTTHEADS= is used.

Syntax

MAXWHTHREADS=*n*;

Required Argument

n
specifies a number indicating the maximum number of threads that can be launched.

Details

An SMP (symmetric multi-processor) can have multiple CPUs that might have one or more cores to perform tasks at a given time. These tasks are either software processes or threads that the operating system schedules. MAXWHTHREADS= controls the number of threads that a single process can create when performing a task in parallel. SPD Server WHERE clause evaluation, parallel group by, parallel join, and parallel sorts will use multiple threads for performance. An appropriate thread count is based on your system architecture, the number of expected concurrent users of the system, and the types of queries they are submitting.

You might need to experiment with different MAXWHTHREADS= values until you find an appropriate setting. Setting MAXWHTHREADS= too low can minimize parallelism. Setting MAXWHTHREADS= too high can overcommit CPU resources. For information to help you determine an appropriate setting for your site, see SAS Technical Papers — SAS Scalable Performance Data Server.

Note: The value specified for MAXWHTHREADS= sets the upper limit for the value that users can specify in the SPDSTCNT macro variable and THREADNUM= table option.

METAAUTHENTIC Parameter File Option

Specifies whether the SAS Metadata Server is used to provide user authentication.

Valid in:	spdsserv.parm parameter file
Default:	NOMETAAUTHENTIC
Interaction:	If you specify any other META* option in the spdsserv.parm parameter file, then SPD Server assumes that METAAUTHENTIC is set as well.

Syntax

[NO]**METAAUTHENTIC**;

Optional Arguments

METAAUTHENTIC
enables user authentication through the SAS Metadata Server. The back-end authentication provider is whatever the SAS Metadata Server was configured for.

NOMETAAUTHENTIC
disables user authentication through the SAS Metadata Server.

METADOMAIN= Parameter File Option

Specifies the domain where the back-end authenticator resides.

Valid in:	spdsserv.parm parameter file
Default:	None
Interaction:	When this option is set, SPD Server assumes that METAAUTHENTIC is set as well.

Syntax

METADOMAIN=*domain-name*;

Required Argument

domain-name
Specifies the SPD server domain name, if needed, where the back-end authenticator used by the SAS Metadata Server resides. No domain is used if this option is not specified.

METAPORT= Parameter File Option

Specifies the port number of the SAS Metadata Server.

Valid in:	spdsserv.parm parameter file
Default:	8561
Interaction:	When this option is set, SPD Server assumes that METAAUTHENTIC is set as well.

Syntax

METAPORT=*port-number*;

Required Argument

port-number
specifies the port number of the SAS Metadata Server. If SAS Metadata Server authentication is configured, but no METAPORT= value is specified, then SPD Server assigns a default value of METAPORT=8561.

METASERVER= Parameter File Option

Specifies the name of the host that the SAS Metadata Server resides on.

Valid in:	spdsserv.parm parameter file
Default:	localhost

Interaction: When this option is set, SPD Server assumes that METAAUTHENTIC is set as well.

Syntax

METASERVER= *host-name*;

Required Argument

host-name
specifies the name of the host that the SAS Metadata Server resides on. If SAS Metadata Server authentication is configured, but METASERVER= is not specified, then SPD Server assigns a default value of METASERVER=`localhost`. This value indicates that the SAS Metadata Server is running on the same host as SPD Server.

MINPARTSIZE= Parameter File Option

Specifies the minimum partition size for SPD Server tables.

Valid in: spdsserv.parm parameter file

Default: 16M

Syntax

MINPARTSIZE=*n* | *n*K | *n*M | *n*G;

Required Argument

n
specifies a number that indicates the amount of memory in terms of bytes (1), kilobytes (1,024 bytes), megabytes (1,048,576 bytes), or gigabytes (1,073,741,824 bytes).

Details

Use MINPARTSIZE= to ensure that large server tables cannot be created with an arbitrarily small partition size. Large server tables with small partition sizes create an excessive number of physical files, which increases clutter and degrades I/O performance. The default MINPARTSIZE= value is 16 MB, unless you are creating tables in a Hadoop domain. If you are using a Hadoop domain, the minimum MINPARTSIZE= is 128 MB.

The most common values for MINPARTSIZE= are in the range 128 MB–256 MB.

MINPORTNO= and MAXPORTNO= Parameter File Options

Specify the range of port numbers that SPD Server user-proxy processes can use to communicate with the client.

Valid in: spdsserv.parm parameter file

Interaction: If you set the MINPORTNO= parameter file option, then you must set the MAXPORTNO= parameter file option.

See: Chapter 11, “Using SPD Server with an Internet Firewall,” on page 115.

Syntax

MINPORTNO= *lower-port-number*;

MAXPORTNO=*upper-port-number*;

Required Argument

port-number
specifies a TCP/IP communication port number.

Details

This option supports the use of SPD Server ports through an Internet firewall, in order to limit the range of ports that are used by the server. If you omit MINPORTNO= and MAXPORTNO=, then the SPD Server user-proxy processes use any port that is available to communicate with the client.

RANDOMPLACEDPF Parameter File Option

Controls placement of the initial data partition for all tables in a domain.

Valid in: spdsserv.parm parameter file

Default: RANDOMPLACEDPF

Syntax

[NO]**RANDOMPLACEDPF**;

Optional Arguments

NORANDOMPLACEDPF
disables random placement of the initial data partition for tables in the domain.

RANDOMPLACEDPF
invokes random placement of the initial data partition for all tables in a domain. The random placement strategy manages large tables efficiently and balances data loads without losing disk space.

RANIOBUFMIN= Parameter File Option

Specifies the minimum random I/O buffer size for proxies.

Valid in: spdsserv.parm parameter file

Syntax

RANIOBUFMIN=*n* | *n*K | *n*M | *n*G;

Required Argument

n

specifies a number that indicates the minimum I/O buffer size in terms of bytes (1), kilobytes (1,024 bytes), megabytes (1,048,576 bytes), or gigabytes (1,073,741,824 bytes). This value becomes the minimum I/O buffer size that is used by the proxy when it performs random I/O and table requests.

REVNUMUPDATE Parameter File Option

Controls whether the revision number is updated on SPD Server table component files.

Valid in:	spdsserv.parm parameter file
Default:	NOREVNUMUPDATE

Syntax

[NO]**REVNUMUPDATE**;

Optional Arguments

NOREVNUMUPDATE

suppresses revision number updates to component files when a table is modified so that the DPF and INDEX file components of a server table will not change the component filenames. There is a performance benefit in not updating revision numbers, particularly if the table contains a large number of DPF partitions, or if dynamic locking is used while performing table updates. There is also a benefit for backup and restore: the DPF partitions do not change names, and the last modified date on the file changes only if the DPF file was physically modified.

REVNUMUPDATE

adds revision numbers to component filenames when the component files are updated. Previous versions of SPD Server used REVNUMUPDATE as a default setting. If an SPD Server user has been using the revision number to track backup and restore versions, the REVNUMUPDATE setting can still be used.

SEQIOBUFMIN= Parameter File Option

Specifies the minimum sequential I/O buffer size.

Valid in:	spdsserv.parm parameter file

Syntax

SEQIOBUFMIN=*n* | *n*K | *n*M | *n*G;

Required Argument

n

specifies a number that indicates the minimum sequential I/O buffer size in terms of bytes (1), kilobytes (1,024 bytes), megabytes (1,048,576 bytes), or gigabytes

(1,073,741,824 bytes). This value becomes the minimum I/O buffer size that is used by the proxy when it performs sequential I/O and table requests.

SHRUSRPRXY Parameter File Option

Enables enhanced sharing of user proxies.

Valid in:	spdsserv.parm parameter file
Default:	SHRUSRPRXY
Interaction:	This option setting can be overridden by the SHARE= sasspds engine LIBNAME option.

Syntax

[NO]**SHRUSRPRXY** ;

Optional Arguments

NOSHRUSRPRXY
disables enhanced sharing of user proxies.

SHRUSRPRXY
enables enhanced sharing of user proxies. Enhanced SPD Server proxy sharing enables users with different SPD Server user and group credentials to use the same proxy. Enhanced user proxy sharing keeps the number of concurrent SPDSBASE process resources from growing too large. A large number of concurrent SPDSBASE processes can create system resource allocation issues in some SPD Server environments.

SORTSIZE= Parameter File Option

Specifies the amount of memory to allocate for sort operations.

Valid in:	spdsserv.parm parameter file

Syntax

SORTSIZE=*n* | *n*K | *n*M | *n*G;

Required Argument

n
specifies a number that indicates the amount of memory in terms of bytes (1), kilobytes (1,024 bytes), megabytes (1,048,576 bytes), or gigabytes (1,073,741,824 bytes).

Details

During parallel sort operations, the memory that SORTSIZE= allocates is divided evenly among the sort threads. For best results, specify SORTSIZE= values in the range 256

MB–1 GB per parallel sort thread, or between 256 * MAXSORTTHREADS= and 1 GB * MAXSORTTHREADS=.

SQLAUDLEN= Parameter File Option

Specifies the maximum length of the SQL statement in the audit log.

Valid in:	spdsserv.parm parameter file
Default:	1,024 characters
Requirements:	Proxy auditing was enabled at SPD Server start-up. For more information, see the installation information for your host environment. WHEREAUDIT is also set.

Syntax

SQLAUDLEN=*length*;

Required Argument

length
specifies the maximum length of the SQL statements in characters. The maximum allowed value is 4,096 characters.

SQLOPTS= Parameter File Option

Overrides a default SQL setting with the option specified in the RESET statement.

Valid in:	spdsserv.parm parameter file
Interaction:	Users can also specify RESET options using SQL explicit pass-through.

Syntax

SQLOPTS= "RESET *option-1* [*option-2*...];

Required Argument

"RESET *option*"
specifies the RESET statement and one or more SQL options. For more information about SQL options that can be submitted in the RESET statement, see Chapter 9, "SQL Query Rewrite Facility," on page 79.

SSLALLOWUNXDS Parameter File Option

Specifies whether TLS or UNIX domain sockets is used to secure local client/server data communication.

Valid in:	spdsserv.parm parameter file
Default:	NOSSLALLOWUNXDS

Applies to:	UNIX platform
Interaction:	(Optional) This parameter file option can be set when SSLSECURE= is set to YES or PREFERRED.
See:	Chapter 6, "Configuring Secure Sockets Communication," on page 35

Syntax

SSLALLOWUNXDS | NOSSLALLOWUNXDS

Required Arguments

NOSSLALLOWUNXDS
specifies that TLS is used to secure local client data communication.

SSLALLOWUNXDS
specifies that UNIX domain sockets communication is used to secure local client data communication. TLS is used to secure remote client data communication.

Details

When SSLSECURE= is set to YES or PREFERRED, TLS is used to secure client/server data communication for both local and remote clients that are configured to use TLS by default.

A UNIX domain socket is an inter-process communication mechanism that provides for exchanging data between processes executing on the same host operating system. All communication for UNIX domain sockets occurs within the operating system kernel, which ensures secure, efficient communication. TLS uses inter-process TCP/IP socket communication for processes on the same host. UNIX domain sockets can offer better performance than inter-process TCP/IP socket communication on the same host.

When TLS is not configured for SPD Server, local data communication between SPD Server and its clients for UNIX use UNIX domain sockets.

SSLCALISTLOC= Parameter File Option

Specifies the location of the public certificate or certificates for trusted certificate authorities (CA).

Valid in:	spdsserv.parm parameter file
Default:	None
Applies to:	UNIX platform
Interaction:	This parameter file option is required when SSLSECURE is set to YES or PREFERRED.
See:	Chapter 6, "Configuring Secure Sockets Communication," on page 35

Syntax

SSLCALISTLOC="*file-path*";

Required Argument

"file-path"
: specifies the location of a single file that contains the public certificate or certificates for all of the trusted certificate authorities (CA) in the trust chain. The CA file must be PEM-encoded (base64). For more information, see “Certificate File Formats” in *Encryption in SAS*.

SSLCERTISS= Parameter File Option

Specifies the name of the issuer of the digital certificate that TLS should use.

Valid in:	spdsserv.parm parameter file
Applies to:	Windows platform
Interactions:	This parameter file option is required when SSLSECURE= is set to YES or PREFERRED. Use SSLCERTISS= and SSLCERTSERIAL=, or use SSLCERTSUBJ=. Do not use both.
See:	Chapter 6, “Configuring Secure Sockets Communication,” on page 35

Syntax

SSLCERTISS="*issuer-of-digital-certificate*"

Required Argument

"issuer-of-digital-certificate"
: specifies the name of the issuer of the digital certificate that should be used by TLS. The SSLCERTISS= option is used with the SSLCERTSERIAL= option to uniquely identify a digital certificate from the Microsoft Certificate Store.

SSLCERTLOC= Parameter File Option

Specifies the location of the digital certificate for the machine's public key.

Valid in:	spdsserv.parm parameter file
Applies to:	UNIX platform
Interaction:	This parameter file option is required when SSLSECURE= is set to PREFERRED or YES.
See:	Chapter 6, “Configuring Secure Sockets Communication,” on page 35

Syntax

SSLCERTLOC="*file-path*";

Required Argument

“file-path”
: specifies the location of a file that contains a digital certificate for the machine's public key. This is used by servers to send to clients for authentication. The certificate must be PEM-encoded (base64). For more information, see “Certificate File Formats” in *Encryption in SAS*.

SSLCERTSERIAL= Parameter File Option

Specifies the serial number of the digital certificate that TLS should use.

Valid in:	spdsserv.parm parameter file
Applies to:	Windows platform
Interaction:	Used with the SSLCERTISS= parameter file option.
See:	Chapter 6, “Configuring Secure Sockets Communication,” on page 35

Syntax

SSLCERTSERIAL="*serial-number*";

Required Argument

"*serial-number*"
: specifies the serial number of the digital certificate that should be used by TLS. The SSLCERTSERIAL= option is used with the SSLCERTISS= option to uniquely identify a digital certificate from the Microsoft Certificate Store.

SSLCERTSUBJ= Parameter File Option

Specifies the subject name of the digital certificate that TLS should use.

Valid in:	spdsserv.parm parameter file
Applies to:	Windows platform
Interactions:	This parameter file option is required when SSLSECURE= is set to YES or PREFERRED.
	Use SSLCERTISS= and SSLCERTSERIAL=, or use SSLCERTSUBJ=. Do not use both.
See:	Chapter 6, “Configuring Secure Sockets Communication,” on page 35

Syntax

SSLCERTSUBJ="*subject-name*";

Required Argument

"*subject-name*"
specifies the subject name of the digital certificate that TLS should use. The SSLCERTSUBJ= option is used to search for a digital certificate from the Microsoft Certificate Store.

SSLCLIENTAUTH Parameter File Option

Specifies whether a server should perform client authentication.

Valid in:	spdsserv.parm parameter file
Default:	NOSSLCLIENTAUTH
Applies to:	UNIX and Windows platforms
Interactions:	(Optional) This parameter file option can be set when SSLSECURE= is set to YES or PREFERRED. Use in conjunction with the SSLCRLCHECK, SSLCRLLOC=, and SSLREQCERT= parameter file options.
See:	Chapter 6, "Configuring Secure Sockets Communication," on page 35

Syntax

SSLCLIENTAUTH | NOSSLCLIENTAUTH;

Required Arguments

SSLCLIENTAUTH
specifies that the server should perform client authentication.

TIP If you enable client authentication, a certificate for each client is needed.

NOSSLCLIENTAUTH
specifies that the server should not perform client authentication.

Details

Server authentication is always performed, but the SSLCLIENTAUTH option enables a user to control client authentication. On SPD Server, this option is needed to encrypt SQL pass-through connections.

SSLCRLCHECK Parameter File Option

Specifies whether a Certificate Revocation List (CRL) is checked when a digital certificate is validated.

Valid in:	spdsserv.parm parameter file
Default:	NOSSLCRLCHECK
Applies to:	UNIX and Windows platforms
Interaction:	Used with SSLCLIENTAUTH.
See:	Chapter 6, "Configuring Secure Sockets Communication," on page 35

Syntax

SSLCRLCHECK | NOSSLCRLCHECK;

Required Arguments

SSLCRLCHECK
specifies that CRLs are checked when digital certificates are validated.

NOSSLCRLCHECK
specifies that CRLs are not checked when digital certificates are validated.

Details

A Certificate Revocation List (CRL) is published by a Certificate Authority (CA) and contains a list of revoked digital certificates. The list contains only the revoked digital certificates that were issued by a specific CA.

SSLCRLLOC= Parameter File Option

Specifies the location of a Certificate Revocation List (CRL).

Valid in:	spdsserv.parm parameter file
Applies to:	UNIX platform
Interaction:	Used with SSLCRLCHECK.
See:	Chapter 6, “Configuring Secure Sockets Communication,” on page 35

Syntax

SSLCRLLOC="*file-path*";

Required Argument

"file-path"
specifies the location of a file that contains a Certificate Revocation List (CRL).

SSLPKCS12LOC= Parameter File Option

Specifies the location of the PKCS #12 encoding package file.

Valid in:	spdsserv.parm parameter file
Applies to:	UNIX platform
Interactions:	(Optional) This parameter file option can be set when SSLSECURE= is set to YES or PREFERRED. When SSLPKCS12LOC= is specified, SSLCERTLOC= and SSLPVTKEYLOC= are ignored.
See:	Chapter 6, “Configuring Secure Sockets Communication,” on page 35

Syntax

SSLPKCS12LOC="*file-path*";

Required Argument

"file-path"
specifies the location of the PKCS #12 DER encoding package file which must contain both the certificate and private key. For more information, see “Certificate File Formats” in *Encryption in SAS*.

SSLPKCS12PASS= Parameter File Option

Specifies the password that TLS requires for decrypting the private key.

Valid in:	spdsserv.parm parameter file
Applies to:	UNIX platform
Interaction:	Used with SSLPKCS12LOC=.
See:	Chapter 6, “Configuring Secure Sockets Communication,” on page 35

Syntax

SSLPKCS12PASS=*password*;

Required Argument

password
specifies the password that TLS requires in order to decrypt the PKCS #12 DER encoding package file. The PKCS #12 DER encoding package is stored in the file that is specified by using the SSLPKCS12LOC= option. The SSLPKCS12PASS= option is required only when the PKCS #12 DER encoding package is encrypted.

SSLPVTKEYLOC= Parameter File Option

Specifies the location of the private key that corresponds to the digital certificate.

Valid in:	spdsserv.parm parameter file
Applies to:	UNIX platform
Interaction:	This parameter file option is required when SSLSECURE= is set to YES or PREFERRED.
See:	Chapter 6, “Configuring Secure Sockets Communication,” on page 35

Syntax

SSLPVTKEYLOC="*file-path*";

Required Argument

"*file-path*"
: specifies the location of the file that contains the private key that corresponds to the digital certificate that was specified by using the SSLCERTLOC= option. The key must be PEM-encoded (base64). For more information, see "Certificate File Formats" in *Encryption in SAS*.

SSLPVTKEYPASS= Parameter File Option

Specifies the password that TLS requires for decrypting the private key.

Valid in:	spdsserv.parm parameter file
Applies to:	UNIX platform
Interaction:	Used with SSLPVTKEYLOC=.
See:	Chapter 6, "Configuring Secure Sockets Communication," on page 35

Syntax

SSLPVTKEYPASS=*password*;

Required Argument

password
: specifies the password that TLS requires in order to decrypt the private key. The private key is stored in the file that is specified by using the SSLPVTKEYLOC= option. The SSLPVTKEYPASS= option is required only when the private key is encrypted. OpenSSL performs key encryption. No SAS system option is available to encrypt private keys.

SSLREQCERT= Parameter File Option

Specifies the protocol that is used to exchange digital certificates at your site

Valid in:	spdsserv.parm parameter file
Applies to:	UNIX platform
Interaction:	Used with SSLCLIENTAUTH.
See:	Chapter 6, "Configuring Secure Sockets Communication," on page 35

Syntax

SSLREQCERT= ALLOW | DEMAND | NEVER | TRY;

Required Arguments

ALLOW
: The server asks for a certificate. If a certificate is not provided, then the session proceeds. If a certificate is provided and if it fails to validate, then the session proceeds.

DEMAND
The server asks for a certificate. If the certificate fails to validate, then the session is immediately terminated. This is the default value if TLS is enabled on SPD Server and the SSLREQCERT= option is omitted from the spdsserv.parm parameter file.

NEVER
The server does not ask for a certificate.

TRY
The server asks for a certificate. If a certificate is not provided, then the session proceeds. If a certificate is provided, and if the certificate fails to validate, then the session is immediately terminated.

SSLSECURE= Parameter File Option

Enables secure sockets communication for SPD Server.

Valid in:	spdsserv.parm parameter file
Default:	YES
Applies to:	UNIX and Windows platforms
See:	Chapter 6, “Configuring Secure Sockets Communication,” on page 35

Syntax

SSLSECURE=NO | PREFERRED | YES;

Required Arguments

NO
specifies that sockets communication is not secured.

PREFERRED
specifies that socket communication is secured if the client is capable of secure communication; otherwise, socket communication is not secure.

YES
requires secure sockets communication for the server. Clients that are not capable of secure socket communication cannot connect to the server.

Details

SPD Server uses Transport Layer Security (TLS) to provide secure sockets communication. Secure sockets communication is supported for SAS clients only in SPD Server 5.3. Change the SSLSECURE= to PREFERRED instead of YES if your server will support JDBC and ODBC clients in addition to SAS clients.

STARSIZE= Parameter File Option

Specifies the amount of memory to allocate for STARJOIN operations.

Valid in:	spdsserv.parm parameter file
Interaction:	When STARSIZE= is omitted, SORTSIZE= applies. See “SORTSIZE= Parameter File Option” on page 102.

Syntax

STARSIZE=*n* | *n*K | *n*M | *n*G;

Required Argument

n
: specifies a number that indicates the amount of memory to allocate for STARJOIN operations in terms of bytes (1), kilobytes (1,024 bytes), megabytes (1,048,576 bytes), or gigabytes (1,073,741,824 bytes).

Details

During STARJOIN operations, the temporary results of Phase 1 of the IN-SET STARJOIN strategy are cached in memory for use by Phase 2 if there is sufficient STARSIZE= memory. Caching Phase 1 temporary results can result in significant performance improvements for STARJOIN. If you omit STARSIZE=, STARJOIN uses the SORTSIZE= option to determine the memory to use for Phase 1 caching.

TMPDOMAIN= Parameter File Option

Specifies a storage location for intermediate tables created by the SQL Query Rewrite Facility.

Valid in:	spdsserv.parm parameter file
Requirement:	TMPDOMAIN= is required by the SQL query rewrite facility.
See:	Chapter 9, "SQL Query Rewrite Facility," on page 79.

Syntax

TMPDOMAIN= *domain-name*;

Required Argument

domain-name
: specifies the name of the SPD Server domain that was defined to store temporary tables in the libnames.parm parameter file.

WHAUDLEN= Parameter File Option

Specifies the maximum length of the WHERE clause in the audit log.

Valid in:	spdsserv.parm parameter file
Default:	512 characters
Applies to:	WHERE clause auditing
Requirements:	Proxy auditing was enabled at SPD Server start-up. For more information, see the installation information for your host environment. WHEREAUDIT is also set.

Syntax

WHAUDLEN= *length*

Required Argument

length
specifies the maximum length of the WHERE clauses in characters. The maximum allowed value is 4,096 characters.

WHEREAUDIT Parameter File Option

Specifies whether SPD Server audits WHERE clauses.

Valid in:	spdsserv.parm parameter file
Requirement:	Proxy auditing was enabled at SPD Server start-up. For more information, see the installation information for your host environment.
See:	"WHERE Clause Auditing" on page 182

Syntax

[NO]**WHEREAUDIT**;

Optional Arguments

NOWHEREAUDIT;
disables WHERE clause auditing.

WHEREAUDIT;
enables WHERE clause auditing.

WHERECOSTING Parameter File Option

Enables dynamic WHERE clause costing on the server.

Valid in:	spdsserv.parm parameter file
Default:	WHERECOSTING

Syntax

[NO]**WHERECOSTING**;

Optional Arguments

NOWHERECOSTING
disables dynamic WHERE clause costing.

WHERECOSTING
enables dynamic WHERE clause costing.

Details

WHERE clause costing causes WHINIT to use a heuristic to determine (among other things) what indexes to use to evaluate the WHERE clause. When WHERE clause costing is disabled by specifying NOWHERECOSTING, WHINIT uses a rules-based evaluation that includes all available indexes to evaluate the WHERE clause. The default setting enables users to use the SPDSWCST macro variable to control whether WHERE clause costing or WHINIT is used. When NOWHERECOSTING is set, the SPDSWCST macro variable has no effect.

WORKPATH= Parameter File Option

Specifies the path for work files created by applications.

Valid in: spdsserv.parm parameter file

Syntax

WORKPATH= *location* | (*location-1 location-2* ...);

Required Arguments

location

specifies the location of an existing directory for work files that are created by applications. Enclose the location in single or double quotation marks when the location contains spaces.

TIP When SPD Server is installed on a cluster with multiple nodes running SPD Server, you can use the keyword @HOSTNAME@ to substitute the short host name in a directory path. Here is an example:

```
workpath=/work/@HOSTNAME@;
```

In a multi-node installation, it is advisable that each server has its own WORKPATH. Use of the @HOSTNAME@ substitution allows each server to use the same spdsserv.parm parameter file with per-host WORKPATH paths.

(*location-1 location-2*...)

specifies a list of existing directories that can be accessed in parallel for work files created by applications. Enclose a location in single or double quotation marks when the location contains spaces.

Chapter 11
Using SPD Server with an Internet Firewall

Using SAS Scalable Performance Data (SPD) Server with an Internet Firewall . 115
Overview of Using SPD Server with a Firewall 115
Assigning SPD Server Ports That Require Firewall Access 116

Using SAS Scalable Performance Data (SPD) Server with an Internet Firewall

Overview of Using SPD Server with a Firewall

SPD Server and its clients communicate through ports that permit requests to be sent to the server and that send and receive data (such as table rows) between client and server. If the server is running with an Internet firewall, the ports that the client and server use must be configured so that the firewall allows the communication. This section describes the SPD Server server and client ports, as well as how to assign and configure them for use with an Internet firewall.

Server clients communicate with the name server via the name server listen port. The name server listen port is used by clients (such as Base SAS) when LIBNAME and SQL CONNECT statements are issued. The LIBNAME and SQL CONNECT statements must be able to pass through a firewall. The name server listen port is also used by ODBC data sources that need to communicate with the name server.

Server clients communicate with the SPD Server host whenever a client needs to complete a LIBNAME connection, or whenever a client needs to issue server operator commands. LIBNAME connections and operator commands must be able to access the SPD Server listen port and the SPD Server operator port through existing firewalls.

When a SPD host server completes a client request for a LIBNAME connection, it creates an SPD Server base user proxy process. The user proxy handles all of the client data requests. The proxy process requires multiple ports: a port to receive data commands from the client, a port to receive operator commands from the client, and a port for each open table to send and receive data between client and server. Therefore, the SPD Server Base user proxy requires a range of port numbers that must be accessible through the firewall.

Assigning SPD Server Ports That Require Firewall Access

SPD Server Name Server Listen Port

You can specify the name server listen port by using well-known port definitions that are declared in the operating system's services file. You can also use the SPD Server command-line interface to specify the listen port. In the services file, the **spdsname** specification corresponds to the listen port. For UNIX installations, you can define the name server listen port in the rc.spds start-up script. The NSPORT parameter in the rc.spds start-up script defines the name server listen port. If NSPORT is not defined in the rc.spds start-up script, the name server uses the **spdsname** service entry.

SPD Server Listen Port and SPD Server Operator Port

You can specify the SPD Server listen and operator ports by using well-known port definitions that are declared in the operating system's services file. You can also use the SPD Server command-line interface. In the operating system's services file, the spdsserv_sas specification corresponds to the SPD Server listen port. The spdsserv_oper specification corresponds to the SPD Server operator port. For UNIX installations, you can also define the SPD Server listen and operator ports in the rc.spds start-up script for UNIX installations. In an rc.spds start-up script, the SRVLPORT parameter defines the listen port, and the SRVOPORT parameter defines the operator port. If the listen and operator ports are not defined, or are defined as zero values, the SPD Server uses spdsserv_sas and **spdsserv_oper** in the operating system's services file. If there are no listen or operator ports defined in the operating system's services file, then SPD Server chooses any available ports for listen and operator port functions by default. This is the normal mode of operation when SPD Server clients and servers run in environments that do not have firewalls.

SPD Server Base Proxy Ports

You must use the SPD Server MINPORTNO= and MAXPORTNO= server parameter specifications to define the available range of ports for the SPD Server Base Proxy processes. You must specify both the MINPORTNO= and MAXPORTNO= parameters when you define the range of port numbers that are available to communicate with SPD Server clients that might be outside of a firewall. If the SPD Server parameters for MINPORTNO= and MAXPORTNO= are not specified, an SPD Server Base proxy process uses any port that is available to communicate with its SPD Server client. This is the normal mode of operation when SPD Server clients and servers run in environments that do not have firewalls.

How many port numbers do you need to reserve for SPD Server Base user proxy processes? Each SPD Server Base user proxy process produces its own command port. You can access the command port via command-line specifications that are issued by an SPD Server client. You can access the operator port for a command port by using PROC SPDO operator commands.

Each SPD Server host table that is opened also creates its own port. Each SPD Server table port becomes a dedicated data transfer connection that is used to stream data transfers to and from the SPD Server client. SPD Server host table ports are normally assigned dynamically, unless MINPORTNO= and MAXPORTNO= parameters have been specified.

If MINPORTNO= and MAXPORTNO= parameters have been specified, then SPD Server host table ports are assigned from within the port range that is defined by the minimum and maximum port parameter statements. The port range that is specified by the MINPORTNO= and MAXPORTNO= parameters must be able to accommodate the maximum number of concurrent LIBNAME connections required at the server, as well as the I/O data streams that travel between the SPD Server Base processes on the host and the SPD Server clients.

Chapter 12

Configuring and Administering SPD Server Using the SAS Management Console

Overview of SAS Management Console 119

Accessing SPD Server Services in SAS Management Console 120

Connect to an SPD Server 120

Password Manager 122
Overview of Password Manager 122
Managing Users with Password Manager 122
Managing Groups with Password Manager 125

ACL Manager 126
Overview of ACL Manager 126
List ACL Resources 126
Add an ACL Resource 128
Delete an ACL Resource 129
Add a User or Group to an ACL Resource 129
Change Resource Permissions 129

Server Manager 129
Overview of Server Manager 129
Refresh Domains 130
Refresh Configuration Information 131
Run Commands 131

Process Profiler 132

Proxy Manager 134
Overview of Proxy Manager 134
Refresh Proxies 135
Interrupt Proxies 135
Cancel Proxies 135

Overview of SAS Management Console

SAS Management Console is a Java application that provides a single point of control for managing multiple SAS application resources. Rather than using a separate administrative interface for each application in your enterprise intelligence environment, you can use the SAS Management Console interface to perform the administrative tasks required to create and maintain an integrated environment.

SAS Management Console manages resources and controls by creating and maintaining metadata definitions for entities such as the following:

- server definitions
- library definitions
- user definitions
- resource access controls
- metadata repositories
- job schedules

Metadata definitions that are created through SAS Management Console are stored in a repository on a SAS Metadata Server where they are available for other applications to use. For example, you can use SAS Management Console to create a metadata definition for a SAS library that specifies information such as the libref, path, and engine type (such as SASSPDS). After SAS Management Console stores the metadata definition for the library in the repository on the SAS Metadata Server, any other application can access the definition to access the specified library.

The SAS Management Console application is a framework. The metadata definitions are created using Java plug-ins, which are application modules that create metadata for a specific type of resource.

For example, administrators can use SAS Management Console to configure SPD Server user and group passwords and ACLs instead of using the traditional SPD Server psmgr utility and SPDO procedure statements.

CAUTION:

ACLs created with SAS Management Console cannot be recovered with the console if the ACLs are damaged or somehow lost. The SPDO procedure enables you to save your ACL code, so that if a problem occurs, your ACLs can be reapplied. You might want to use PROC SPDO to create your ACLs instead of SAS Management Console.

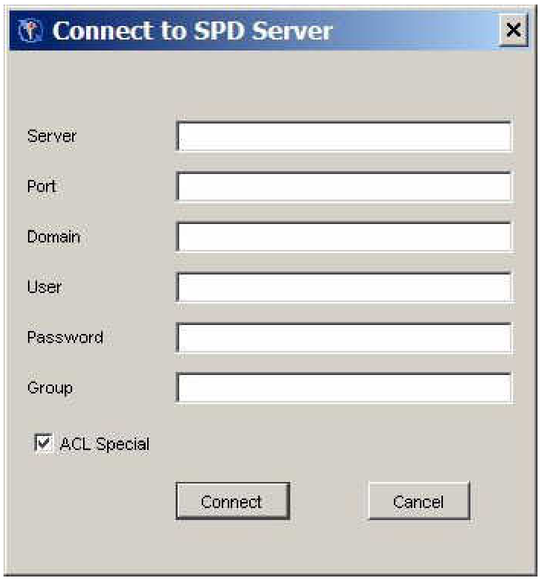
Accessing SPD Server Services in SAS Management Console

The left portion of SAS Management Console contains a navigation tree of available management tools. Click the **SPD Manager** folder to access SPD Server services. The **SPD Manager** folder contains the SPD Server Password Manager, ACL Manager, Server Manager, Process Profiler, and Proxy Manager.

Before you can access any of the SPD Server Management utilities, you must connect to an SPD Server. For more information, see "Connect to an SPD Server" on page 120.

Connect to an SPD Server

On the **Users** tab in the SAS Management Console window, click **Connect** to open the Connect to SPD Server window:



The Connect to SPD Server window contains input fields for the following components. The components in the window follow the same usage rules as the components that you would use in a LIBNAME statement to connect to an SPD Server host.

Server
: name of the SPD Server host

Port
: the SPD Server name server listen port

Domain
: the SPD Server domain

User
: user name that has special privilege on SPD Server.

Password
: password associated with the user name

Group
: optional group name

ACL Special
: select this box to invoke the user’s special privilege in the LIBNAME session. If this box is not checked, the user will connect as a regular user.

Complete the required information, and then click **Connect**. After you connect to an SPD Server host, you can use the SPD Server Management utilities. The status bar at the bottom of the active tab indicates that a connection exists.

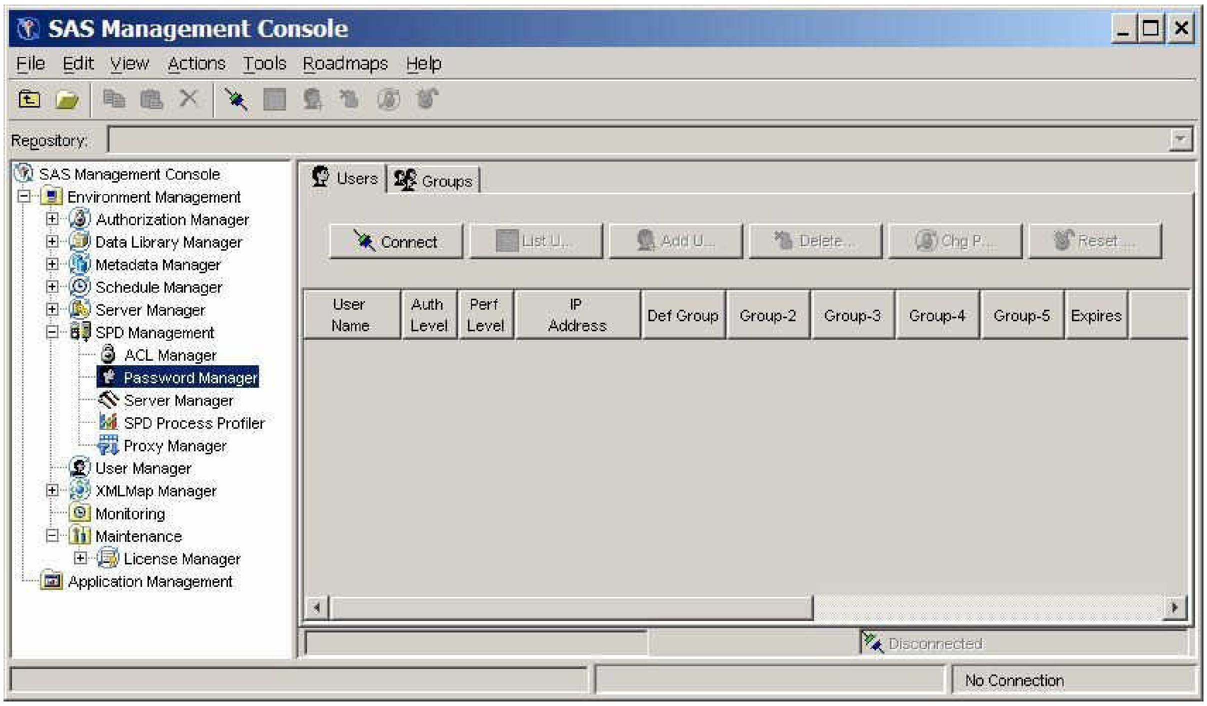
Note: Do not confuse the SAS Management Console status indicator in the lower right corner of the SAS Management Console window with the SPD Server connection status at the bottom of the active tab.

Password Manager

Overview of Password Manager

You can use Password Manager in the SAS Management Console window to configure SPD Server user and group passwords. Use the **Users** and **Groups** tabs of the Password Manager to access the configuration information for individual users or for user groups.

If you open Password Manager in the SAS Management Console window when no server connection exists, the display resembles the following:

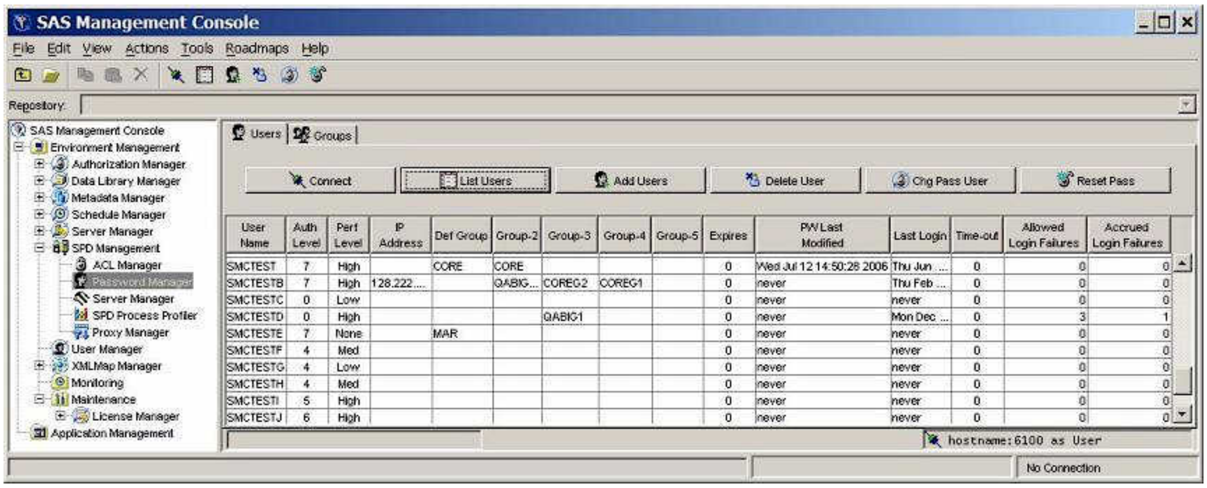


When no server connection exists, no data is displayed. To display data, you must connect to an SPD Server. For more information, see "Connect to an SPD Server" on page 120.

Managing Users with Password Manager

Users Tab

On the **Users** tab in the SAS Management Console window, click **List Users** to display the defined users and groups.



User Name	Auth Level	Perf Level	IP Address	Def Group	Group-2	Group-3	Group-4	Group-5	Expires	PW Last Modified	Last Login	Time-out	Allowed Login Failures	Accrued Login Failures
SMCTEST	7	High		CORE	CORE				0	Wed Jul 12 14:50:28 2006	Thu Jun ...	0	0	0
SMCTESTB	7	High	128.222....		QABIG...	COREG2	COREG1		0	never	Thu Feb ...	0	0	0
SMCTESTC	0	Low							0	never	never	0	0	0
SMCTESTD	0	High				QABIG1			0	never	Mon Dec ...	0	3	1
SMCTESTE	7	None		MAR					0	never	never	0	0	0
SMCTESTF	4	Med							0	never	never	0	0	0
SMCTESTG	4	Low							0	never	never	0	0	0
SMCTESTH	4	Med							0	never	never	0	0	0
SMCTESTI	5	High							0	never	never	0	0	0
SMCTESTJ	6	High							0	never	never	0	0	0

The window contains the following components:

User Name
name of the user. You cannot change this field directly. To change a user name, delete the user and then add a new user. For more information, see "Delete a User" on page 124 and "Add a User" on page 124.

Auth Level
numeric authorization level, in the range 0–7. To change the value, select the field and edit it.

Perf Level
this setting is not yet implemented in SPD Server. In the future, this field will be used to indicate to the server how to manage resources for the associated user.

IP Address
IP address of the workstation that is being used by the individuals listed in the User Name column.

Def Group
default group for a user. Use the drop-down list to change the currently defined group.

Group 2 - Group 5
shows the numbered groups 2–5 that are assigned to each user. Use the drop-down list to change the currently defined groups.

Expires
number of days remaining until the current password expires. The value 0 indicates that the password never expires.

PW Last Modified
date and time when a user's password was last modified.

Last Login
date and time of the last user login.

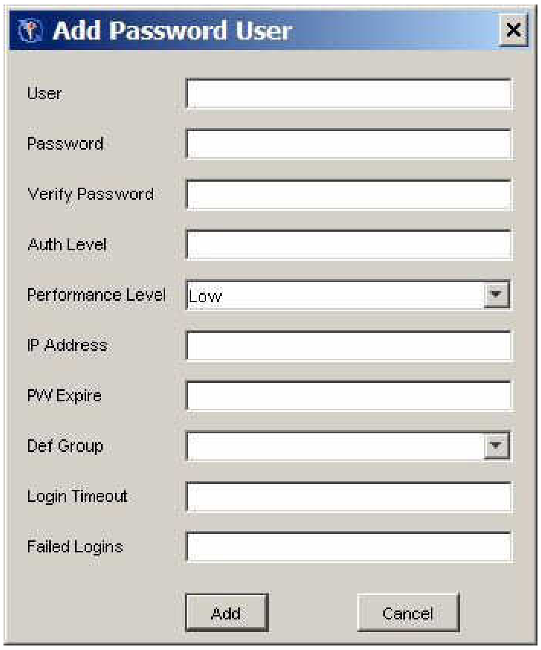
Time-out
number of idle days since the user's last login. When the number of days since a user's last login equals the **Time-out** value, the user's access is disabled. For example, if the **Time-out** value is 7, and if a given user does not log in at least every seven days, then the user's access is disabled. The value 0 indicates that the user account never times out.

Allowed Login Failures
number of consecutive login failures that is allowed before the user's access is disabled. The value 0 indicates that unlimited login failures are allowed.

Accrued Login Failures
current number of consecutive failed login attempts by this user.

Add a User

To add a user, click **Add Users**. Complete the values for **User**, **Password**, **Verify Password**, **Auth Level**, **Performance Level**, **IP Address**, **PW Expire**, **Def Group**, **Login Timeout**, and **Failed Logins**. The **Verify Password** field requires you to re-enter the password to ensure that the two items match and that you did not make any keyboard errors when you entered the password string. Performance Level reflects the performance class of the user and should be **Low**, **Medium**, or **High**. For more information about performance levels, see "Server Performance Levels" on page 85.


Add Password User

User	
Password	
Verify Password	
Auth Level	
Performance Level	Low
IP Address	
PW Expire	
Def Group	
Login Timeout	
Failed Logins	

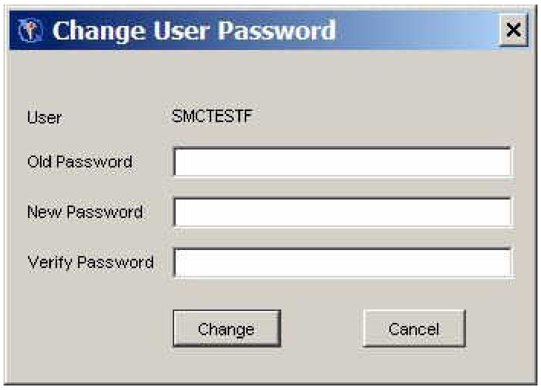
User, **Password**, **Auth Level**, **Def Group**, and **Failed Logins** are required values. The **IP Address**, **PW Expire**, and **Login Timeout** values are optional. If you omit the optional settings, they default to "no limits".

Delete a User

To delete a user, select the user from the list and click **Delete**. The user is deleted and the list is updated.

Change a Password

To change a user's password, select the user and click **Chg Pass User**. Enter the user's old password and new password in the Change User Password dialog box, and click **Change**.



Reset a Password

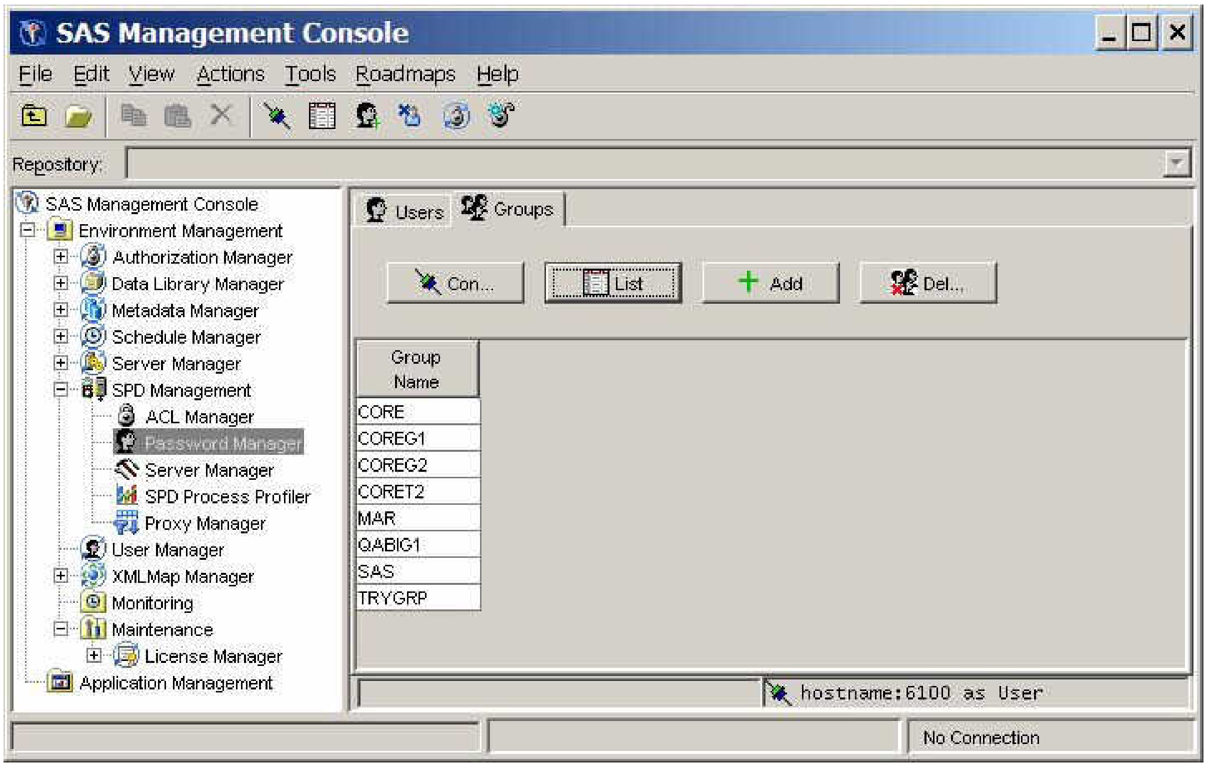
To reset the password for a selected user, click **Reset Pass**. Specify the user's new password and click **Change**. You do not need to know the user's current password to change it. The user must change the password before regaining the ability to connect to the server.

You use **Reset Pass** to reset a user's password after that user has been disabled. Users can be disabled for excessive login failures or for a login time-out.

Managing Groups with Password Manager

Groups Tab

On the **Groups** tab in the SAS Management Console window, click **List** to display the existing defined groups. When you first connect to the Password Manager or after you make changes to a group, the window lists the existing defined groups by default.



Group Name
CORE
COREG1
COREG2
CORET2
MAR
QABIG1
SAS
TRYGRP

Add a Group

To add a group, click **Add**. Enter a group name in the Add Group window and click **Add**. The group name is added, and the list is updated.

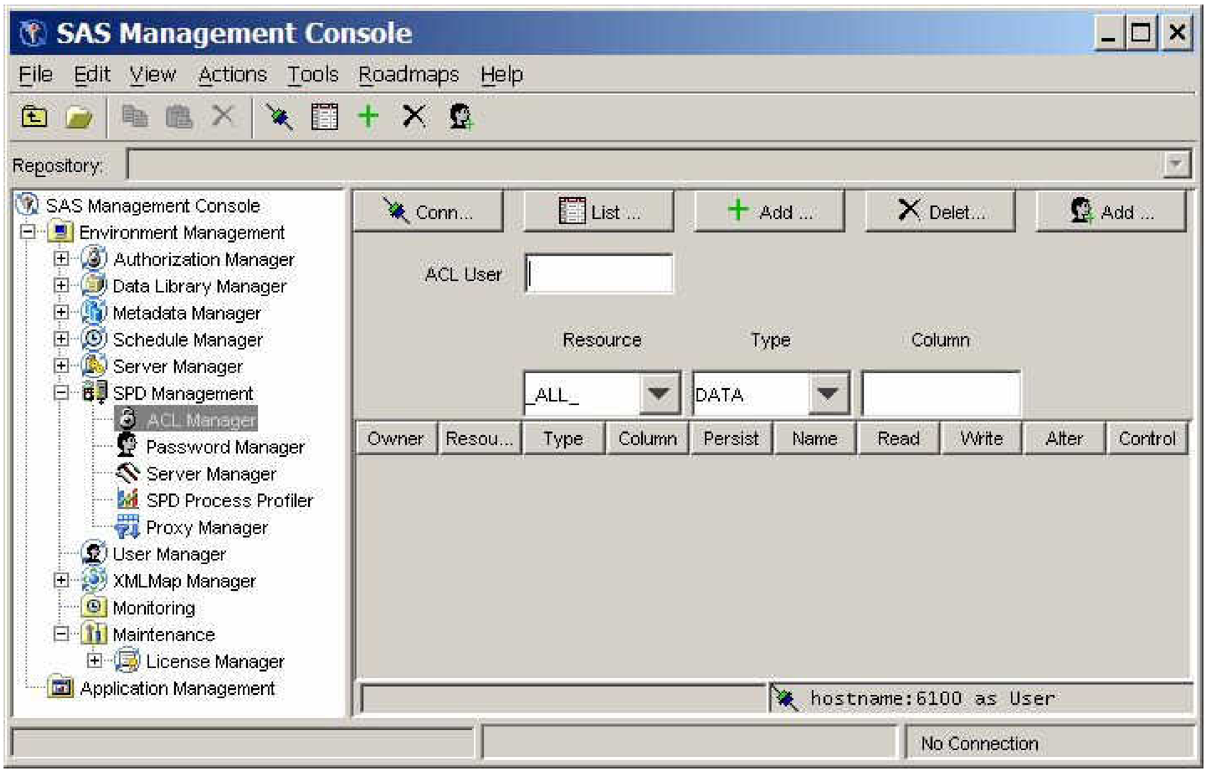
Delete a Group

To delete a group, select the group from the list and click **Delete**. The group is deleted and the list is updated.

ACL Manager

Overview of ACL Manager

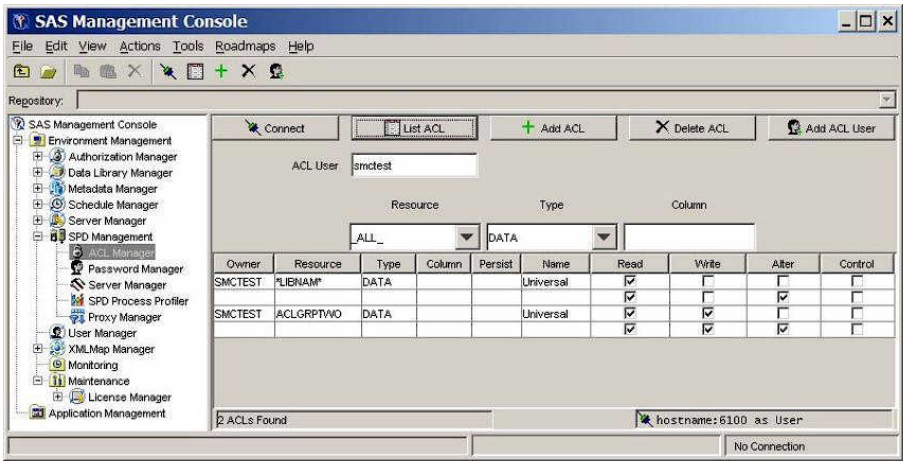
Click the ACL Manager folder in the SAS Management Console window to manage ACL resources. The ACL Manager panel resembles the following display:



You must connect to an SPD Server host machine before you can use the SPD Management utilities. For information about connecting to an SPD Server host, see "Connect to an SPD Server" on page 120.

List ACL Resources

To display the ACL resources that have been defined, click **List ACL** in the ACL Manager panel of the SAS Management Console window.



ACL User: smctest

Resource: _ALL_ Type: DATA Column:

Owner	Resource	Type	Column	Persist	Name	Read	Write	Alter	Control
SMCTEST	*LIBNAM*	DATA			Universal	✓	☐	☐	☐
						✓	☐	✓	☐
SMCTEST	ACLGRPTWO	DATA			Universal	✓	✓	☐	☐
						✓	✓	✓	☐

2 ACLs Found

hostname:6100 as User

No Connection

The ACL Manager display contains the following components:

Owner
: resource owner. You cannot change this field directly. To change a resource owner, delete the resource, and then add a new one.

Resource
: resource name. You cannot change this field directly. To change a resource name, delete the resource, and then add a new one.

Type
: type of resource (for example, DATA, CATALOG, or VIEW). You cannot change this field directly. To change the type of a resource, delete the current resource, and then add a new one.

Column
: column name, if the resource is limited by a column constraint. You cannot change the column name directly. To change the column name, delete the existing resource and then add a new one.

Persist
: a Boolean flag. When this value is set to *Yes*, the ACL resource definition continues to exist if the referenced resource is deleted. If the **Persist** setting is blank, the ACL resource definition is deleted when the referenced resource is deleted.

Name
: name of a user or group to which the Read, Write, Alter, and Control permissions are applied for this resource. **Universal** is the default setting for all unnamed groups or users.

Read
: if this check box is selected, the user or group has permission to read this resource.

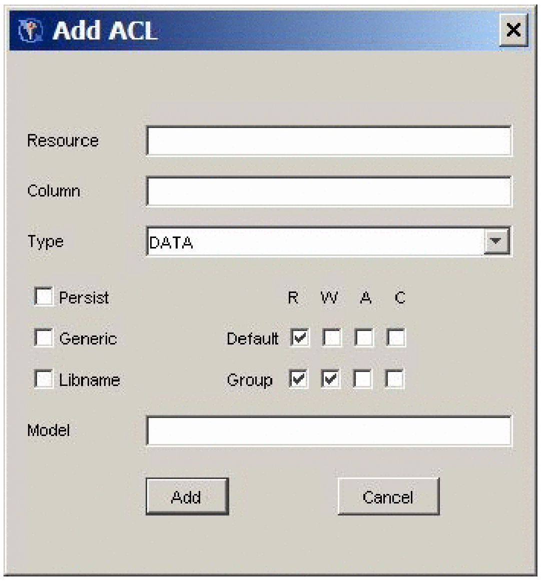
Write
: if this check box is selected, the user or group has permission to write to this resource.

Alter
: if this check box is selected, the user or group has permission to alter this resource.

Control
: if this check box is selected, the user or group has permission to modify the permissions of other users and groups that are associated with this resource.

Add an ACL Resource

To add an ACL resource, click **Add ACL** in the ACL Manager panel of the SAS Management Console window.



Add ACL

Resource

Column

Type DATA

☐ Persist
☐ Generic
☐ Libname

	R	W	A	C
Default	✓	☐	☐	☐
Group	✓	✓	☐	☐

Model

Specify the following values in the Add ACL window:

Resource
name of the resource that you are adding.

Column
column restrictions for the resource that you are adding. If there are no restrictions, leave this field blank.

Type
type of the resource (for example, DATA, CATALOG, or VIEW).

Persist
a Boolean flag. If you select this check box, the ACL resource definition continues to exist if the referenced resource is deleted. If you do not select this check box, the ACL resource definition is deleted when the referenced resource is deleted.

Generic
select this check box if the resource name is a generic name.

LIBNAME
select this check box if the resource is a LIBNAME resource.

R, W, A, C
select the appropriate default and group permissions to grant Read, Write, Alter, and Control permission to the resource.

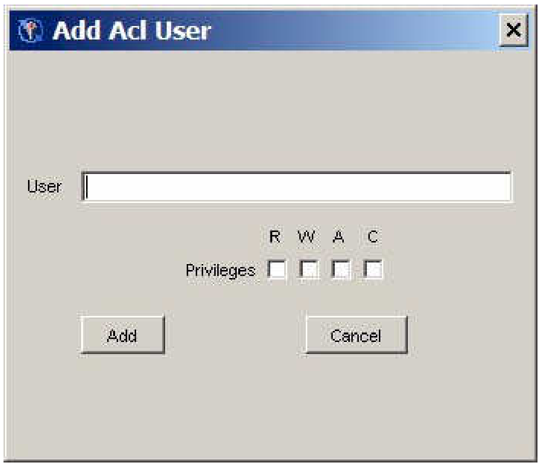
Model
specify the name of an existing ACL resource to use as a model for this ACL resource.

Delete an ACL Resource

To delete an ACL resource, select the appropriate row in the ACL resource table and click **Delete ACL**. The ACL resource is removed, and the list is automatically updated.

Add a User or Group to an ACL Resource

To add a user or group to an ACL resource, click **Add ACL User** in the ACL Manager panel. In the Add Acl User window, enter the user or group name in the **User** field. Select the boxes that correspond to the default Read, Write, Alter, and Control permissions that you want to grant. Then click **Add**.



The user is added, and the ACL listing is automatically updated. You cannot delete an individual user or group from an ACL resource. To delete a user, you must delete the entire ACL resource, and then add it, omitting that user.

Change Resource Permissions

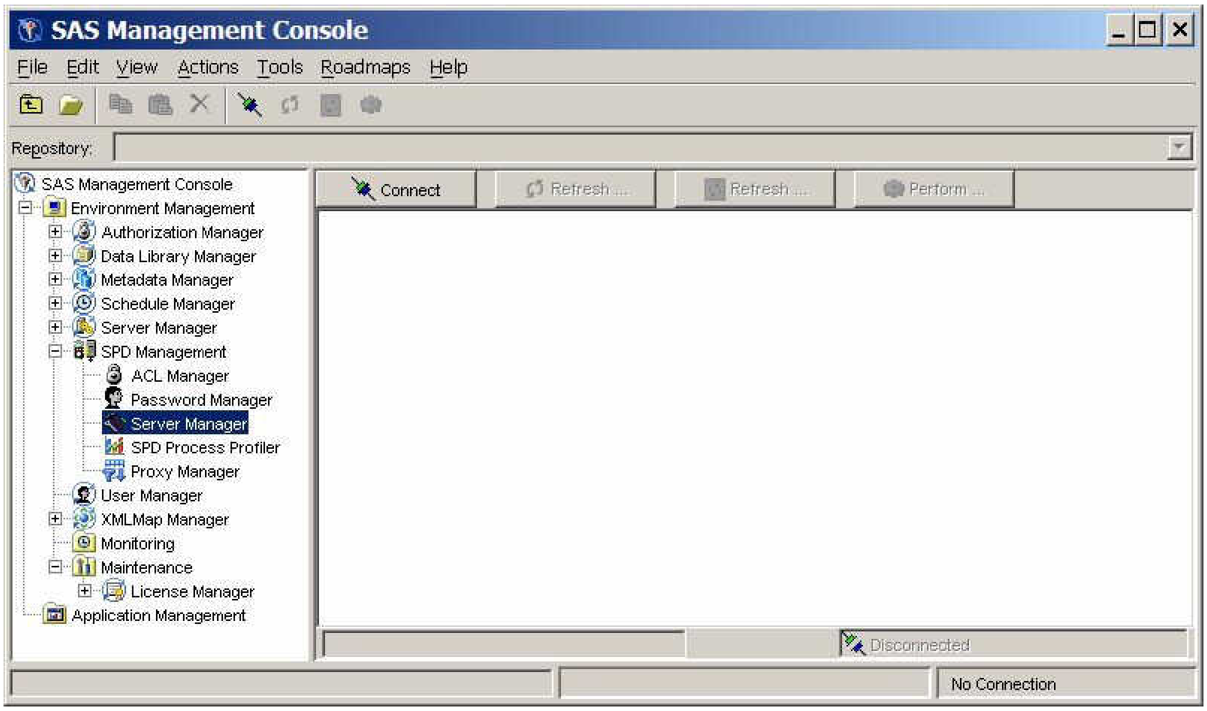
Each ACL resource has at least one set of permissions called universal permissions. Universal permissions are the default permissions for the ACL resource if no other permissions are applied. If any group names or user names have permissions for the ACL resource, they are displayed.

Each set of permissions has four attributes: Read, Write, Alter, and Control. To enable a default and group permission for each resource, select the appropriate check box.

Server Manager

Overview of Server Manager

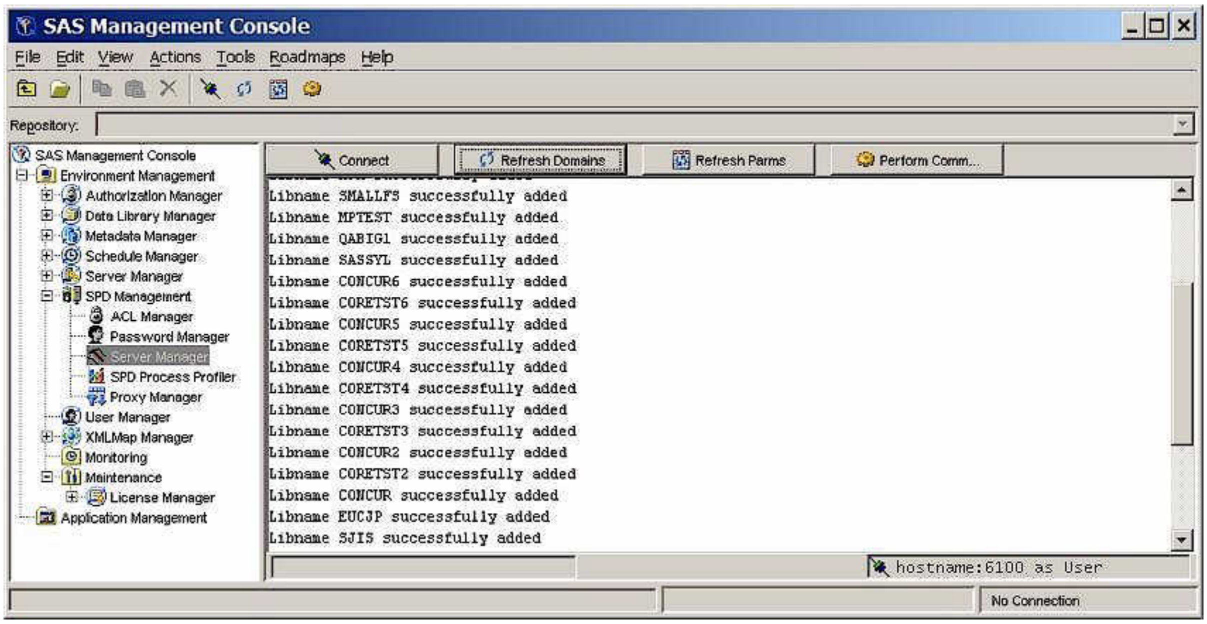
Click the **Server Manager** folder in the SAS Management Console window to refresh the SPD Server configuration and to run selected SPD Server utilities. The Server Manager panel resembles the following display when no server connection exists:



You must connect to an SPD Server host machine before you can use the SPD Server Management utilities. For information about connecting to an SPD Server host, see "Connect to an SPD Server" on page 120.

Refresh Domains

To reload the current libnames.parm parameter file, click **Refresh Domains** in the Server Manager panel of the SAS Management Console window. The libnames.parm parameter file describes the list of available domains.

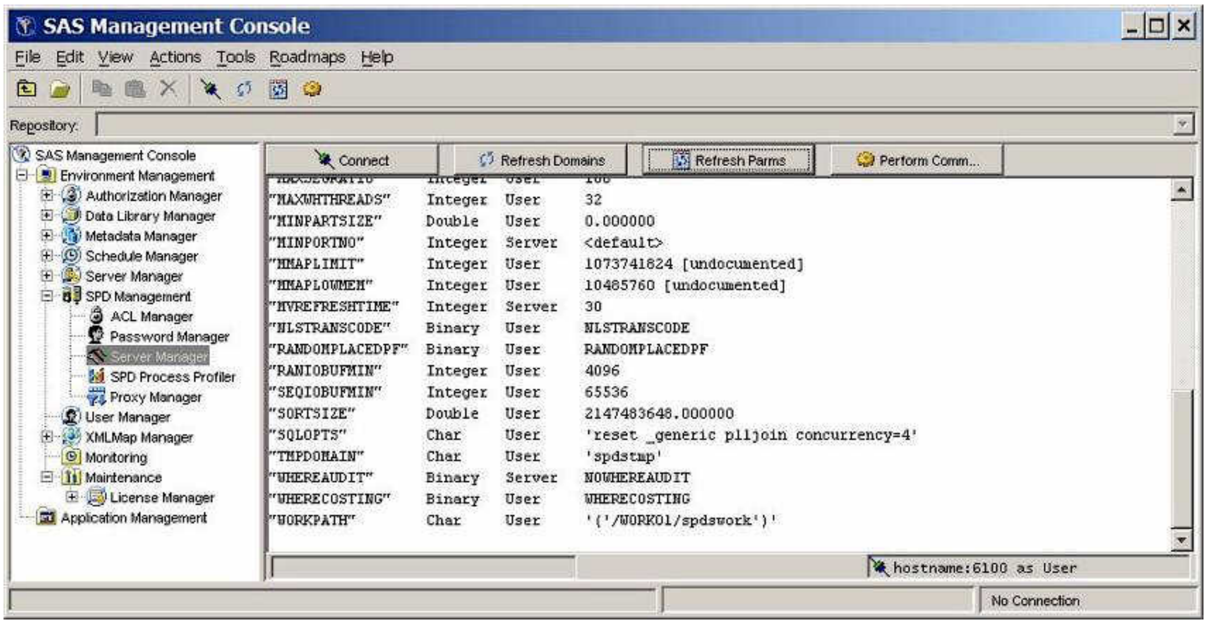


```
Libname SMALLFS successfully added
Libname MPTEST successfully added
Libname QABIG1 successfully added
Libname SASSYL successfully added
Libname CONCUR6 successfully added
Libname CORETST6 successfully added
Libname CONCUR5 successfully added
Libname CORETST5 successfully added
Libname CONCUR4 successfully added
Libname CORETST4 successfully added
Libname CONCUR3 successfully added
Libname CORETST3 successfully added
Libname CONCUR2 successfully added
Libname CORETST2 successfully added
Libname CONCUR successfully added
Libname EUCJP successfully added
Libname SJIS successfully added
```

```
hostname:6100 as User
```

Refresh Configuration Information

To reload the current spdsserv.parm parameter file, click **Refresh Parms** in the SPD Server Manager panel. The spdsserv.parm parameter file controls the server configuration and options.

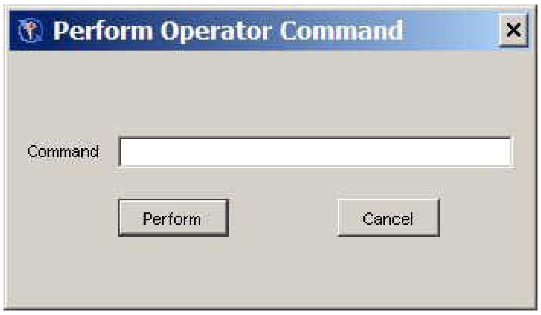


```
"MAXWHTHREADS"     Integer  User    32
"MINPARTSIZE"      Double   User    0.000000
"MINPORTNO"        Integer  Server  <default>
"MMAPLIMIT"        Integer  User    1073741824 [undocumented]
"MMAPLOWMEM"       Integer  User    10485760 [undocumented]
"MVREFRESHTIME"    Integer  Server  30
"NLSTRANSCODE"     Binary   User    NLSTRANSCODE
"RANDOMPLACEDPF"   Binary   User    RANDOMPLACEDPF
"RANIOBUFMIN"      Integer  User    4096
"SEQIOBUFMIN"      Integer  User    65536
"SORTSIZE"         Double   User    2147483648.000000
"SQLOPTS"          Char     User    'reset _generic plljoin concurrency=4'
"TMPDOMAIN"        Char     User    'spdstmp'
"WHEREAUDIT"       Binary   Server  NOWHEREAUDIT
"WHERECOSTING"     Binary   User    WHERECOSTING
"WORKPATH"         Char     User    '('/WORK01/spdswork')'
```

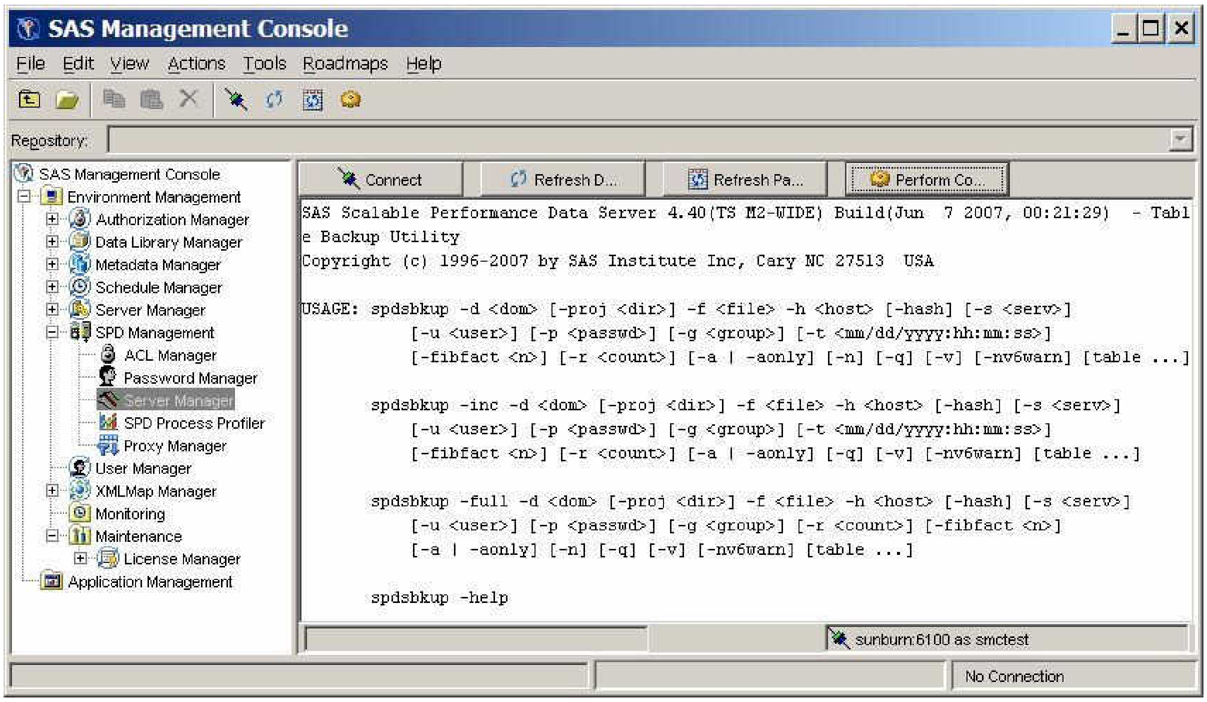
```
hostname:6100 as User
```

Run Commands

To run an SPD Server operator command or utility function, click **Perform Command** in the Server Manager panel. Enter the command or utility in the Perform Operator Command window, and then click **Perform**.



The following example shows the Server Manager panel after the **spdsbkup** command was issued:

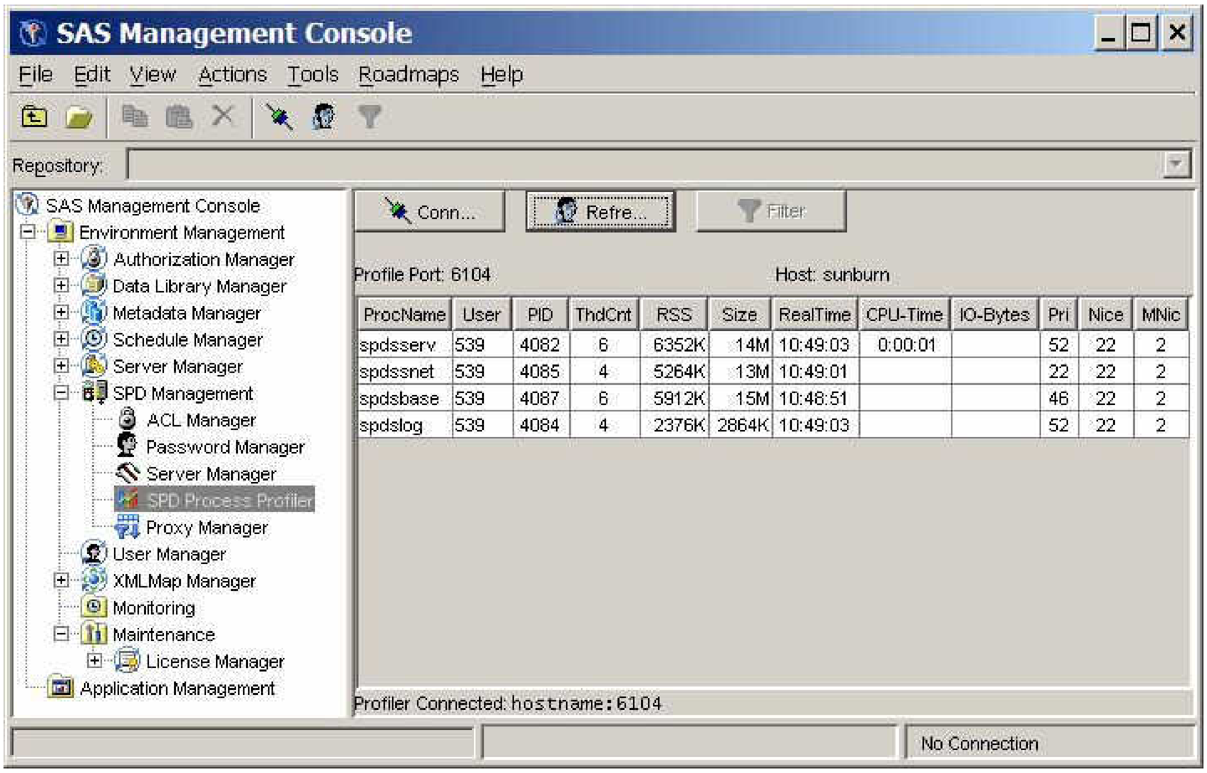


```
SAS Scalable Performance Data Server 4.40(TS M2-WIDE) Build(Jun  7 2007, 00:21:29)  - Tabl
e Backup Utility
Copyright (c) 1996-2007 by SAS Institute Inc, Cary NC 27513  USA

USAGE: spdsbkup -d <dom> [-proj <dir>] -f <file> -h <host> [-hash] [-s <serv>]
           [-u <user>] [-p <passwd>] [-g <group>] [-t <mm/dd/yyyy:hh:mm:ss>]
           [-fibfact <n>] [-r <count>] [-a | -aonly] [-n] [-q] [-v] [-nv6warn] [table ...]

       spdsbkup -inc -d <dom> [-proj <dir>] -f <file> -h <host> [-hash] [-s <serv>]
           [-u <user>] [-p <passwd>] [-g <group>] [-t <mm/dd/yyyy:hh:mm:ss>]
           [-fibfact <n>] [-r <count>] [-a | -aonly] [-q] [-v] [-nv6warn] [table ...]

       spdsbkup -full -d <dom> [-proj <dir>] -f <file> -h <host> [-hash] [-s <serv>]
           [-u <user>] [-p <passwd>] [-g <group>] [-r <count>] [-fibfact <n>]
           [-a | -aonly] [-n] [-q] [-v] [-nv6warn] [table ...]

       spdsbkup -help
```

Process Profiler

Click the **Process Profiler** folder in the SAS Management Console window to view server resources that are monitored by the Performance Server.

Note: The SPD Server Process Profile Manager is not supported on Windows or on Linux on X64.



Profile Port: 6104 Host: sunburn

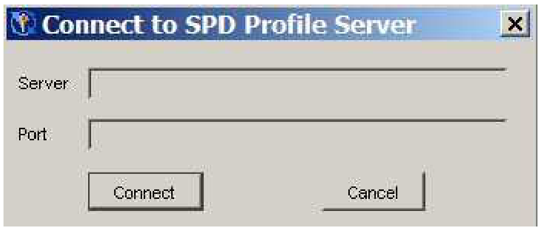
ProcName	User	PID	ThdCnt	RSS	Size	RealTime	CPU-Time	IO-Bytes	Pri	Nice	MNic
spdsserv	539	4082	6	6352K	14M	10:49:03	0:00:01		52	22	2
spdssnet	539	4085	4	5264K	13M	10:49:01			22	22	2
spdsbase	539	4087	6	5912K	15M	10:48:51			46	22	2
spdslog	539	4084	4	2376K	2864K	10:49:03			52	22	2

Profiler Connected: hostname:6104

The Performance Server gathers SPD Server process performance information and distributes it across the SPD Management section of SAS Management Console. This information consists of memory and resource allocations that are attributable to users and SPD Server processes that are spawned by an SPD Server name server. All SPD Server users must connect to an SPD Server name server before their SPD Server session can be spawned. Each SPD Server name server owns a dynamic family of subordinate SPD Server processes. SPD Server users and jobs create and terminate these processes.

To access a server's process performance information, the Performance Server application spdsperf must be running for the targeted SPD Server name server. SAS Management Console must be able to connect to the listening port of the SPD Server Performance Server.

You must first connect the Process Profiler to the Performance Server. Click **Connect** in the Process Profiler panel to open the Connect to SPD Profile Server dialog box.



Enter your host name or server name, and the Performance Server's listening port, and then click **Connect**. (You specify the Performance Server's listening port number when you start the Performance Server application.)

After SAS Management Console is connected to the Performance Server, the information that the Performance Server captures is displayed.

Note: Some host systems provide varying amounts of available resource information. Performance and resource information can vary from host to host.

The host performance profile information is automatically updated whenever the Performance Server performs another capture. You specify the frequency with which the Performance Server captures information using the **-c** option when you start the Performance Server. For more information about Performance Server start-up settings, see Chapter 22, "Setting Up the Performance Server," on page 201.

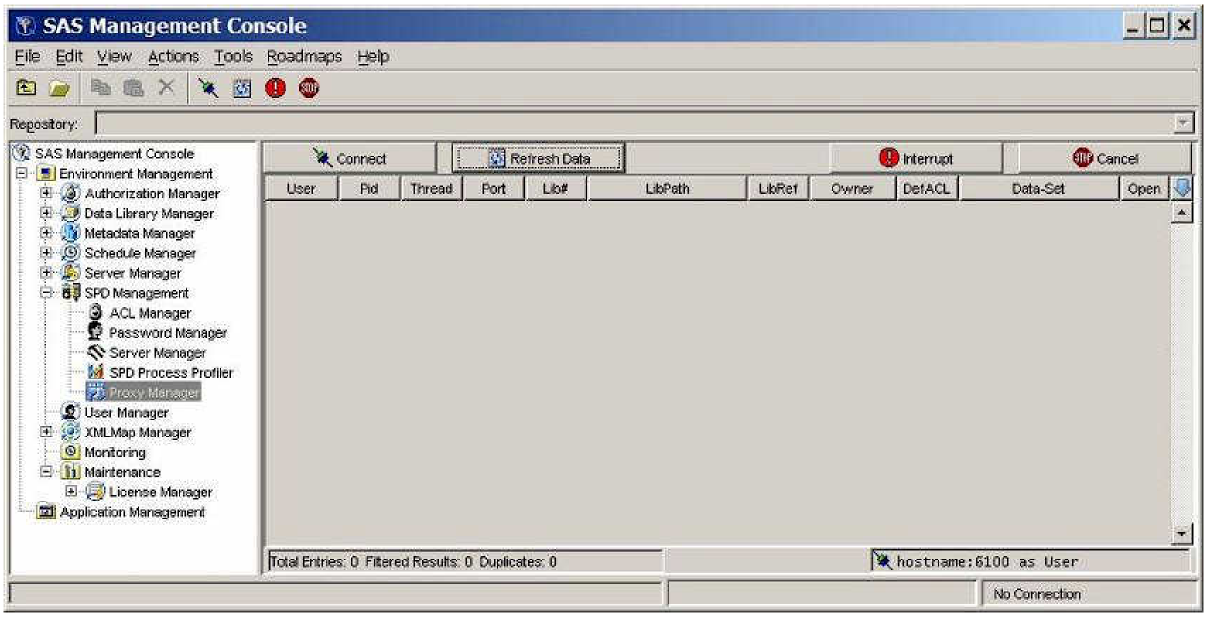
If the SAS Management Console user also connects to the same SPD Server name server through ACL Manager, Password Manager, or Server Manager, the user information that is displayed is the SPD Server user name that was used to connect with the LIBNAME statement. Otherwise, the user name of the user that started SPD Server and the component SPD Server processes is displayed.

Proxy Manager

Overview of Proxy Manager

Click the **Proxy Manager** folder in the SAS Management Console window to display tables that list all users that access a specific SPD Server host. The current libref allocations are displayed for each user, as well as information about the proxy that serves each libref. Available proxy information includes the process ID, the port number, the library path, and, if the libref was established with record locking, the thread ID. For each allocated libref, you can view every data set that is accessible to or open in the libref. If a libref was established with ACLSPECIAL=YES, then all data sets in the specified domain are visible and accessible to that libref, regardless of any connection settings that are established through the SPD Management utilities in SAS Management Console.

Before you can perform any operations in the Proxy Manager, you must be connected to an SPD Server host. For more information about connecting to a host, see "Connect to an SPD Server" on page 120. After you connect to an SPD Server host, click **Refresh Data** in the Proxy Manager to update the table data.



Total Entries: 0 Filtered Results: 0 Duplicates: 0

hostname:6100 as User

No Connection

You can filter, sort, reorder, and hide Proxy Manager table columns to display proxies of interest. Click on the column headings and select the appropriate choice from the menu.

You can apply filters to any number of columns. The following filter options are available:

- Show all values in the column.
- Show only the highlighted value or values in the column.
- Exclude the highlighted value or values in the column and show all others.

You can drag column headings to a new place in the table to reorder columns. To hide or reveal a column, use the blue down arrow that is above the vertical scroll bar on the right side of the Proxy Manager.

Refresh Proxies

Click **Refresh Data** to show the most current proxy data. You must refresh the data after an initial connection or after a proxy state has been manipulated. Because a proxy's state is dynamic, each refresh provides only a current snapshot of the proxies. The status of the data might change immediately after you refresh the data. If no users are currently logged on to the server, a window displays a message to that effect.

Interrupt Proxies

Click **Interrupt** to interrupt the proxies for a selected libref'. The proxy's activity is halted the next time it attempts to process data from its socket. The frequency with which the proxy accesses its socket is unpredictable and can vary depending on many variables. However, the proxy interrupt operation is typically the first method that you use to halt a proxy from accessing a given data set or domain.

Cancel Proxies

Click **Cancel** to cancel all proxies in the selected libref. The proxy's activity is immediately halted, and any open data connections are immediately closed. Use this method to stop a proxy from accessing a data set or a given domain if a previous interrupt operation is unacceptably delayed.

Part 4

Security

Chapter 13
Security Overview *139*

Chapter 14
ACL Security *141*

Chapter 15
Defining WHERE Constraints on Tables *169*

Chapter 16
Symbolic Substitution *173*

Chapter 17
DICTIONARY.PWDB and DICTIONARY.ACLS *177*

Chapter 18
Audit File Facility *181*

Chapter 19
Securing SPD Server Against Unknown Client Attacks *185*

Chapter 13
Security Overview

Security Overview 139
UNIX File-Level Protection 139
User IDs and Passwords 139
Special Privilege 139
Server Domains 140
ACL Security 140
Row-Level Security 140
Auditing 140

Security Overview

SPD Server uses host permissions, SPD Server user IDs, domains, access control lists (ACLs), and table WHERE constraints to secure resources. SPD Server also provides auditing.

UNIX File-Level Protection

Each session of SPD Server is attached to a user with a UNIX or Windows user ID. If SPD Server runs on UNIX, all files that the software creates are protected according to the permissions for creating UNIX files that are associated with that UNIX user's ID. SPD Server can read and write only files that have the appropriate file and directory access permissions to the SPD Server user's ID. Use the UNIX UMASK command to restrict the permissions for creating files.

User IDs and Passwords

Users must be registered in the password database in order to access SPD Server. Administrators register users in the password database by using the psmgr utility.

Special Privilege

When registering users with the psmgr utility, administrators can define authorization levels for users. Users can be assigned an authorization level from 0 to 7. The numbers 0–3 are equivalent: they specify a regular, non-privileged user. The numbers 4–7 are equivalent: they specify special privilege.

Users with special privilege can do the following things that regular users cannot do:

- update the password database
- modify ACLs to which they have not otherwise been granted access
- access other user’s resources.

All users connect to SPD Server as regular users, regardless of their authorization level. Users with special privilege must specify the ACLSPECIAL=YES LIBNAME option to invoke their special access in the SAS session. For more information about the psgmr utility, see Chapter 25, “Password Database Utility,” on page 213.

Server Domains

Users connect to SPD Server by specifying a server domain instead of by specifying the physical location of the server tables. A server domain is a logical name that is mapped to that location. Administrators define the valid server domains in the libnames.parm parameter file on each SPD Server. In effect, server domains isolate users from the file system.

ACL Security

SPD Server provides optional additional security on regular users by supporting creation of Access Control Lists (ACLs) on SPD Server resources. ACLs can be defined on all SPD Server resources, including domains, tables, table columns, catalogs, catalog entries, and utility files. By default, only the owner (creator) of a resource has access to it, unless the owner defines ACLs that grant other users access. Resource owners can grant ACL permissions to specific users, to specific groups of users (called an ACL group), and to all SPD Server users or all groups (universal permissions). For more information, see Chapter 14, “ACL Security,” on page 141.

Row-Level Security

Table owners can associate a WHERE clause with their tables so that when users access a table, they can see only the table rows that remain after the WHERE clause filter has been processed. The filtering is applied any time the table is accessed by a DATA step or by an SQL query. When table WHERE constraints are used with symbolic substitutions, table owners can create row-level security that filters table rows based on the values for SPD Server user ID, group name, or special privilege. For more information about table WHERE constraints, see Chapter 15, “Defining WHERE Constraints on Tables,” on page 169.

Auditing

SPD Server supports SQL audit logging of submitted SQL queries and proxy auditing of access to SPD Server resources. The auditing functionality is optional. For more information, see Chapter 18, “Audit File Facility,” on page 181.

Chapter 14
ACL Security

How ACLs Work to Control Access to SPD Server Resources **141**
Overview 141
Precedence of Permission Checks 142
Evaluation Process 143

Creating and Managing ACLs **144**

ACL Security Model **144**
User Ownership of Domains and Resources 144
ACL Permissions 144
ACL Groups 145
Granting Permissions with ACLs 145
ACL Types 145
Giving ACL Control to Others 147

ACL Examples **148**
Overview of Examples 148
Domain Security Examples 149
LIBACLINHERIT Example 151
Anonymous User Account Example 152
Domain Security and Administration Example 154
Domain Security and Group Access Example 158
Bring a Table Offline to Refresh 161
Bring a Domain Offline in Order to Refresh Tables 163
Special User Example 164
Column-Level Security Example 165

How ACLs Work to Control Access to SPD Server Resources

Overview

Each server user has a unique user ID and can be a member of one or more ACL groups. The user must specify his or her user name in the SPD Server LIBNAME statement in order to connect to the server. The group name is optional and, if not specified, will default to the first group that is defined for the user in the password database.

When a user creates a resource, their user ID and active group membership are saved in the resource metadata, and are later used to authorize access to the resource. Initially, only the owner of the resource is authorized to access the resource, with two exceptions:

- Resources created by the Anonymous user are accessible to all server users. For more information about the Anonymous user, see "Anonymous User ID" on page 57.
- Users who have special privilege in the password database and who specify the ACLSPECIAL= option in the LIBNAME statement can access the resource. For more information, see "Special Privilege" on page 139.

When opening a resource, the server compares the user ID from the SPD Server LIBNAME statement against the user ID that is stored with the resource. If the user IDs are the same, the requesting user is granted access to the resource. If they are not the same, the requesting user can access the resource only if the owner has created an ACL for the resource, and the ACL grants him or her access.

A resource ACL can define access permissions to specific users and groups. It can also grant universal access to all members of the owner's group and universal access to all users. The server uses precedence levels to evaluate eligibility for access. The precedence check stops when a match is found for the requesting user's credentials, or the precedence check completes without finding a match, which indicates the user was not intended to have access.

Precedence of Permission Checks

The precedence of the permission checks for resources in a domain can differ depending on whether administrators set the LIBACLINHERIT=YES option with the OWNER= specification in the domain definition.

When a user attempts to access resources in a domain that does not have LIBACLINHERIT=YES, the following ACL precedence of permissions checks are made on the resource:

1. If user-specific permissions are defined for the user in the ACL, the user gets these permissions.
2. In the absence of user-specific permissions in the ACL: If group-specific permissions are defined in the ACL for the user's group, the user gets the permissions defined for his or her group.
3. In the absence of user- and group-specific permissions in the ACL: The user is evaluated based on whether any universal group permissions are defined on the ACL. The universal group permissions apply only to users who are members in the group of the user who created the resource. If the user is a member of the owner's group, he or she gets these permissions.
4. If none of the above apply: the user gets any universal user permissions defined on the ACL.

When a user attempts to access resources in a domain that has LIBACLINHERIT=YES, the following ACL precedence of permissions checks are made on the resource:

1. If user-specific permissions are defined for the user in the ACL, the user gets these permissions.
2. In the absence of user-specific permissions in the ACL: If group-specific permissions are defined in the ACL for the user's group, the user gets the permissions defined for his or her group.
3. In the absence of user- and group-specific permissions in the resource ACL: If permissions are defined for the user in the domain ACL and the resource belongs to

the OWNER= of the domain, then the user gets the user permissions from the domain ACL on the resource.

4. In the absence of user- and group-specific permissions in the resource ACL and user permissions in the domain ACL: If permissions are defined for the user's group in the domain ACL and the resource belongs to the OWNER= of the domain, then the user gets the group permissions from the domain ACL on the resource.
5. In the absence of user- and group-specific permissions in the resource ACL and user- and group-specific permissions in the domain ACL: The user is evaluated based on any universal group permissions defined on the resource. If the user is a member in the group of the user who created the resource, then he or she gets these permissions.
6. If none of the above apply: The user gets any universal user permissions defined on the ACL.

In both scenarios, a specific user or group permission assignment in the ACL overrides any universal permissions that are defined.

Evaluation Process

Consider a user who is trying to open a resource that he or she does not own, and the owner created an ACL on the resource. The server will use the resource ACL and follow the precedence checks to determine whether the user has access. The user precedence check is TRUE if the requesting user ID matches a user record in the ACL. The user access is allowed or denied based on the permissions defined in that user record. If there is no record that matches the requesting user ID in the resource ACL, then the server proceeds to the group check.

The group precedence check is TRUE if the requesting user's group matches a group record in the ACL. The requesting user is allowed or denied access to the resource based on the permissions defined in that group record. If neither the user precedence check nor the group precedence check is TRUE, then universal permissions are checked to determine access.

When a TRUE precedence is found in the ACL, the requesting user is given or denied access to the resource based on the access being requested and the permissions defined for the TRUE precedence level. For example, if the user is requesting Read access to the resource, and the owner created an ACL on the resource for the user that grants Read permission only, then Read access is allowed. However, if the same user requests Write access to the resource, access is denied. When a precedence check for an ACL is FALSE, indicating that no ACL record exists for the precedence level check, then the evaluation proceeds to the next precedence level until a precedence level evaluates to TRUE, or all levels are checked.

The ability to define permissions that deny access to specific users is useful when you want to allow access to a resource to all members in a group except for one particular member. This can be done by creating an ACL on the resource that grants the group Read permission to the resource. Then, modifying the ACL to deny Read permission to the particular member. Because the user precedence check is performed before the group precedence check, the denied member will be TRUE and he or she will be denied access. The other group members will not match the user ACL precedence, checking FALSE, and will proceed to the group precedence check, which will be TRUE. They will have group Read permission to the resource. UNIX File System permissions cannot prohibit a particular user from having group access to a file.

The key thing to remember is that the server uses the first TRUE precedence record to determine access to the resource. Universal permissions do not necessarily grant access to everyone. If a user is denied access before the evaluation proceeds to the universal access level, the user will not be granted the universal access.

Creating and Managing ACLs

Resource owners can use the SPDO procedure or SAS Management Console to create and manage ACLs. An advantage of using PROC SPDO is that the programs used to create the ACLs can be retained and used to re-create the ACLs if the ACLs are somehow lost or damaged. For more information about PROC SPDO, see Chapter 26, "SPDO Procedure," on page 227. However, before creating ACLs, you should understand the security model.

ACL Security Model

User Ownership of Domains and Resources

Resource ownership begins with the domain definition in the libnames.parm parameter file. Administrators can attach an OWNER= specification to any defined domain. Only the system user whose SPD Server user ID matches the OWNER= specification (and users who have special privilege) can create tables in this domain. The owner can grant other server users Read and Write permission to the domain by defining a domain ACL. If a domain definition does not contain an OWNER= specification, then any SPD Server user can create resources in the domain.

Each resource created is also tagged with the user ID and group of the user who created it. This user is referred to as the resource owner. Only the resource owner (or users with special privilege) can access a table, unless the owner grants others access through ACLs.

ACL Permissions

The following permissions can be specified in an ACL to grant universal permission to all users:

READ
: grants universal Read access to the resource (read or query).

WRITE
: grants universal Write access to the resource (append to or update an existing resource).

ALTER
: grants universal Alter access to the resource (rename, delete, or replace a resource, and add or delete indexes associated with a table).

The following properties can be specified in an ACL to grant permissions to users in the owner's group:

GROUPREAD
: grants group Read access to the resource (read or query)

GROUPWRITE
: grants group Write access to the resource (append to or update)

GROUPALTER
: grants group Alter access to the resource (rename, delete, or replace a resource, and add or delete indexes associated with a table).

To grant specific users and groups permissions to a resource, resource owners can define user lists and group lists in the ACL. A user list defines either a grant (Y) or denial (N) for each of the following properties for a given user ID. The group list defines either a grant (Y) or denial (N) for each of the following properties for a given group name.

READ
: grants the named user or group Read access to the resource (read or query)

WRITE
: grants the named user or group Write access to the resource (append to or update)

ALTER
: grants the named user or group Alter access to the resource (add, rename, delete, or replace a resource, and add or delete indexes associated with a table)

CONTROL
: grants the named user or group Control access to the resource (ability to modify the resource's ACL).

ACLs are created in PROC SPDO. Use the ADD ACL statement to create an ACL. User lists and groups lists can be added to an existing ACL using the ACL MODIFY statement.

ACL Groups

SPD Server ACL groups are similar to UNIX ACL groups. Each SPD Server user can belong to one or more groups.

SPD Server administrators can associate a given SPD Server user ID with up to 32 ACL groups, depending on how SPD Server is configured. By default, a user connects to the server with the first group defined for the user in the password database. Users can connect to SPD Server with a different group from their group list by specifying the ACLGRP= option in the SPD Server LIBNAME statement. The ACLGRP= value must match one of the groups that are defined for that user.

Granting Permissions with ACLs

An ACL permits three distinct levels of permission on a resource.

1. First, you can grant universal permissions to users who are not in the same ACL group as the resource owner.
2. Second, you can grant Group permissions to users who are in the same ACL group as the resource owner.
3. Third, you can grant User permissions to a specific user ID or Group permissions to a specific group.

ACL Types

Domain ACLs

A user whose user name is specified as OWNER= on a domain can control access to his or her domain by creating a domain ACL. A domain ACL is created by specifying the /LIBNAME option when you add an ACL. The permissions defined in the domain

ACL are applied whenever a server user attempts to connect to the domain. The owner must grant a user either Read or Write permission in the domain's /LIBNAME ACL for the connection to succeed. The Read permission gives a user Read and Write permission to existing resources in the domain, based on the resource ACLs, but does not allow the user to create new resources in the domain. Write permission in the /LIBNAME ACL is necessary to allow a user to create new resources in the domain.

Domain ACLs for LIBACLINHERIT=YES Domains

When creating an ACL for a domain that specifies LIBACLINHERIT=YES, SAS recommends granting Read permission to other users only. Write permission to a LIBACLINHERIT=YES domain would allow other users to update the owner's tables in addition to creating tables in the domain. Alter access would allow other users to add and remove columns and indexes in the owner's tables, in addition to creating tables in the domain. To protect the integrity of the owner's tables, neither Write nor Alter permission should be defined in an ACL for a LIBACLINHERIT=YES domain.

Resource ACLs

Users who have been granted the ability to create resources in a domain will need to create ACLs for their resources to allow other users and groups to access the resources. In PROC SPDO, the resource type for all resources except the domain ACL is indicated by using the SET ACLTYPE statement. The SET ACLTYPE statement is specified before the ADD ACL statement and other statements that act on an ACL in PROC SPDO. If you omit the SET ACLTYPE statement from an ACL request, the server will assume that the resource type is DATA.

All resources except table columns are identified in an ACL entry by using a one-part name. When a two-part name is used, it is assumed that the first part is a table name and the second part is a column name.

Generic ACLs

SPD Server supports generic ACLs to enable resource owners to define permissions for a class of resources that have a common prefix. A generic ACL is created by specifying the /GENERIC option when you add the ACL entry. Marking an ACL as generic causes the ACL to function as if an asterisk (*) wildcard was appended to the end of the ACL entry's name. For example, if you have tables named Salesne, Salesse, Salesmw, Salessw, Salespw, and Salesnw, you could make a generic ACL entry named Sales to secure all of the tables. This wildcard behavior enables you to make a single ACL entry cover many resources instead of making separate entries for each resource.

Note: If you specify /GENERIC when you define a table-column ACL, the /GENERIC option applies to the table name, not to the column name. You cannot use wildcard characters with column names.

If you create a generic ACL for a resource that has a full name ACL, the full name ACL takes precedence. If you create multiple generic ACLs that cover the same table name, the longer generic ACL takes precedence. For example, if you have table MYTABLE and created ACL MYTABLE and generic ACL MYT*, the MYTABLE ACL takes precedence for permission checks on the table. If you create table MYTABLE, generic ACL MYT*, and generic ACL M*, the generic MYT* takes precedence for permission checks on the table.

Column ACLs

Table owners can control access to table contents at the column level by defining column ACLs. Column ACLs can be defined for individual users at the user level, or for collections of users at the group level. SPD Server enforces precedence for user and group ACL permissions: First, user ACL permissions are applied, and then group ACL restrictions are applied. User permissions override group permissions.

When you use an ACL statement to create a protected column in a table, all individual users or groups are automatically denied access to the protected column until you explicitly grant them ACL permission to access it. When you issue an ACL statement to grant or deny the contents of a table column to a single user or to a user group, the protected column automatically becomes unavailable to *all* other users and user groups, unless you specifically give them access to the protected column.

Consider a scenario in which a testing department hires a new member, Joe. Joe has applied for classified security clearance, but his or her security clearance level will not be certified for several weeks. All members of the department use a table called Testing that contains a column of classified information. Joe needs access to all of the Testing table except the protected column, and the rest of his or her group needs access to the whole Testing table. Here are steps to give Joe and the other members of the department the correct permissions:

1. Submit a user-level ACL statement to restrict the secure column in table Testing from Joe.

   Joe is explicitly denied access, but because the column is now a protected entity; all other users who access the Testing table are also denied access to the column by default.

2. Issue a group-level ACL column permission to the user group Testgroup. Explicitly grant group Testgroup access to the protected column.

   After a column is protected with ACL security, you must grant explicit permissions in order for any user (or group of users) to be able to access the column content.

3. SPD Server reads the user-level ACL permissions first, and gives Joe access to the table Testing, but restricts him or her from the secure column.

4. SPD Server reads the group ACL permissions and grants all of the Testgroup members access to the full table, including the secure column.

Now consider another scenario, in which John manages a group Devgroup whose members record their billable project hours and codes in a table. In that table, manager John keeps billing-rate information based on employee salaries in a protected column Rate. Only John should be able to see the entire table, and the rest of the Devgroup should be able to see the table minus the Rate column. In this case, you create column security by protecting the Rate column with a user-level ACL permission statement for John. The Devgroup members can have full table permissions at the group level, but cannot see the protected column because John's user-level column security ACL overrides any group-level ACLs for the Devgroup table.For example code that implements column-level security, see “Column-Level Security Example” on page 165.

Persistent ACLs

A persistent ACL is an ACL that is not removed from a resource when the resource is deleted. When you are using PROC SPDO, use the /PERSIST option to identify a persistent ACL.

Giving ACL Control to Others

A resource owner can permit other users to alter his or her ACL by granting a specific user or group Control access to that ACL. In PROC SPDO, this is done with the MODIFY ACL statement.

ACL Examples

Overview of Examples

This section contains examples of PROC SPDO requests that create and modify ACLs to perform common security tasks. They show the following:

- how to secure a domain with a domain ACL, as well as how authorized users can modify the domain ACL
- the difference in permission precedence when LIBACLINHERIT is set versus when it is not
- how the user account can be used
- how generic ACLs might be used
- how group assignments might be made
- how to bringa table or a whole domain offline in order to refresh tables
- how to define and alter column security.

The examples assume that the following domains are defined in the libnames.parm parameter file:

```
libnames.parm:
---------------------------------
LIBNAME=d1
  pathname=/IDX1/spdsmgr/d1
  owner=admin ;
LIBNAME=d2
  pathname=/IDX1/spdsmgr/d2
  owner=prod1 ;
LIBNAME=colsec
  pathname=/IDX1/spdsmgr/colsec
  owner=boss ;
LIBNAME=onepath
  pathname=/IDX1/spdsmgr/onepath ;
```

The examples assume that the following users exist in the password database:

```
Password Database List:

User     Level  Entry Type  Group
----------------------------------
ADMINGRP   0   GROUP ENTRY
GROUP1     0   GROUP ENTRY
GROUP2     0   GROUP ENTRY
GROUP3     0   GROUP ENTRY
GROUP4     0   GROUP ENTRY
PRODGRP    0   GROUP ENTRY
ADMIN1     7   user ID     ADMINGRP
ADMIN2     7   user ID     ADMINGRP
PROD1      7   user ID     PRODGRP
PROD2      7   user ID     PRODGRP
USER1      0   user ID     GROUP1
```

```
USER2      0   user ID     GROUP2
USER3      0   user ID     GROUP3
USER4      0   user ID     GROUP4
USER5      0   user ID     GROUP1
USER6      0   user ID     GROUP2
USER7      0   user ID     GROUP3
USER8      0   user ID     GROUP4
BOSS       7   user ID     ADMINGRP
EMPLOYEE   0   user ID
```

Domain Security Examples

Create a Domain ACL

When OWNER= is set on a domain in the libnames.parm parameter file, no other user besides the one specified as OWNER= can access the domain unless the user is given permissions by the owner. The domain owner can grant permissions to access a domain by defining a domain ACL. The following code creates a domain ACL for domain d2 to grant access permissions to different groups.

```
/* Libref d2 is assigned to connect to     */
/* domain d2 as domain owner (prod1).      */

libname d2 sasspds 'd2'
  server=zztop.5162
  user='prod1'
  password='spds123'
  IP=YES ;

/* PROC SPDO connects to libref d2. */

PROC SPDO library=d2 ;

/* The session context is set to the domain owner.  */

  set acluser prod1 ;

/* The ADD ACL statement creates the domain ACL */

  add ACL / LIBNAME ;

/* The MODIFY ACL statement modifies the domain ACL */
/* to grant group permissions to domain d2. ProdGrp  */
/* is granted full access to the domain, including */
/* ACL access.                                      */
  modify ACL /
  LIBNAME prodgrp=(y,y,y,y)
  group1=(y,y,n,n)
  group2=(y,n,n,n)
  group3=(y,n,n,n) ;

/* Specific users are given access to the domain. */
  modify ACL /
  LIBNAME user7=(y,n,n,n)
  admin1=(y,n,n,n) ;
  list ACL _all_ ;
quit ;
```

Connect to the Domain ACL as an Authorized User

The user Prod2 is in ProdGrp, the group that has permissions to control the domain ACL. Any user in that group can modify the domain ACL.

However, because the ACL was created by user Prod1, the user Prod2 must use the user ID Prod1 to modify the domain ACL. This is allowed because the group was given control. User Prod1 still remains the owner of the domain ACL.

```
/* Libref prod2d2 is created to connect to domain d2 */
/*  as user prod2 */
  libname prod2d2 sasspds 'd2'
  server=zztop.5162
  user='prod2'
  password='spds123'
  IP=YES ;

PROC SPDO library=prod2d2 ;

/* The ACLUSER statement sets the session */
/* context to 'prod1', who owns the */
/* ACL to be modified */

  set acluser prod1 ;
/* The domain ACL is modified to deny Group1 */
/* access to the domain and to give Group 4 */
/* Read-only access. */
  modify ACL /
  LIBNAME group1=(n,n,n,n)
    group4=(y,n,n,n) ;
  list ACL _all_ ;
quit ;
```

Modify the Domain ACL with Special Privilege

A user who has been given special privilege in the password database can also change domain ACLs. In the following example, the user Admin1 modifies the domain ACL. The user activates his or her privilege to operate under the user ID of another user by specifying the ACLSPECIAL= option in the LIBNAME statement. As in the previous example, the user Admin1 must set the session context to the user ID Prod1 to modify the ACL.

```
/* Libref admin1d2 is created to enable user */
/* Admin1 to connect to domain d2.   */
  libname admin1d2 sasspds 'd2'
  server=zztop.5162
  user='admin1'
  password='spds123'
  ACLSPECIAL=YES
  IP=YES ;
```

```
PROC SPDO library=admin1d2 ;
  set acluser prod1 ;

/* The MODIFY ACL statement specifies a grouplist */
/* that gives AdminGrp Read-only access to the domain */
  modify ACL /
  LIBNAME admingrp=(y,n,n,n) ;
    list ACL _all_ ;
quit ;
```

LIBACLINHERIT Example

If the LIBACLINHERIT domain option is turned on for a domain in the libnames.parm parameter file, the ACL precedence of permission checks changes for resources that belong to the domain owner. Specifying LIBACLINHERIT=YES in a domain definition extends the permissions that are defined in the domain ACL to the owner's resources. For detailed information about LIBACLINHERIT, see Chapter 8, "Configuring Server Domains," on page 59.

The following example defines two domains: one that has LIBACLINHERIT=YES defined and one that does not. Then, it creates a table in each domain, and creates domain ACLs on each domain that enable user Anonymous to print the tables. Implement the code at your site to see the difference in how permissions are applied.

Here are statements for the libnames.parm parameter file:

```
LIBNAME=LIBINHER
    pathname=/IDX1/spdsmgr/spds41test/libinher
    LIBACLINHERIT=YES
    owner=admin;
 LIBNAME=noinher
    pathname=/IDX1/spdsmgr/spds41test/noinher
    owner=admin;
```

Here is code that creates the tables and ACLs and prints the tables:

```
/* Connect to libinher as admin. *
ibname libinher sasspds 'libinher'
   server=zztop.5129
   user='admin'
   password='spds123';

/* Connect to noinher as admin. */
libname noinher sasspds 'noinher'
   server=zztop.5129
   user='admin'
   password='spds123';

/* Create tables. */
data libinher.admins_table
     noinher.admins_table ;

   do i = 1 to 10;
     output;
   end;
run;


/* Set up access for user anonymous.    */

/* Create domain ACL for domain libinher */
PROC SPDO library=libinher;

set acluser admin;
add acl / LIBNAME;

/* Allow users read-only */
/* access to the domain.             */
modify acl / LIBNAME read;
list acl _all_;
quit;


/* Create domain ACL for domain noinher. */
PROC SPDO library=noinher;

set acluser admin ;
add acl / LIBNAME ;

/* Allow users read-only  */
/* access to the domain.    */

modify acl / LIBNAME read ;
list acl _all_;
quit;

/* Connect to domains as user anonymous. */
libname a_inher sasspds 'libinher'
   server=zztop.5129
   user='anonymous';
libname a_noher sasspds 'noinher'
   server=zztop.5129
   user='anonymous';

/* Print the tables. */
proc print data=a_inher.admins_table;
   title 'with libaclinher';
run;

proc print data=a_noher.admins_table;
   title 'without libaclinher';
run;
```

Anonymous User Account Example

SPD Server uses a general ID that is called Anonymous. Any person who can connect to the server can do so using the Anonymous user ID. Any table that is created by the Anonymous user can be viewed by all users who have access to that table's domain. The Anonymous user can place ACLs on a table to limit access.

```
/* John logs in using the anonymous */
/* user ID and creates a table.      */

     libname john sasspds 'onepath'
       server=zztop.5162
       user='anonymous'
       password='anonymous'
       IP=YES ;

     data john.anonymous_table ;
       do i = 1 to 100 ;
       output ;
       end ;
     run ;

/* Mary can also log in as anonymous  */
/* and read the table that John       */
/* created.                           */

     libname mary sasspds 'onepath'
       server=zztop.5162
       user='anonymous'
       IP=YES ;

     proc print data=mary.anonymous_table
       (obs=10) ;
       title
         'mary reading anonymous_table' ;
     run ;

/* User1 can log in and read the table */
/* that John created.                  */

     libname user1 sasspds 'onepath'
       server=zztop.5162
       user='user1'
       password='spds123'
       IP=YES ;

     proc print data=user1.anonymous_table
       (obs=10) ;
       title
         'user1 reading anonymous_table' ;
     run ;

/* Tables created by the anonymous user   */
/* can have ACLs.                      */

     PROC SPDO library=john ;

/* assign who owns the ACL */

     set acluser anonymous ;

/* The MODIFY statement sets an ACL so */
/* only user 'anonymous' can read   */
/* the table.                          */

     add ACL anonymous_table ;
     modify ACL anonymous_table /
       anonymous=(y,n,n,n);

     list ACL _all_;
     quit ;

/* Now, only user 'anonymous' can */
/* read the table.                     */

     libname user1 sasspds 'onepath'
       server=zztop.5162
       user='user1'
       password='spds123'
       IP=YES ;

     proc print data=user1.anonymous_table
       (obs=10) ;
       title
         'user1 trying to read anonymous_table' ;
     run ;

     proc print mary sasspds 'onepath'
       server=zztop.5162
       user='anonymous'
       password='anonymous'
       IP=YES ;

     proc print data=mary.anonymous_table
       (obs=10) ;
       title
         'mary reading anonymous_table' ;
     run ;

/* Mary can't write to anonymous_table. */

     data mary.anonymous_table ;
       do i = 1 to 100 ;
       output ;
       end ;
     run ;
```

Domain Security and Administration Example

A common security measure assigns an SPD Server user ID to act as the owner of a domain and to provide control over it. Typically, one or two user IDs are defined to administer table loads and refreshes. These user IDs can perform all the jobs that are required to create, load, refresh, update, and administer SPD Server security. Using one or two user IDs centralizes the data administration on the server. More than one user ID for data administration spreads responsibility and provides backup. The following example demonstrates how to grant different groups access to the domain and tables, and how different groups can control resources in the domain.

```
libname d1 sasspds 'd1'
        server=zztop.5162
        user='admin1'
        password='spds123'
        IP=YES ;

      PROC SPDO library=d1 ;

/* Assign who owns the ACLs. */

      set acluser admin1 ;

/* Add a domain ACL to d1.  */

      add ACL / LIBNAME ;
```

The MODIFY statement in the following code enables the following actions:

- Any user who is in the same group as Admin can read, write, or alter tables and can modify the ACL access to the domain.
- Users in Group1 and Group2 are granted Read access to the domain.
- Users in Group3 and Group4 are granted Read and Write access to the domain.

```
      modify ACL / LIBNAME
        admingrp=(y,y,y,y)
        group1=(y,n,n,n)
        group2=(y,n,n,n)
        group3=(y,y,n,n)
        group4=(y,y,n,n) ;

      list ACL _all_;
      quit ;

/* Create two tables. */

      data d1.admin1_table1 ;
        do i = 1 to 100 ;
        output ;
        end ;
      run ;

/* Admin1 has write priviliges to */
/* the domain.                    */

      data d1.admin1_table2 ;
        do i = 1 to 100 ;
        output ;
        end ;
      run ;


/* Generic ACLs allow all users to */
/* read tables created by admin1   */
/* unless a specific ACL is placed */
/* on a resource.                  */
```

```
      PROC SPDO library=d1 ;

/* Assign who owns the ACLs. */

      set acluser admin1 ;
```

The two ACL commands in the following code give Read privileges to members of the ACL group AdminGrp for any table that is created by the user Admin1. The user Admin1 has Read access to the domain.

This ACL is a good example for data marts and warehouses that do not contain sensitive data. A generic ACL gives broad access to tables in a domain. You must use generic ACLs correctly (or not at all) if you need to restrict access to sensitive data to specific users or groups of users. If a table in a domain with generic ACLs is not specifically protected by its own ACL, there is a risk of allowing access by any user to sensitive data.

```
     add ACL / generic
       read ;
     modify ACL / generic read
       admingrp=(y,n,n,y) ;
     list ACL _all_;
     quit ;

/* Test access for a user in group1. */

    libname user1d1 sasspds 'd1'
      server=zztop.5162
      user='user1'
      password='spds123'
      IP=YES ;

    proc print data=user1d1.admin1_table1
      (obs=10) ;
      title
        'read admin1_table1 by user1' ;
    run ;

    proc print data=user1d1.admin1_table2
      (obs=10) ;
      title
        'read admin1_table2 by user1' ;
    run ;

/* Test access for a user in group2. */

    libname user2d1 sasspds 'd1'
      server=zztop.5162
      user='user2'
      password='spds123'
      IP=YES ;

    proc print data=user2d1.admin1_table1
      (obs=10) ;
      title
        'read admin1_table1 by user2' ;
    run ;
    proc print data=user2d1.admin1_table2
      (obs=10) ;
      title
        'read admin1_table2 by user2' ;
    run ;
```

When any ACL is placed on a specific table, that ACL takes precedence over the generic ACL. The ACL in the following code provides the following access:

- gives Group1 Read access to admin1_table2.
- gives AdminGrp Read and Control access to admin1_table2.
- prevents users who are not granted specific access to admin1_table2 from reading, writing, altering, or controlling the table. The ACL in the code takes precedence over the generic Read ACL.

```
    PROC SPDO library=d1 ;

/* Assign who owns the ACLs. */

    set acluser admin1 ;

/* This ACL takes precedence over the */
/* generic ACL for users that try to  */
/* access admin1_table2.              */

    add ACL admin1_table2 ;
    modify ACL admin1_table2 /
      group1=(y,n,n,n)
      admingrp=(y,n,n,y) ;
    list ACL _all_;
    quit ;

/* Test access for a user in group1. */

    libname user1d1 sasspds 'd1'
      server=zztop.5162
      user='user1'
      password='spds123'
      IP=YES ;

    proc print data=user1d1.admin1_table2
      (obs=10) ;
      title
        'read admin1_table2 by user1' ;
    run ;

/* Test access for a user in group2. */

    libname user2d1 sasspds 'd1'
      server=zztop.5162
      user='user2'
      password='spds123'
      IP=YES ;


    proc print data=user2d1.admin1_table2
```

```
          (obs=10) ;
          title
            'read admin1_table2 by user2' ;
        run ;
```

Domain Security and Group Access Example

This section of code provides an overview of server domain security and group access using PROC SPDO.

Permissions are often granted to a group of users rather than to individual users. This example shows how to grant different groups of users access to the domain owned by the user Admin, and then extends the access to the tables. Granting permissions in this way makes administration both simpler and more secure. Admin1 is the owner of the domain and can determine access to the resources. In the following example, PROC SPDO grants the following user access:

- Any user in AdminGrp is granted Read, Write, and Alter access to the domain.
- Any user in Group1 or Group2 is granted Read access to the domain.
- Any user in Group3 or Group4 is granted Read and Write access to the domain.

```
       libname d1 sasspds 'd1'
         server=zztop.5162
         user='admin'
         password='spds123
         IP=YES ;

       PROC SPDO library=d1 ;

 /* Assign who owns the ACLs. */

       set acluser admin ;

 /* Add a domain ACL to d1. */

       add ACL / LIBNAME ;

/* Allow any user in same group */
/* as admin to read, write, or  */
/* alter tables in the domain.   */

       modify ACL / LIBNAME
         admingrp=(y,y,y,n)
         group1=(y,n,n,n)
         group2=(y,n,n,n)
         group3=(y,y,n,n)
         group4=(y,y,n,n) ;
       list ACL _all_;
       run;

/* Admin1 has write privileges to */
/* the domain.                     */

       data d1.admin1_table1 ;
         do i = 1 to 100 ;
         output ;
         end ;
       run ;

/* Generic ACL allows all users to */
/* read tables created by admin1.   */

       PROC SPDO library=d1 ;

/* Assign who owns the ACLs. */

       set acluser admin1 ;

/* Modify domain ACL for groupread  */
/* and groupwrite.  The ACL MUST    */
/* include groupread to enable       */
/* other users  in the same group    */
/* as admin2 to be able to read      */
/* tables that were created by      */
/* admin2.                          */

       add ACL admin1_table1 /
         generic
         read
         groupread
         groupalter ;

       list ACL _all_;
       run;

/* a\Admin1 has write privileges to    */
/* the domain.                        */

       data d1.admin1_table2 ;
         do i = 1 to 100 ;
         output ;
         end ;
       run ;

/* Generic ACL allows all users to    */
/* read the tables.                    */

       PROC SPDO library=d1 ;

/* Assign who owns the ACLs. */

       set acluser admin1 ;

/* Add a table and modify domain ACL */
/* for groupread and groupwrite. The  */
/* ACL MUST include groupread to give */
/* users in the same group as admin2  */
/* the ability to read tables created */
/* by admin2.                          */

      add ACL admin1_table2 /
```

```
        group1=(y,n,n,n)
        admingrp=(y,n,n,y) ;
        list ACL _all_;
      run;

/* Admin2 has write privileges to the */
/* domain.                             */

      data admin2d1.admin2_table ;
        do i = 1 to 100 ;
        output ;
        end ;
      run ;

/* Admin2 must use PROC SPDO to allow */
/* users read access to the table.    */
/* The PROC SPDO example below uses   */
/* generic syntax with a read.  This  */
/* provides any user outside of the   */
/* admingrp read access to tables     */
/* that were created by admin2.  The */
/* groupread and groupalter allow     */
/* access by users within admingrp.   */

     PROC SPDO library=admin2d1 ;

/* Assign who owns the ACLs. */

     set acluser admin2 ;

/* Modify domain ACL for groupread   */
/* and groupwrite. The ACL MUST       */
/* include groupread if other users   */
/* in the same group as admin2 need   */
/* to read tables created by admin2.  */

     add ACL / generic
       read
       groupread
       groupalter ;

     list ACL _all_;

/* Admin (same group) can read the     */
/* table.                               */

     proc print data=d1.admin2_table
       (obs=10) ;
       title 'read by admin' ;
     run ;

/* Admin has been given the ability to */
/* modify or replace tables created by */
/* admin2 with 'groupalter'.           */

     data d1.admin2_table ;
```

```
        do i = 1 to 100 ;
        output ;
        end ;
      run ;

/* Provide other users in same group   */
/* read access to the table.           */

      PROC SPDO library=admin2d1 ;

/* Assign who owns the ACLs. */

      set acluser user3 ;

/* Modify domain ACL for groupread    */
/* and groupwrite. The ACL MUST        */
/* include groupread if other users in */
/* the same group as admin2 are to be  */
/* able to read tables that were       */
/* created by admin2.                   */

      add ACL user3_table /
        groupread ;
      list ACL _all_;
```

Bring a Table Offline to Refresh

The following scenario explains how to bring a table offline and refresh it:

1. Revoke Read access to all users, except the user that will perform the refresh.

   ```
       libname d2 sasspds 'd2'
         server=zztop.5162
         user='prod1'
         password='spds123'
         IP=YES ;
   ```

   This example assumes that the table prod1_table is already loaded in the domain and that the groups who use the table have access.

   ```
   PROC SPDO library=d2 ;

   /* Assign who owns these ACLs. */

   set acluser prod1 ;
   ```

2. Modify the table ACL:

   a. Revoke Read and Control access by user IDs that are in the same group. This step prevents locks during table refreshes.

   b. Revoke Read access by users that are in Group1 through Group4 to prevent locks during the refresh process.

      Note: If a user is actively accessing a data table when the ACLs for that table are modified, the user continues to have access. This situation can create a table lock that prevents the table refresh from occurring. By revoking the table's Read privileges before the refresh occurs, new SPD Server jobs cannot access the table.

Existing jobs continue to run and can finish under the lock.

TIP You can also use the PROC SPDO operator commands to identify any users that might be running unattended jobs, and disconnect them so that the refresh can take place.

```
modify ACL prod1_table /
  prodgrp=(n,n,n,n)
  group1=(n,n,n,n)
  group2=(n,n,n,n)
  group3=(n,n,n,n)
  group4=(n,n,n,n) ;
```

3. Modify table ACLs to allow user Prod1 to perform table refreshes. Because user Prod1 is part of the group ProdGrp, Prod1 loses access to the table when the permissions are changed. As the domain and table owner, Prod1 can still modify ACLs to gain access.

   ```
   modify ACL prod1_table /
   prod1=(y,y,y,y) ;
   list ACL _all_;
   quit;
   ```

   Now user Prod1 has full access to refresh the table.

   ```
   data d2.prod1_table ;
   do i = 1 to 100 ;
   output ;
   end ;
   run ;

   PROC SPDO library=d2 ;

   /* Specify who owns the ACLs */

   set acluser prod1 ;
   ```

4. After the table has been refreshed, modify the ACL to allow Read access again.

   ```
   modify ACL prod1_table /
   prodgrp=(y,n,n,y)
   group1=(y,n,n,n)
   group2=(y,n,n,n)
   group3=(y,n,n,n)
   group4=(y,n,n,n) ;
   list ACL _all_ ;
   run ;
   ```

You do not need to issue an ADD ACL command for prod1_table. When you delete or replace a table, you do not delete the ACLs. The ACL for that table remains until one of the following actions has occurred:

- The table ACL is deleted using PROC SPDO delete syntax.
- The table is deleted and another user creates a table with the same name.

If one of these actions occurs, the ACLs have not been deleted. Deleting the table releases any rights that owner has on the table. The exception is when persistent ACLs are used.

Bring a Domain Offline in Order to Refresh Tables

You can approach this type of table refresh in two ways.

- You can minimize contention and table locking by revoking privileges of users and groups who will not be involved in the refresh process.
- Alternatively, you can revoke Read access at the domain level, which allows the IDs that are used to refresh the warehouse to have complete control of resources in the domain. This example turns off all Read access to the domain, except for IDs that are in the production group (ProdGrp). This approach allows the production IDs to have full control over the tables and resources.

Note: Any user who is currently accessing the domain continues to have access until they are disconnected. This situation can cause a lock to occur. You can use the PROC SPDO operator commands to identify the user and disconnect the process so that the refresh can occur.

This example assumes that the tables are already loaded in the domain and that the groups that use them have access to the domain.

```
  libname d2 sasspds 'd2'
    server=zztop.5162
    user='prod1'
    password='spds123'
    IP=YES ;

  PROC SPDO library=d2 ;

/* Assign who owns the ACLs. */

   set acluser prod1 ;

modify ACL / LIBNAME
prodgrp=(y,y,y,y)
group1=(n,n,n,n)
group2=(n,n,n,n)
group3=(n,n,n,n)
group4=(n,n,n,n);
list ACL _all_ ;
run ;

/* Modify ACL for tables to be refreshed. */

PROC SPDO library=d2 ;

/* Set who owns the ACLs. */

set acluser prod1 ;

/* Modify table ACL to revoke read and */
/* control by users in same group, */
/* which prevents locks during table */
/* refreshes. */

modify ACL prod1_table /
prodgrp=(n,n,n,n);
```

```
/* Modify table ACL to allow the */
/* 'prod1' user to refresh the */
/* table. */

modify ACL prod1_table /
prod1=(y,y,y,y) ;
list ACL _all_;

/* Refresh warehouse table(s). */

data d2.prod1_table ;
do i = 1 to 100 ;
output ;
end ;
run ;

PROC SPDO library=d2 ;

/* Assign who owns the ACLs. */

set ACLUSER prod1 ;

/* Allow users and groups access to */
/* the domain again. */

modify ACL / LIBNAME
group1=(y,n,n,n)
group2=(y,n,n,n)
group3=(y,n,n,n)
group4=(y,n,n,n) ;

list ACL _all_ ;
run ;
```

Special User Example

SPD Server users are divided into two levels: 0 through 3 and 4 through 7. Level 4 through level 7 users can log on as an SPD Server “super user” who can do the following tasks:

- access any table
- change table ACLs
- disconnect users
- perform administrative functions when necessary

SPD Server super users can perform database administrator functions. SPD Server super users cannot change the ownership of a table, but they can assume the identity of the table owner to do required work. This type of situation occurs when a user needs access and the table owner or domain owner is out of the office. SPD Server super users can access any data in any domain, so this special privilege should be given to few users and should be granted with extreme care.

Assume that the table user1_table1 is loaded, and that users in Group1 have Read permissions to that table. User4 is a member of Group4, and Group4 does not have Read access to the table. User1 is the owner of table user1_table1 in domain d2. User1 is on vacation, and User4 has been given an assignment that requires Read access to the table user1_table1 to create a report for management.

Management has approved User4 for access to the table. The super user Prod1 uses the ACLSPECIAL= option to modify the ACLs and to give User4 Read access to the table.

```
     libname prod1d2 sasspds 'd2'
       server=zztop.5162
       user='prod1'
       password='spds123'
       aclspecial=YES
       IP=YES ;

     PROC SPDO library=prod1d2 ;

/* Assign to the user to who owns   */
/* the ACL that will be modified.    */

     set acluser user1 ;

/* Give user 'user4' read access */
/* to user1_table1.                  */

     modify ACL user1_table1 /
       user4=(y,n,n,n) ;
     list ACL _all_ ;
     quit;
```

Column-Level Security Example

The goal of column-level security is to allow only privileged users to access sensitive columns of tables that other users are not permitted to access.

```
libname user1 sasspds 'onepath' server=zztop.5161 user='user1'
       password='spds123';
libname user2 sasspds 'onepath' server=zztop.5161 user='user2'
       password='spds123' aclgrp='group2';
libname user6 sasspds 'onepath' server=zztop.5161 user='user3'
       password='spds123' aclgrp='group2';

/* Generate some dummy data. */
data user1.t;
id=1;
salary=2000;
run;

/* Example of only user2 in group2 */
/* being allowed to read column    */
/* salary.                          */

PROC SPDO library=user1 ;

/* Assign who owns the ACLs. */
```

```
set acluser;

/* Clean Up */
delete ACL t;
delete ACL t.salary;

/* Create an ACL on table t to     */
/* allow members of group2 to read */
/* table.                           */

add ACL t;
modify ACL t / group2=(y,n,n,n);

/* Create an ACL on column t.salary*/
/* to only allow user2 of group2 to */
/* read the column.                 */

add ACL t.salary;
modify ACL t.salary / user2=(y,n,n,n);
quit;

/* Let both users print the table. */
/* Only user2 can access column    */
/* salary.                          */

proc print data=user2.t;
run;

proc print data=user6.t;
run;

/* Example of every BUT user2 in */
/* group2 being allowed to read  */
/* column  salary.               */

PROC SPDO library=user1 ;

/* Assign who owns the ACLs. */
set acluser;

/* Clean up column ACL. */
delete ACL t.salary;

/* Create an ACL on column t.salary */
/* to only allow members of group2 to */
/* read the column.                 */

add ACL t.salary;
modify ACL t.salary / user2=(y,n,n,n);


/* User permissions have priority over */
/* group permissions.  So now deny     */
/* user2 access to column salary.       */

modify ACL t.salary / user2=(n,n,n,n);
```

```
quit;

/* Let both users print the table. */
/* Only user6 can access column    */
/* salary.                           */

proc print data=user2.t;
run;

proc print data=user6.t;
run;
quit;
```

Chapter 15

Defining WHERE Constraints on Tables

Table WHERE Constraints 169
Overview of Table WHERE Constraints 169
Creating and Controlling Table WHERE Constraints 169
Table Constraint WHERE Clauses 170
Table Constraint Restrictions and Limitations 170
Example Table Constraint WHERE Clauses 170

Table WHERE Constraints

Overview of Table WHERE Constraints

SPD Server table WHERE constraints enable table owners to associate a WHERE clause with a table. This association means that the WHERE clause is applied when a user accesses the table. As a result, the user sees only the table rows that remain after the table owner's WHERE clause filter has been processed. The filtering is applied when the table is accessed by a DATA step or by an SQL query. You can use table WHERE constraints with symbolic substitutions to create row-level security that filters table rows. Filtering is based on the values for user ID, group ID, or ACLSPECIAL privileges. SPD Server table WHERE constraints do not affect normal table ACL settings, such as Read, Write, and Column access.

WHERE constraints are not applied to a table that is opened for Write access. The table owner should grant Update access only to users who can modify any row of the table, delete any row of the table, or append new rows to the table.

Creating and Controlling Table WHERE Constraints

To create a table WHERE constraint, you must be the table owner. Use PROC SPDO statements to define, delete, and describe table WHERE constraints.

To add a table WHERE constraint, enter the following:

```
constraint add table-name;
where where-clause-expression;
```

To describe a table WHERE constraint, enter the following:

```
constraint describe table-name;
```

To remove a table WHERE constraint, enter the following:

```
constraint remove table-name;
```

Table Constraint WHERE Clauses

A table constraint WHERE clause can be any WHERE clause that SPD Server can parse and interpret. The WHERE clause can use any function that SPD Server supports. Generally, the WHERE clause uses symbolic substitution to create row-level security. This security is based on the values for user ID, group, or ACLSPECIAL privileges. For more information, see "Symbolic Substitution" on page 173.

The table constraint WHERE clause is applied when a user accesses the table. If the WHERE clause specifies the table owner, then the WHERE clause is also applied when the owner accesses the table. Table owners should be careful to construct WHERE clause constraints that allow only the table owner to see all of the table rows.

Table Constraint Restrictions and Limitations

The SPD Server SQL COPY TABLE and UPDATE TABLE extensions do not function on a table that has a table constraint WHERE clause. You must use a different method (such as PROC COPY or SQL CREATE TABLE as SELECT) to copy the table. If you try to use COPY TABLE or UPDATE TABLE, the destination table contains rows that were filtered by the WHERE constraints on the source table only. The destination table does not inherit the WHERE constraints of the source table.

Example Table Constraint WHERE Clauses

Assume that the following users are defined in the password database:

```
User      Password    Group    ACLSPECIAL
----      --------    -----    ----------
William   worker1     group1   no
Guy       worker2     group1   no
Frank     worker3     group2   no
Ed        worker4     group2   no
Bossman   worker5     group2   no
Aclspec   worker6     group3   yes
```

The user **Bossman** creates a table that contains sensitive data. Only some users or user groups are authorized to see certain rows in the table. **Bossman** uses WHERE constraints to control which table rows are available to different users or user groups.

```
libname foo sasspds 'tmp' ... user="bossman";
DATA foo.employees;
input user $ grp $ salary $ position $ state $ region $;
cards;
WILLIAM GROUP1 70000 Engr1 CA W
GUY GROUP1 60000 Engr1 CA W
ED GROUP1 50000 Engr2 NJ E
FRANK GROUP2 80000 Engr2 TX S
TOM GROUP2 65000 Engr3 WA W
BOSSMAN GROUP2 80000 Mgr NJ E;
```

The following code provides Read access for all users. The code uses table WHERE constraints to determine which table rows a user or user group can read:

```
PROC SPDO lib=foo;
SET ACLUSER;
ADD ACL employees;
MODIFY ACL employees / read;
quit;
```

The user **Bossman** can create table WHERE constraints to control access to the table in the following ways:

- **Bossman** can read any row of the table. Any other user can read only rows where the value in the User column matches the user's user ID.

  ```
  PROC SPDO lib=foo;
  SET ACLUSER;
  constraint add employees;
   WHERE (User = "@SPDSUSR")
   or ("@SPDSUSR" = "BOSSMAN");
  constraint describe employees;
  quit;
  ```

- **Bossman** or an ACLSPECIAL user can read any row of the table. Any other user can read only rows where the value in the User column matches the user's group ID.

  ```
  PROC SPDO lib=foo;
  SET ACLUSER;
  constraint remove employees;
  constraint add employees;
   WHERE (grp= "@SPDSGRP")
   or ("@SPDSUSR" = "BOSSMAN")
   or ("@SPDSSPEC" = "TRUE");
  constraint describe employees;
  quit;
  ```

- **Bossman** can read all rows. Other users can read only rows where the value in the User column matches the user's user ID, except for user **Guy**. User **Guy** can also read rows for employees from the state of California.

  ```
  PROC SPDO lib=foo;
  SET ACLUSER;
  constraint remove employees;
  constraint add employees;
   WHERE (User = "@SPDSUSR")
   or ("@SPDSUSR" = "BOSSMAN")
   or ("@SPDSUSR" = "GUY" and state = "CA");
  constraint describe employees;
  quit;
  ```

- **Bossman** can read all rows. Other users can read only rows where the value in the User column matches the user's user ID, or rows where the value in the Salary column is less than or equal to $6,000 per month (rounded down to the nearest dollar).

  ```
  PROC SPDO lib=foo;
  SET ACLUSER;
  constraint remove employees;
  constraint add employees;
   WHERE (User = "@SPDSUSR")
   or ("@SPDSUSR" = "BOSSMAN")
   or (floor(salary/12) <= 6000);
  ```

```
constraint describe employees;
quit;
```

- **`Bossman`** can read all rows. User **`William`** can read rows for employees who belong to the West region. User **`Ed`** can read rows for employees who belong to the East region. User **`Frank`** can read rows for employees who belong to the South region.

```
PROC SPDO lib=foo;
SET ACLUSER;
constraint remove employees;
constraint add employees;
 WHERE (("@SPDSUSR" = "BOSSMAN")
 or ("@SPDSUSR" = "WILLIAM" and REGION = "W")
 or ("@SPDSUSR" = "ED" and REGION = "E")
 or ("@SPDSUSR" = "FRANK" and REGION = "S"));
constraint describe employees;
quit;
```

Chapter 16

Symbolic Substitution

Symbolic Substitution **173**
Overview of Symbolic Substitution 173
Symbolic Substitution and Row-Level Security 173
Symbolic Substitution Example 174

Symbolic Substitution

Overview of Symbolic Substitution

SPD Server SQL supports symbolic substitution of the following items in SQL queries:

- a user ID, which is substituted by @SPDSUSR
- a group, which is substituted by @SPDSGRP
- a user who has special privilege, which is substituted by @SPDSSPEC

The right-hand side of symbolic substitution statement must be in all uppercase text (for example, @SPDSUSR=SOMEUSER).

Symbolic Substitution and Row-Level Security

A powerful use of symbolic substitution is to deploy row-level security on sensitive tables that use views. Suppose that only certain users or groups can access a sensitive table. You can use symbolic substitution to create a single view to the table that provides restricted access based on user ID or groups. You can grant Universal access to the view, but only users or groups that meet the symbolic substitution constraints can see the rows.

For another example, imagine a table that contains sensitive information has a column that contains group names or user IDs. You can use symbolic substitution to create a single view that allows users to access only the rows that contain their user ID or group. You can grant Universal access to the view, but each user or group is allowed to see only their user or group rows.

CAUTION:

SPD Server SQL symbolic substitution uses an 8-byte literal string (blank padded if necessary) to replace user and group names. Symbolic substitution will not match a column that is less than 8 characters wide. If the table column that contains user IDs or group names is not at least 8 characters wide, symbolic substitution will evaluate the WHERE- predicate on that column to be FALSE, which can result in incorrect results.

Symbolic Substitution Example

```
proc sql;
     connect to sasspds
       (dbq="path1"
        server=host.port
        user='anonymous');

  /* Queries comparing literal rows are  */
  /* only selected if the symbolic       */
  /* substitution evaluates as 'true'.    */

    select *
    from connection
    to sasspds(
       select *
       from mytable
       where "@SPDSUSR" = "SOMEUSER");

    select *
    from connection
    to sasspds(
       select *
       from mytable
       where "@SPDSGRP" = "SOMEGROUP");

    select *
    from connection
    to sasspds(
       select *
       from mytable
       where "@SPDSSPEC" = "TRUE");

  /* Queries based on column values will only  */
  /* select appropriate columns.                */

    select *
    from connection
    to sasspds(
       select *
       from mytable
       where usercol = "@SPDSUSR");

    select *
    from connection
    to sasspds(
       select *
       from mytable
       where grpcol = "@SPDSGRP");

  /* Create a view to worktable that allows    */
  /* users FRED or BOB, groups BCD or ACD, or  */
  /* someone with ACLSPECIAL to read the table. */
```

```
   execute(create view workview as
       select *
       from worktable
       where "@SPDSUSR" in ("FRED", "BOB") or
             "@SPDSGRP" in ("BCD", "ACD") or
             "@SPDSSPEC" = "TRUE")
   by sasspds;

 /* Create a view to worktable that allows users    */
 /* to access only rows where the column "usergrp"  */
 /* matches their group. The userID BOSS can access */
 /* any group records where the column "userid" is  */
 /* "BOSS"                                          */

   execute(create view workview as
       select *
       from worktable
       where usergrp = "@SPDSGRP" and
       ("@SPDSUSR" = "BOSS" or userid != "BOSS"))
   by sasspds;
disconnect from sasspds;
quit;
```

Chapter 17
DICTIONARY.PWDB and DICTIONARY.ACLS

DICTIONARY.PWDB and DICTIONARY.ACLS **177**
Overview of Dictionaries 177
Example: Listing the Users in the Password Database Using SQL Pass-Through 177
Example: Listing ACL Objects Using SQL Pass-Through 178

DICTIONARY.PWDB and DICTIONARY.ACLS

Overview of Dictionaries

In addition to providing dictionary information for tables and columns, SPD Server provides information about the users in the password database and the ACL objects that are available. The following listing shows the column definitions for DICTIONARY.PWDB and DICTIONARY.ACLS:

```
DICTIONARY.PWDB {user char(8) Label = 'User'
  auth_lvl char(5) Label = 'Authorization Level'
  ip_addr char(16) Label = 'IP Address'
  defgrp char(8) Label = 'Default Group'
  othgrps char(40) Label = 'Other Groups'
  expire char(6) Label = 'Expire Period'
  mod_date char(32) Label = 'Password Last Modified'
  log_date char(32) Label = 'Last Login'
  timeout char(6) Label = 'Timeout Period'
  strikes char(6) Label = 'Failed Login Attempts'}

DICTIONARY.ACLS {owner char(8) Label = 'Owner'
  group char(8) Label = 'Group'
  defacs char(56) Label = 'Default Access'
  grpacs char(56) Label = 'Group Access'}
```

Example: Listing the Users in the Password Database Using SQL Pass-Through

To use SQL pass-through, you must first establish an SQL pass-through connection to SPD Server using ACLSPECIAL=YES.

```
proc sql;

  connect to sasspds     (dbq='tmp'      server=localhost.5400
     user='admin'
     password='spds123'
     ACLSPECIAL=YES);
```

Note: Without ACLSPECIAL=YES, you get the result set only for the users who are making a pass-through connection and not for all users.

To list all the users in the password database, submit the following command:

```
       select *
       from connection
       to sasspds
          (select *
           from dictionary.pwdb)
```

To select only the user name and last log in date, submit the following command:

```
       select *
       from connection
       to sasspds
          (select user, log_date
           from dictionary.pwdb);
```

Example: Listing ACL Objects Using SQL Pass-Through

To list all ACL objects for a user by using a pass-through connection, first establish an SQL pass-through connection to SPD Server using ACLSPECIAL=YES.

```
proc sql;

  connect to sasspds     (dbq='tmp'      server=localhost.5400
     user='admin'
     password='spds123'
     ACLSPECIAL=YES);
```

Note: Without ACLSPECIAL=YES, you get the result set only for the users who are making a pass-through connection and not for all users.

Then, submit the following command:

```
       select *
       from connection
       to sasspds
          (select *
           from dictionary.acls);
```

To find any ACL objects that specify Jones as the owner, submit the following command:

```
       select *
       from connection
       to sasspds
```

```
(select *
 from dictionary.acls
 where owner = "Jones");
```

Chapter 18
Audit File Facility

Audit File Facility **181**
Overview 181
Proxy Auditing 181
WHERE Clause Auditing 182
SQL Query Auditing 183

Audit File Facility

Overview

SPD Server supports SQL audit logging of submitted SQL queries and proxy auditing of access to SPD Server resources. SPD Server proxy auditing and SQL audit logging (spdsaud) are enabled when the server is started using the -AUDITFILE or -SQLAUDITFILE parameters. You can enable proxy auditing, SQL audit logging, or both.

SPD Server auditing logs access to SPD Server resources. Auditing also logs implicit or explicit SQL pass-through queries that are submitted to SPD Server. Separate audit logs are created for proxy auditing and SQL audit logging. SPD Server includes three SAS programs (auditwithwhere.sas, auditraw.sas, and auditsql.sas) in the **`/samples`** directory of your SPD Server installation. You can use these programs to input the audit logs in to SAS tables. You can then query the SAS tables to determine access to SPD Server tables and resources.

Proxy Auditing

Proxy auditing helps you determine access to SPD Server resources. The audit record contains the following information:

- the activity timestamp
- the primary path of the domain that contains the resource
- the LIBNAME of the domain
- the user ID of the SPD Server user that is accessing the resource
- the resource name
- the resource type
- the SPD user ID of the resource

- the SPD group ID of the resource
- the resource operation type for librefs:
  - ASSIGN
- the resource operation type for tables:
  - DELETE
  - RENAME
  - OPEN
  - REOPEN
  - REPAIR
  - TRUNC
- the resource operation type for clusters:
  - CREATE
  - UNDOCL
  - ADDCL
- the resource operation type for a WHERE clause:
  - WHERE
- the resource operation mode for librefs:
  - ACCESS
- the resource operation mode for tables and clusters:
  - OUTPUT
  - INPUT
  - UPDATE
  - UTILITY
- access requested to a resource by a user
- access granted to a resource for the user
- the ACLs that are associated with a resource

WHERE Clause Auditing

WHERE clause auditing provides an audit record that contains the following information:

- the length of the WHERE clause
- the contents of the WHERE clause

You enable WHERE clause auditing by using the WHEREAUDIT option. The maximum size that can be allocated to WHERE clauses is controlled by the WHAUDLEN option. For more information about these options, see Chapter 10, “Setting Server Operational Parameters,” on page 83.

SQL Query Auditing

SQL audit logging provides a record of the SQL queries that were submitted to the SPD Server server. The SQL audit record contains the following information:

- the SQL query timestamp
- the type of the SQL query
  - 1=SELECT
  - 2=DROP
  - 3=ALTER
  - 4=CREATE
  - 5=DESCRIBE
  - 6=UPDATE
  - 7=DELETE
  - 18=RESET
  - 19=BEGIN ASYNC
  - 20=END ASYNC
- the number of rows that were returned for an SQL SELECT statement
- the elapsed time, in seconds, required to process the SQL query
- the user ID of the user that submitted the query
- the group ID of the user that submitted the query
- the default LIBNAME for the query, used for any table that is not referenced by a two-part name
- the number of characters in the query
- the text of the submitted SQL query

You control the maximum size that can be allocated in the SQL log for an SQL statement by using the SQLAUDLEN option. For more information, see Chapter 10, “Setting Server Operational Parameters,” on page 83.

Chapter 19

Securing SPD Server Against Unknown Client Attacks

Securing SPD Server Against Unknown Client Attacks **185**
Overview of SPD Server Security for Unknown Client Attacks 185
SPDSLOGBADCONNECT= Environment Variable 185

Securing SPD Server Against Unknown Client Attacks

Overview of SPD Server Security for Unknown Client Attacks

SPD Server is a client-server-based architecture that allows outside clients to connect to SPD Server processes on well-known published ports. SPD server also uses ephemeral (unpublished) ports for internal communication between SPD Server processes and clients.

SPD Server provides protection against attacks from unknown clients in the following ways:

- SPD Server uses a proprietary communication protocol. All messages that are received are examined to determine whether they are valid. Any invalid message is discarded.
- SPD Server provides the option of logging any unknown communications in the SPD Server log files. Unknown communications can be logged from the spdsnsrv process, the spdserv process, the spdssnet process, and the spdsbase process. The log messages include the IP address of the client that sent the message.
- SPD Server processes effectively handle an invalid message by continuing to function properly for subsequent valid messages.

In some cases, third-party port checking software can cause unexpected growth in SPD Server logs. If this is the case, set the SPDSLOGBADCONNECT= environment variable in your rc.spds start-up script to control how port access attempts are logged.

SPDSLOGBADCONNECT= Environment Variable

To enable logging of invalid messages, export the SPDSLOGBADCONNECT=Y environment variable in your rc.spds start-up script. You must export the variable value before any SPD Server processes are started. All SPD Server processes that communicate will check the status of the environment variable on start-up, and log accordingly.

Usage:

Specify the SPDSLOGBADCONNECT=Y environment variable in your SPD Server rc.spds start-up script.

```
export SPDSLOGBADCONNECT=[Y|N]
```

Default:

The default value of SPDSLOGBADCONNECT= is N, which configures SPD Server processes to not log failed connections.

Part 5

Advanced Configuration Topics

Chapter 20
Optimizing Server Performance *189*

Chapter 21
Optimizing Disk Storage *195*

Chapter 22
Setting Up the Performance Server *201*

Chapter 23
Managing SPDSBASE Processes *205*

Chapter 20
Optimizing Server Performance

SPD Server Performance and Usage Tips **189**
Symmetric Multiple Processor (SMP) Utilization **189**
File System Performance Concepts **190**
Overview of File System Performance 190
Defining Directories 191
Disk Striping 191
RAID Levels 191
Transient Storage 192
SPD Server Domains **192**
Data and Index Separation 192
Configuring an SPD Server Domain 193
Optimizing Performance with Macro Variables **193**

SPD Server Performance and Usage Tips

SPD Server gives good performance when run using default configuration settings. To realize the full benefits of the server's design and capabilities, you might want to configure some of the software's options to modify the default behaviors. The configuration changes will depend on the computing environment, table size and complexity, and indexing structures. The server itself can be configured. This chapter discusses items that should be considered to optimize performance.

Symmetric Multiple Processor (SMP) Utilization

SPD Server uses parallel processing where possible to increase performance. Parallel processing uses multiple processors to execute more than one set of instructions, or threads, concurrently. The server is oriented to exploit parallelism whenever it can improve I/O times and processor utilization.

A fundamental question about parallelism is whether using additional CPUs on a specific problem will deliver data faster. Extra CPUs do not guarantee faster results every time. The amount of CPU-intensive work that a thread must do needs to take long enough to justify the cost of the thread. The cost of the thread is creating it, managing it, and interacting with other threads involved in the same parallel algorithm.

If not properly matched to the workload, the parallel algorithm can use more CPU time without reducing data delivery time. Additional threads can create conflicting demands for critical system resources such as physical memory. Excessive execution times can occur if too many threads attempt to access a large table at the same time, because many threads demand large amounts of physical memory. Extreme resource constraints can result in slower overall processing.

The server focuses on the following areas to speed overall processing using parallelism:

- User-definable parallel execution blocks for SQL explicit pass-through statements.
- Parallel aggregation for common summary functions when performing SELECT [...] GROUP BY statements.
- WHERE clause evaluation for indexed and non-indexed strategies.
- Overlapped table and concurrent index updates when appending to tables.
- Index creation when creating multiple indexes.
- Optimize PROC SORT BY clauses.
- Pipelined read-ahead when concurrently accessing multiple tables.

File System Performance Concepts

Overview of File System Performance

SPD Server uses several file types in its data storage model. Data objects in the server consist of one or more component files. Each component file is itself a collection of one or more disk files. These are called the partitions of the component.

Component files create partitions when any of the following conditions is true:

- The current partition exceeds the user-specified PARTSIZE= value: Subsequent partitions are allocated in cyclical fashion across the set of directories that are specified in the DATAPATH= statement for the server domain. Partitioning uses file-level striping to create PARTSIZE-sized files that complement the disk-level striping that your operating system's volume manager software creates. The server uses a default PARTSIZE= setting of 16 MB. PARTSIZE= determines a unit of work for parallel operations that require full table scans. Examples of parallel operations that require full table scans are WHERE clause evaluation and SQL GROUP-BY summarization. Trade-offs are balancing increased numbers of files used to store the table versus the work savings realized through parallel partitions. Extra partitions mean that files are opened to process a table, but have fewer rows in each partition.
- The current partition exceeds the RLIMIT_FILESIZE value: In UNIX systems, RLIMIT_FILESIZE is a system parameter that defines the maximum size of a single disk file. In Windows, SPD Server uses a default RLIMIT_FILESIZE value of 2 GB.
- The current partition exceeds the space on the file system where it has been created.

Defining Directories

SPD Server enables the user to define a set of directories that contain component files and their partitions. Normally, a single directory path is constrained by some volume limit for the file system, or the maximum amount of disk space that the operating system understands.

Most UNIX and Windows systems offer a volume manager utility. You can use volume manager utilities to create file systems (volumes) that are greater than the available space on a single disk. System administrators can use these utilities to create large, multi-gigabyte volumes. These volumes can be spread across a number of disk partitions, or even span multiple disk devices. Volume manager utilities generally support creation of disk volumes that implement one of the common RAID (redundant arrays of inexpensive disks) configuration levels.

Disk Striping

A defining feature of all RAID levels is disk striping. Striping organizes the linear address space of a volume into pieces that are spread across a collection of disk drive partitions. For example, a user can configure a volume across two 1-GB partitions on separate disk drives A and B with a stripe size of 64K bytes. Stripe 0 lives on drive A, stripe 1 lives on drive B, stripe 2 lives on drive A, and so on.

By distributing the stripes of a volume across multiple disks it is possible to do the following:

- achieve parallelism at the disk I/O level
- use multiple kernel threads to drive a block of I/O

This also reduces contention and data transfer latency for a large block I/O because the physical transfer can be split across multiple disk controllers and drives.

RAID Levels

Here is a brief summary of RAID levels that are relevant to SPD Server:

RAID-0
: High performance with low availability. Physically losing a disk means that data is lost. No redundancy exists to recover volume stripes on a failed disk.

RAID-1
: Disk mirroring for high availability. Every block is duplicated on another mirror disk, sometimes referred to as shadowing. In the event one disk is lost, the mirror disk is still likely to be intact, preserving the data. RAID-1 can also improve Read performance because a device driver has two potential sources for the same data. The system can choose the drive that has the least load or latency at a given point in time. The downside to RAID-1: iIt requires twice the number of disk drives as RAID-0 to store a given amount of data.

RAID-5
: High performance and high availability at the expense of resources. An error-correcting code (ECC) is generated for each stripe written to disk. The ECC distributes the data in each logical stripe across physical stripes. This is done in such a way that if a given disk in the volume is lost, data in the logical stripe can still be recovered from the remaining physical stripes. RAID-5's downside is resource

utilization; RAID-5 requires extra CPU cycles and extra disk space to transform and manage data using the ECC model.

RAID-1+0
: Many RAID systems offer a combination of RAID-1 (pure disk mirroring) and RAID-0 (striping) to provide both redundancy and I/O parallelism in a configuration known as RAID-1+0 (sometimes referred to as RAID-10). Advantages are the same as for RAID-1 and RAID-0. The only disadvantage is the requirement for twice as much disk space as the pure RAID-0 solution. Generally, this configuration tends to be a top performer if you have the disk resources to pursue it.

Regardless of RAID level, disk volumes should be hardware-striped when using the SPD Server software. This is a significant way to improve performance. Without hardware striping, I/O will bottleneck and constrain SPD Server performance.

Transient Storage

You should configure a RAID-0 volume for WORKPATH= storage for your SPD Server. When sizing this RAID-0 volume, keep in mind that the WORKPATH= that you set up a given server host must be shared by all of its SQL and LIBNAME proxy processes that exist at a given point in time.

Consider using one or more RAID-0 volumes to locate the database domains that will support TEMP=YES LIBNAME assignments. This LIBNAME statement option creates a temporary storage domain that exists only for the duration of the LIBNAME assignment. This is the SPD Server equivalent of the SAS WORK library. All data objects (tables, catalogs, and utility files) that are created in the TEMP=YES temporary domain are automatically deleted when you end the SAS session.

SPD Server Domains

Server domains define the primary directory path and can, if desired, define other directories for placing the data and index components of server tables. The PATHNAME=, METAPATH=, DATAPATH=, and INDEXPATH= domain path options determine the placement of the server's component and partition files.

Data and Index Separation

The section on “File System Performance Concepts” on page 190 discussed how distributing I/O load across different disk drives can improve performance. Further load distribution can be achieved by separating data and index components of server tables. To do this, use the DATAPATH= and INDEXPATH= options when configuring domains.

For example, when performing complex WHERE clause evaluations, multiple threads are active on index component files and the data component file at the same time. Splitting the index and data file components into different volumes can improve performance by reducing disk contention and increasing the level of parallelism down to the disk-access level.

A word of caution when using DATAPATH= and INDEXPATH= options to distribute the data and index components: Take extra care when performing and restoring disk backups of the server tables using a system backup and restore utility. When making a backup, ensure that the metadata, data, and index component partition files are of the same generation and are in their respective directories.

When restoring a backup, restore the component partitions to the same directories where they were created. To avoid this restore problem, create symbolic links with the original directory path that point to the restore directories. Of course, if the components are not separated using the path options, this restore issue does not apply.

The backup and restore issues are not an issue when using the SPD Server backup and restore utilities. These utilities resolve any component files when backing up or restoring tables. For more information, see Chapter 27, “Backing Up and Restoring SPD Server Data,” on page 269.

Configuring an SPD Server Domain

Suppose a user has designated four volumes. Volumes exist for (1) SPD Server metadata, (2) data components, (3) index components, and (4) proxy working storage, as follows:

- **/dmart_domain** is a 500 GB volume
- **/dmart_data** is a 3 TB volume
- **/dmart_index** is a 3 TB volume
- **/spds_work** is a 1 TB volume

The user wants to configure a server domain called **dmart**. Dmart will use **/dmart_domain** for the primary directory; dmart data components will reside in **/dmart_data**; and dmart index components will reside in **/dmart_index**. The **/spds_work** volume should be configured for proxy working storage.

The configuration is made in two steps:

1. In the spdsserv.parm parameter file, enter the following line:

   ```
   WORKPATH=/spds_work;
   ```

2. In the libnames.parm parameter file, enter the following domain definition:

   ```
   libname=dmart
      path=/dmart_domain
      roptions="datapath=('/dmart_data')
      indexpath=('/dmart_index')";
   ```

For more information about configuring component files, see Chapter 21, “Optimizing Disk Storage,” on page 195.

Optimizing Performance with Macro Variables

The default macro variable values automate processing decisions. The default settings provide good performance. However, optimal performance requires changes to the default settings of some macro variables. Before you make changes to the default settings, consider conducting performance testing first. After you quantify performance parameters by using several macro variable settings, you can customize SPD Server so that it solves your business or data problems with maximum efficiency.

Each SPD Server installation is different. You might want to change many values, or just a few. Either way, macro variables are flexible and easy to manipulate.

You can place the macro variable assignment anywhere in the open code of a SAS program except in the data lines. The most convenient location for your %LET statements to initialize macro variables is in your autoexec.sas file or at the beginning of a program. The macro variable assignment is valid for the duration of your session or the executing program. Macro variable values remain in effect until they are changed by a subsequent assignment.

For more information, see "SPD Server Macro Variables" in *SAS Scalable Performance Data Server: User's Guide*.

Chapter 21
Optimizing Disk Storage

Introduction **195**

SPD Server Component File Types and Sizes **195**

Configuring Domain Disk Space **196**
Example 1: Primary File System Storage for All Component Files 196
Example 2: Add Paths for Data and Index Component Files with ROPTIONS= . 197
Example 3: Adding More File Systems to a Path Option
When Its File System Is Full 197

How Table Partition Size Affects the Data Component **198**

Component File Pathnames **198**

Introduction

This section discusses how to manage large SPD Server data stores that can consume terabytes of disk storage.

How you configure SPD Server disk storage is important, whether you have many SPD Server users or a just a few large-scale users. To effectively configure SPD Server disk storage for your installation, you need to understand the four types of component files that SPD Server creates, the relative sizes of these files, and when these files are created.

SPD Server Component File Types and Sizes

SPD Server uses four types of component files. Component files are physical file entities that SPD Server uses to track table and index metadata. When component files are combined, they form a logical structure that SPD Server understands and interprets as a single table. The following table lists the relative sizes of the four types of SPD Server component files.

File Type	File Extension	Relative Size	Number of Component Files
Table metadata	.mdf	Very small	1

File Type	File Extension	Relative Size	Number of Component Files
Table data	.dpf	Large	1-to-many
Index	.idx	Medium-to-large	1 or more per index
Index	.hbx	Medium-to-large	1 or more per index
Cluster table metadata	.cdf	Very small	1

At a minimum, an SPD Server table consists of two component files: the metadata .mdf file and the data .dpf file. The size of the data file component depends on two factors: the size of a table column and the number of columns. The data .dpf component file can be many gigabytes in size. A single table can consist of many .dpf files. SPD Server is not constrained by an operating system file system size limit (such as the 2 GB limit on file size that some UNIX systems impose).

An SPD Server index uses two index component files: the .hbx file and the .idx file. The .hbx file maintains a global view of the index and contains an entry for each key that exists in the index. The .idx file contains the row identifiers for each value in the .hbx file.

The size of the .hbx file depends on the cardinality of the index keys. The higher the cardinality of the index keys, the larger the .hbx file. The size of the .idx file is more difficult to determine because it depends on how the data is distributed.

The best-case scenario for creating an optimally sized .idx file occurs when the table is sorted by the indexed columns. The worst-case scenario for creating an optimally sized .idx file occurs when the index keys are distributed throughout the table.

Configuring Domain Disk Space

You must define a primary file system for each server domain. In addition, you can define initial and overflow storage locations for the .dpf data component files, as well as for the two index component (.hbx and .idx) files.

Example 1: Primary File System Storage for All Component Files

This example shows how to define the primary file system for a server domain in the libnames.parm parameter file. The primary file system is the base directory that you assign to a server domain with the PATHNAME= statement.

Here is a sample libnames.parm parameter file entry for a UNIX system:

```
LIBNAME=everyone
  pathname=/disk1/usertables;
```

Here is a sample libnames.parm parameter file entry for Windows:

```
LIBNAME=everyone
  pathname=d:\usertables;
```

When SPD Server users create new tables in a server domain, you must keep in mind that the metadata component (.mdf) must start in the primary file system. If all the available space in the primary file system is consumed, SPD Server cannot create new tables until disk space becomes available.

Example 1 stores all the component files (metadata, data, and index data) in the primary file system. This arrangement can cause problems if you use large tables. Large tables can quickly fill up the primary file system. To avoid this problem, you can store the data and index components separately from the primary file system.

Example 2: Add Paths for Data and Index Component Files with ROPTIONS=

This example shows how to store data and index component files separately from the primary file system. Domain path options are defined by specifying ROPTIONS= in the domain definition.

Here is a sample libnames.parm parameter file entry for a UNIX system that specifies ROPTIONS=:

```
   LIBNAME=everyone pathname=/disk1/usertables
      roptions="datapath=('/disk2/userdata' '/disk3/userdata'
                        '/disk12/userdata' '/disk13/userdata')
      indexpath=('/disk4/userindexes' '/disk5/userindexes'
                 '/disk14/userindexes' '/disk15/userindexes')";
```

Here is a sample libnames.parm parameter file entry for a Windows system that specifies ROPTIONS=:

```
   LIBNAME=everyone pathname=d:\usertables
      roptions="datapath=('e:\userdata' 'f:\userdata')
         indexpath=('g:\userindexes' 'h:\userindexes')";
```

In Example 2, the PATHNAME= directory stores metadata files for server tables in the **`everyone`** server domain. The initial and overflow stores for the data and index files are directed to other file systems. In Example 2, users who create large tables will not quickly exhaust the primary file system because the primary file system is reserved for small metadata files only. The larger data and index files are stored in the other file systems that are specified with the DATAPATH= and INDEXPATH= options.

Example 3: Adding More File Systems to a Path Option When Its File System Is Full

If the existing file system is running out of space, simply add file systems to your path as follows:

Sample libnames.parm entry for a UNIX system:

```
LIBNAME=everyone pathname=/disk1/usertables
   roptions="datapath=('/disk2/userdata' '/disk3/userdata'
                     '/disk12/userdata' '/disk13/userdata')
   indexpath=('/disk4/userindexes' '/disk5/userindexes'
              '/disk14/userindexes' '/disk15/userindexes')";
```

Sample libnames.parm entry for a Windows system:

```
LIBNAME=everyone
```

```
pathname=d:\usertables
   roptions="datapath=('e:\userdata'
                       'f:\userdata'
                       'i:\userdata')
   indexpath=('g:\userindexes'
              'h:\userindexes'
              'j:\userindexes')";
```

Users can create new server tables in a domain as long as space is available for their metadata files in the primary file system. When the primary file system is exhausted, you must create a new server domain. You cannot expand storage for the .mdf components by adding the METAPATH= specification to your ROPTIONS= value in your libnames.parm parameter file. METAPATH= can be used for overflow metadata for existing tables (update and append operations to existing tables).

How Table Partition Size Affects the Data Component

When partitioning data, the server uses file systems that are specified in the DATAPATH= option to distribute partitions in a cyclic, round-robin pattern. Instead of creating partitions until the first file system is full, the server randomly chooses a file system from the DATAPATH= list for the first partition. Then, it sequentially assigns partitions to successive file systems in the DATAPATH= list. The software continues to cycle through the file system set as many times as needed until all data partitions for the table are stored.

If you set PARTSIZE=0, the server uses the DATAPATH= file systems strictly for overflow. It creates partitions in the first file system, up to the file size limit of your operating system. When the first file system is full, it proceeds to the second file system, and so on. When the primary file system is full, specifying additional data paths will enable users to append to an existing table. However, users will not be able to create new tables in the domain. In order for a new table to be created, the primary file system must be able to store the first metadata file partition for the table. You will need to either free space in the primary file system or create a new server domain to enable new tables to be created.

Component File Pathnames

SPD Server creates pathnames for table data partition files and index files in this form:

table_name.file_extension.primary_path.n1.n2.spds9

It creates metadata file pathnames and cluster table pathnames in this form:

table_name.file_extension.0.0.0.spds9

table_name
: For a data file and index file, *table-name* is the domain DATAPATH= or INDEXPATH= location where the table data file or index file is stored and the table filename, unless the domain definition does not specify these options. In that case, *table-name* is the domain PATHNAME= location and table filename.

  For a metadata file and cluster table, *table-name* is the domain's PATHNAME= location and the table filename. All metadata files are stored in the primary path of the domain. Information about any METAPATH= overflow files is stored in the metadata file.

file-extension
: specifies the component file type, and for index files, also the index name. Valid extensions are .cdf, .dpf, .idx*index-name*, .hbx*index-name*, or .mdf.

primary-path
: the PATHNAME= value for the domain.

n1
: the component file number (if you have 20 .dpf files for a table, this will be 0-19).

n2
: the update count for the table. This field is not always used.

Here are examples of component filenames that might be created for a table named Table1.

Metadata file:

UNIX and Windows:

```
table1.mdf.0.0.0.spds9
```

Data file:

UNIX:

```
table1.dpf._disk1_usertables.1.0.spds9
```

Windows:

```
table1.dpf.d__usertables.1.0.spds9
```

Index files:

UNIX:

```
table1.idxy._disk1_usertables.1.0.spds9
table1.hbxy._disk1_usertables.1.0.spds9
```

In this example, the index name is Y.

Windows:

```
table1.idxy.d1__usertables.1.0.spds9
table1.hbxy.d__usertables.1.0.spds9
```

Note: The primary path portion of Windows data and index filenames have a double underscore (_ _) between the drive name and the directory name.

SPD Server provides a listing utility, spdsls, to enable you to list the component files in a domain. For more information, see Chapter 29, “Domain List Utility,” on page 299.

Chapter 22

Setting Up the Performance Server

Overview of SPD Server Performance Server 201

Starting SPD Server Performance Server 202

Performance Server Log File 203

Converting Your spdsperf Log into a Table 203

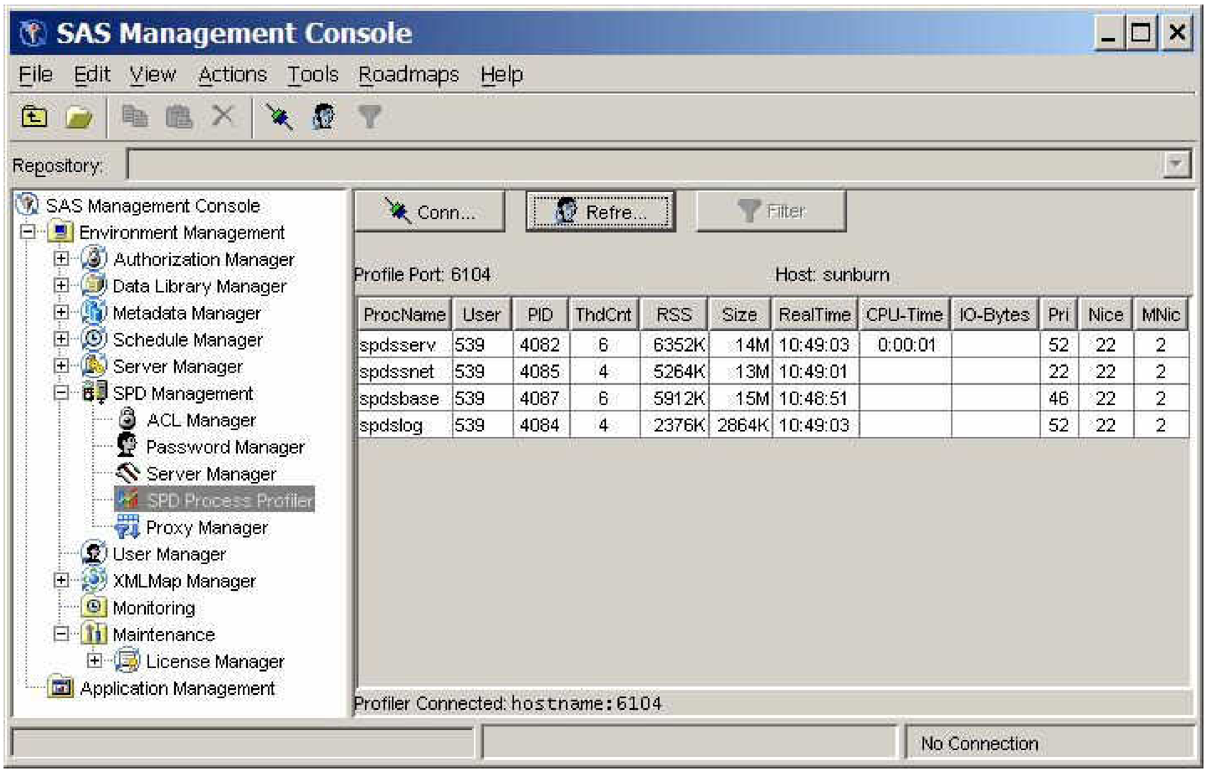
Overview of SPD Server Performance Server

SPD Server provides a performance monitoring server called spdsperf. SPD Server Performance Server is an optional component and is not required for the normal operation of the server.

Note: SPD Server Performance Server is currently not available for the Windows or Linux X64 platforms.

The performance server gathers server process performance information and posts it to the **Server Management** section of the SAS Management Console application. The information consists of memory and resource allocations by users, and server processes that were spawned by SPD Server. SPD Server owns a dynamic family of subordinate server processes that users and jobs create and terminate.

The information that is gathered by the performance server is stored in the SAS Management Console. The SAS Management Console has a folder that is reserved for SPD Server management. When you expand the **SPD Management** folder, the next to last utility is **SPD Process Profiler**. Highlight the **SPD Process Profiler** utility to display the process information table, which contains performance summary statistics. Each row in the table provides information about a process that was spawned by the SPD Server.



Profile Port: 6104 Host: sunburn

ProcName	User	PID	ThdCnt	RSS	Size	RealTime	CPU-Time	IO-Bytes	Pri	Nice	MNic
spdsserv	539	4082	6	6352K	14M	10:49:03	0:00:01		52	22	2
spdssnet	539	4085	4	5264K	13M	10:49:01			22	22	2
spdsbase	539	4087	6	5912K	15M	10:48:51			46	22	2
spdslog	539	4084	4	2376K	2864K	10:49:03			52	22	2

Profiler Connected: hostname:6104

The SPD Process Profiler displays information about memory and resource allocations. For this reason, you can use the SAS Management Console to review which server processes are occupying host computing resources; how the resources are distributed across users and processes at a given point in time; and whether the resource uses and distributions are appropriate for your computing environment.

You can do more than display performance summary statistics in the SAS Management Console application. You can also configure the performance server when you launch it to create text log files that can be saved locally on the SPD Server host machine. SPD Server is shipped with a PERL utility called process_perf_log that can parse the log that the performance server created and a SAS program that can be used to convert the log to an SPD Server table.

Starting SPD Server Performance Server

The performance server is started by running the rc.perf script. SPD Server is shipped with a partially configured rc.perf script. This rc.perf script is in your SPD Server installation folders at `.../samples/rc.perf`. Before running the script, modify the rc.perf script as follows:

1. Modify the NSPORT assignment to be the NSPORT assignment in your rc.spds start-up script.
2. Modify the SNPORT assignment to be the SNPORT assignment in your rc.spds start-up script.
3. Modify the PFPOPT assignment to be an available listen port to your spdsperf server for the SPD Server SAS Management Console client to connect to.
4. The script uses the -PARGS setting to specify how many times the performance server should capture performance information snapshots before shutting down. The

rc.spds script specifies 0, which indicates an infinite number of performance information captures. If you do not change the default number of information captures from 0 (infinity), consider modifying your rc.killspds script to shut down the rc.perf process when you shut down SPD Server.

Then, run rc.perf from PROC SPDO by using the SPDSCMD command option. Here is an example:

```
libname special sasspds 'public' host="hostname" service="5400" user="spdsadm"
  passwd="spdsadmpw" aclspecial=yes;

  %let tmp=path-to-domain-definitions/public;
  %let site=path-to-server-install;
  %let log=path-to-server-install/logs/

   proc spdo lib=special;
     spdscmd "&site/rc.perf -log &tmp/spdsperf.out";
  quit;
```

Performance Server Log File

It is recommended that you configure the performance server to save the process performance information to a text log file. To create this log file, include the rc.perf -log option and the pathname where you want to create the file when you start the performance server.

Your SPD Server installation includes a PERL utility called process_perf_log that is also in the **`.../samples`** directory of your SPD Server installation. When you use the process_perf_log PERL script with your SPD Server log files, the files are parsed and formatted for use as an input file for SAS processing.

Your SPD Server installation also includes a sample SAS program for importing parsed log file data to an SPD Server table. This program, PerfDataSamples.sas, is found in the **`.../samples`** directory as well.

Converting Your spdsperf Log into a Table

Follow these steps to convert your spdsperf log file into an SPD Server table.

1. Execute the process_perf_log utility on your log file to create a formatted input file. The syntax for the utility is as follows:

   ```
   ../samples/process_perf_log src-path log-path dst-path
   ```

   Here are the conditions:

   src-path
   : the name of the log file that is created when you run rc.perf with the -log option.

   log-path
   : the path to your SPD Server Installation log directory.

   dst-path
   : the full path to the location where the formatted SAS input file is to be created.

Here’s a sample command:

```
proc spdo lib=special;
     spdscmd "&site/process_perf_log &tmp/spdsperf.out &log &tmp/spdsperf.log";
  quit;
```

2. Open PerfDataSample.sas in your SAS session and modify the program as follows:
   - Assign a libref to the SPD Server domain where you want the output table to be created (Special).
   - Modify the INFILE statement to specify the full path to the input file created by process_perf_log (&tmp/spdsperf.log).
3. Run PerfdataSample.sas to create the table.

You can use PROC CONTENTS to describe the output table and PROC PRINT to see its contents. You can also use the table in SAS jobs.

Chapter 23
Managing SPDSBASE Processes

About SPD Server SPDSBASE Processes **205**

SPD Server User Proxy **205**

SPD Server SQL Proxy **206**

SPD Server SQL User Proxy **206**

Enhanced SPDSBASE Sharing **206**
Configuring Enhanced SPDSBASE Sharing 206
Managing Concurrent SPDSBASE Process Resources 207

About SPD Server SPDSBASE Processes

The SPDSBASE process is responsible for accessing or creating SPD resources for users. Several SPDSBASE processes can be active simultaneously in an SPD Server installation, handling work requests for different users or different SAS sessions. The SPDSBASE process can take on the role of either an SPD Server user proxy, an SPD Server SQL proxy, or an SPD Server SQL user proxy.

SPD Server User Proxy

The SPD Server user proxy is created by the SPD Server client, via the SASSPDS engine LIBNAME statement. The user proxy has the credentials of the LIBNAME statement, which include:

- the user’s SPD Server user ID
- the user’s SPD Server group name
- the user’s ACLSPECIAL privileges

If there are multiple librefs in the same SAS session that have the same LIBNAME credentials, the librefs share the same user proxy. Any SPD Server resources (such as tables and views) that are accessed by the user proxy on behalf of a SAS procedure are granted access based on the credentials of the user proxy.

SPD Server SQL Proxy

The SPD Server SQL proxy plans and executes SQL queries from SAS PROC SQL for implicit or explicit SQL statements. The SASSPDS engine can respond in two ways. The SASSPDS engine can request that the SPD Server host create an SQL proxy, or the SASSPDS engine can use an existing user proxy as the SQL proxy. (The user credentials of the SQL connection must be same as the credentials for the user proxy for this to happen.) The SASSPDS engine then sends implicit or explicit SQL statements to the SQL proxy on behalf of PROC SQL.

SPD Server SQL User Proxy

SQL queries given to the SQL proxy can either create or query SPD resources. Complex SQL queries can query resources from multiple librefs that have different user credentials. As a result, the SQL proxy creates its own user proxy, called the SQL user proxy, to access resources for the SQL query.

SQL user proxies have either credentials for IP=YES librefs for implicit SQL, or credentials for LIBGEN=YES librefs for explicit SQL. The credentials include the following:

- the user’s SPD Server user ID
- the user’s SPD Server group name
- the user’s ACLSPECIAL privileges

If there are multiple IP=YES or LIBGEN=YES librefs that have the same user credentials, the librefs share the same SQL user proxy. Any SPD Server resources (such as tables and views) that are accessed by the SQL proxy are authorized access based on the credentials of the SQL user proxy that is currently servicing that LIBNAME.

Enhanced SPDSBASE Sharing

Configuring Enhanced SPDSBASE Sharing

You specify SPDSBASE enhanced user proxy sharing settings via the SHARE= LIBNAME option, or via the SHRUSRPRXY spdsserv.parm parameter file option.

SHRUSRPRXY spdsserv.parm Option

SHRUSRPRXY enables enhanced sharing of user proxies and defines this as the default behavior for SPD Server. SPD Server proxy sharing enables users with different SPD Server user and group credentials to use the same proxy. Enhanced user proxy sharing keeps the number of concurrent SPDSBASE process resources from growing too large. A large number of concurrent SPDSBASE processes can create system resource allocation issues in some SPD Server environments. The SHRUSRPRXY spdsserv.parm option setting can be overridden by the SHARE= SASSPDS engine LIBNAME option.

Syntax

[NO]SHRUSRPRXY

Default

SHRUSRPRXY

SHARE= LIBNAME Option

When SHRUSRPRXY is set in the spdsserv.parm file, the SHARE= LIBNAME option can disable enhanced sharing of user proxies. When NOSHRUSRPRXY, the LIBNAME option can enable it.

Syntax

SHARE= YES | NO

Default

If the SHARE= LIBNAME option is not specified, SPD Server enhanced user proxy sharing settings default to the configuration defined by the SHRUSRPRXY setting in the spdsserv.parm parameter file.

Managing Concurrent SPDSBASE Process Resources

The following situations can cause a large number of SPDSBASE processes to be created during a SAS session:

- Jobs that use multiple LIBNAME statements with different user credentials.
- A user who is a member of multiple groups requires a separate LIBNAME statement for each group credentials. The result is a separate user proxy for each libref. SPD Server supports up to 32 groups for each user, but you cannot specify more than one group in a LIBNAME statement.
- Librefs with different credentials require their own SQL user proxy. The SQL proxy assigns all librefs that have LIBGEN=YES options for explicit SQL, or librefs that are configured via the IP=YES option setting for implicit pass-through SQL to an SQL user proxy.

When you use the SAS Java proxy to access the Hadoop cluster, it requires additional SPD Server process resources, as well as additional start-up requirements for the user proxy. For more information, see *SAS Scalable Performance Data Server: Processing Data in Hadoop*.

You can address resource allocation issues by enabling enhanced sharing of SPD Server user proxies. Enhanced SPD Server proxy sharing does not require the following matching credentials:

- matching SPD Server User ID credentials
- matching SPD Server Group credentials
- matching SPD Server ACLSPECIAL credentials

This means that almost all librefs in the same SAS session are able to share the same user proxy. The same is also true for any IP=YES or LIBGEN=YES librefs that are generated by the SQL proxy. Almost all of those librefs can share the same SQL user proxy. Shared proxies contain a control thread for each libref assigned to the user proxy. The control thread contains the user attributes of the libref, which are used to authenticate any resources accessed by that libref.

Enhanced proxy sharing is recommended if you are experiencing resource allocation issues due to a large number of concurrent spdsbase processes. However, turning the feature off is beneficial in the following situations:

- If the user proxy terminates unexpectedly due to a failure, turning it off reduces the number of librefs affected.
- Turning it off enables you to kill the user proxies for a given job.

For example, if a user submits a job that is consuming too many computing resources and enhanced user proxy sharing is not enabled, you could kill the user proxies for only that user job. However, killing all the user proxies for the SAS session, rather than for that particular job, is easier if enhanced user proxy sharing is enabled.

Part 6

System Management

Chapter 24
Maintaining SPD Server *211*

Chapter 25
Password Database Utility *213*

Chapter 26
SPDO Procedure *227*

Chapter 27
Backing Up and Restoring SPD Server Data *269*

Chapter 28
Index Utility *289*

Chapter 29
Domain List Utility *299*

Chapter 30
Directory Cleanup Utility *305*

Chapter 31
Debugging Tools *313*

Chapter 24

Maintaining SPD Server

Maintaining Server Tables **211**
Maintaining Server Indexes **211**
Overview of SPD Server Maintenance Tools **211**

Maintaining Server Tables

SPD Server tables do not reuse space. When an SQL statement to delete one or more rows from a table is issued, the row is marked deleted and the space is not reused. You must copy the table in order to recapture the space.

Maintaining Server Indexes

When a table is modified as the result of an append or update, all SPD Server indexes on the table are updated. When the index is updated, the per-value segment lists can potentially fragment or some disk space might be wasted. A highly fragmented server index can negatively impact the performance of queries that use the index. You can reorganize server indexes to eliminate fragmentation and reclaim wasted disk space by using the ixutil utility program. For more information, see Chapter 28, “Index Utility,” on page 289.

Overview of SPD Server Maintenance Tools

SPD Server provides the following tools to help you manage SPD Server users and data:

SPDO Procedure
: The SPDO procedure is the operator interface for SPD Server. Available only on systems where SAS is installed, this procedure can be used to define and manage SPD Server Access Control Lists (ACLs), define row-level security for tables, manage proxies, define and manage cluster tables, refresh server parameters and server domains, perform table management functions such as truncating tables, and execute SPD Server utilities from a central point. For more information, see Chapter 26, “SPDO Procedure,” on page 227.

Password Database Utility
The password database utility psmgr creates and manages the password database that enables access to SPD Server. Use this facility to define and manage users, groups, and group memberships, as well as to change user privileges and performance levels.

Table List Utility
The table list utility spdsls lists the contents of an SPD Server domain directory, or lists all other component files for a given SPD Server table component file. This utility can also be used to learn the size of a server table. For more information, see Chapter 29, “Domain List Utility,” on page 299.

Index Utility
The index utility ixutil enables you to reorganize an SPD Server hybrid index to improve query performance and minimize disk space. The utility also prints the disk usage statistics and the contents of indexes, and enables you to create, list, and delete join indexes. For more information, see Chapter 28, “Index Utility,” on page 289.

Backup and Restore Utilities
The standard file system backup and restore facilities that native operating systems provide are generally inadequate for backing up and restoring SPD Server tables. Server tables can be enormous in size, surpassing the file size limits maintained by some operating environments. The spdsbkup utility can be used to perform an initial full backup of server tables followed by incremental backups of changed files only. The spdsrstr utility restores server tables incrementally. For more information, see Chapter 27, “Backing Up and Restoring SPD Server Data,” on page 269.

Directory Cleanup Utility
The directory cleanup utility spdsclean can be used to perform routine maintenance functions on permanent, working, and temporary directories used by SPD Server. For more information, see Chapter 30, “Directory Cleanup Utility,” on page 305.

Debugging tools
See Chapter 31, “Debugging Tools ,” on page 313 for more information.

Chapter 25
Password Database Utility

Overview of the Password Database Utility 213

Invoking the psmgr Utility 214
Running psmgr Interactively 214
Running psmgr in Batch Mode 215

Enabling 32 Groups for SPD Server Users 216

Functional Listing of psmgr Commands 216

psmgr Commands 217
ADD Command 217
AUTHORIZE Command 219
CHGAUTH Command 219
CHGEXPIRE Command 220
CHGIP Command 220
CHGTIMEOUT Command 221
CHGPASS Command 221
CHGPERFCLASS Command 221
DELETE Command 222
EXPORT Command 222
GROUPDEF Command 222
GROUPDEL Command 223
GROUPMEM Command 223
GROUPS Command 224
HELP Command 224
IMPORT Command 224
LIST Command 225
RESET Command 225
QUIT Command 226

Overview of the Password Database Utility

The password database utility, psmgr, manages the password database that enables access to SPD Server. When you start SPD Server, the command line option -ACLDIR specifies the location (directory path) of the password database. The owner of the password database, typically the same user who started SPD Server, can update the database. The psmgr utility is located in the /bin directory of your installation.

The password database contains the following attributes and capabilities for each system user:

- a user ID
- a password
- an access privilege
- an optional IP address
- an optional password expiration time
- an optional ACL group name
- an optional time limit between successful logins
- an optional number of login failures that can occur before the user ID is disabled
- an optional user-performance class.

A user ID is restricted to eight characters and does not need to correspond to any system user ID. A password is also restricted to eight characters. All alphanumeric characters and the underscore symbol are acceptable for use in user IDs and passwords. A password must contain a minimum of six characters. At least one character must be numeric, and at least one character must be alphabetic. A new password must be different from the last six passwords for that user. The password cannot contain the user ID.

If a user has three consecutive failed attempts to connect to SPD Server, that user ID is no longer enabled. That user cannot connect to the SPD Server until an administrator resets the user ID.

Invoking the psmgr Utility

There are two ways that you can run the psmgr utility. You can run the utility interactively, or you can run it in batch mode with an input file.

Running psmgr Interactively

In interactive mode, psmgr will accept either a full command or a partial command. For example, if you enter **`add`**, psmgr will prompt you one line at a time for information to add the user: the user ID, their password (twice to verify it), privilege, expiration time, and so forth, until all necessary information to add a user is collected. You can also enter the first few arguments. For example, enter **`add smith abc123 abc123`**, and the utility will prompt you for values for the remaining arguments.

The commands and arguments are positional and when entered, must be separated by blank spaces. If you specify an insufficient number of arguments, the utility prompts you for the remaining arguments. Two commands, ADD and GROUPMEM, have optional arguments. An optional argument is an argument that is not required to add a user or a group membership to SPD Server. An optional argument either has a default value, if a value is not specified, or the argument represents a useful attribute that can be added later, or not at all. An example of an optional argument is the ADD command's *ip-address* argument.

Optional arguments are indicated in the syntax as follows:

```
[argument-value | - ]
```

When using psmgr interactively, you can avoid entering a value for an optional argument in one of two ways:

- If you are skipping a single argument, enter a carriage return in place of the argument value.
- If you are skipping multiple arguments, enter a hyphen or minus "–" sign for each argument value. SPD Server either skips or populates each argument with its default value, depending on the argument.

  For example, the following command adds user smith with password abc123 (twice), and privilege=0, and opts out of specifying other SPD Server user attributes for user smith.

  ```
  add smith abc123 abc123 0 - - - - - -
  ```

Password operands, which are obtained with a prompt, are not echoed back to the computer.

To invoke psmgr interactively on both UNIX and Windows, enter the following command:

```
psmgr <full-path-specification-to-password-table>
```

TIP To avoid having to specify the path to the password database:

- On Windows: Click the Windows **Start** button and select **All Programs** ⇨ **SAS** ⇨ **SAS SPD Server 5.3** ⇨ **Account Manager**. The SPD Server Account Manager is launched in a command window.
- On UNIX, switch to the /site directory and execute the pwdb script.

Running psmgr in Batch Mode

To run psmgr in batch mode, you submit a command file to the psmgr utility. Here is the content of a command file named Pscmds:

```
groupdef group1
groupdef group2
groupdef group3

add spduser1 newpwd1 newpwd1 0 - - group1 - - -
add spduser2 newpwd2 newpwd2 0 - - - - - -
groupmem spdsuser2 group1 group2 group3 !
```

The command file defines three groups, named group1 through group3, and then adds two users. spduser1 is given privilege=0 and is assigned group 1 as the primary group. All other user attributes are left undefined. Spdsuser2 is given privilege=0. All other user attributes are left undefined. The GROUPMEM command is used to assign spduser2 membership to group1, group2, and group3.

In the input file, note the following:

- Command arguments are separated by spaces.
- Optional arguments are omitted by specifying a hyphen (-).
- Commands are specified on one line.

Submit the command file to the psmgr utility on both UNIX and Windows as follows:

```
psmgr path-to-password-database < pscmds
```

TIP On UNIX, to avoid specifying the path to the password database, switch to the / site directory and specify **`pwdb < pscmds`**

Enabling 32 Groups for SPD Server Users

SPD Server supports user membership in up to 32 groups. However, the default group limitation is five groups per SPD Server user, which suits the majority of SPD Server users. To invoke the ability to grant users as many as 32 group memberships, start the psmgr utility with the **`-ver 5`** switch. For example, enter the following:

```
psmgr path-to-password-database -ver 5
```

Functional Listing of psmgr Commands

Table 25.1 *Functional Listing of psmgr Commands*

Task	Command	Description
Add a user or group	ADD	Adds a new user to the password database.
	GROUPDEF	Creates a new ACL group entry in the password database.
Modify a user account	AUTHORIZE	Authorizes a user to modify the password database.
	CHGAUTH	Changes the authorization level for a user.
	CHGEXPIRE	Changes the expiration date for the specified user's password.
	CHGIP	Changes the IP address from which the user must connect to the SPD Server.
	CHGTIMEOUT	Changes the logon time-out date for a user's password.
	CHGPASS	Changes the password for a user to a permanent password.
	CHGPERFCLASS	Changes the performance class of a user.
	DELETE	Deletes a user ID from the password database.
	GROUPMEM	Updates the ACL group list for a user ID.

Task	Command	Description
Delete a group	GROUPDEL	Deletes an ACL group entry from the password database.
List information	GROUPS	Lists the ACL groups that are in the password database.
	HELP	Displays general or command-specific help for the psmgr utility.
	LIST	Lists the contents of the password database or information about a specific user.
Special tasks	EXPORT	Exports the current password database into a flat file.
	IMPORT	Imports user information from a flat file to the password database. The flat file must have been created with the EXPORT command.
	RESET	Resets a password for a user to a new temporary, one-time password.
	QUIT	Terminates the psmgr session.

psmgr Commands

ADD Command

Adds a new user to the password database.

Syntax

```
add <username> <passwd> <passwd> <privilege>
[<ip-addr>|-] [<expiretime>|-] [<group>|-]
[<timeout>|-] [<failures>|-] [<class>|-]
```

Arguments

<username>
an SPD Server user ID. The user ID is restricted to eight characters. All characters must be alphanumeric or underscores. The SPD Server user ID does not have to correspond to any system user ID, unless nonnative authentication is configured.

<passwd> <passwd>
the user's password, which is restricted to eight characters. The password database requires a password with a minimum of six characters. At least one character must be

numeric, and at least one character must be alphabetic. The argument is repeated to verify the password.

Note: This password expires after the first logon to SPD Server. The user must change the password by using either the NEWPASSWD= or the CHANGEPASS= LIBNAME option. Password changing techniques do not apply to users who rely on LDAP Authentication for SPD Server access.

<privilege>
an authorization level number in the range 0 to 7. The authorization level number assigns access privileges to the user.

The numbers 0–3 are equivalent. Use the numbers 0–3 to specify a regular, non-privileged user.

The numbers 4–7 are equivalent. Use the numbers 4–7 to specify a user with special privilege. Users with special privilege can update the password database and override any ACL restrictions on server tables.

Note: All users connect to SPD Server as regular users, regardless of their authorization level. Users with special privilege must specify the ACLSPECIAL=YES LIBNAME option to invoke their special access in the SAS session.

CAUTION:
Authorization levels 4–7 should be granted with care. Users who have authorization levels 4–7 can access all SPD Server data with the PROC SPDO SPDSCMD statement.

<ip-addr>
a numerical IP address. The IP address on which the SAS, ODBC, JDBC, or SQL client software is running must match the IP address that is entered in the password database. This argument restricts the user's access to SPD Server to the specified IP address. A hyphen (-) indicates that no IP address is specified.

Note: The IP address is not verified.

<expiretime>
the length of time, in days, after which the user must change his password. A hyphen (-) indicates that no password expiration time is being specified. The time is measured from the day that you add the user.

<group>
the default group for the user. A hyphen (-) indicates that no default group is being specified. If a group name is specified, the group definition must have been created by a previous "GROUPDEF Command" on page 222. You can change group affiliation by using the "GROUPMEM Command" on page 223.

<timeout>
the maximum amount of time that is allowed between successful logins before the account is no longer enabled. A hyphen (-) indicates that no time-out is being specified.

<failures>
the number of password failures. A hyphen (-) indicates that no failure limit is being specified. The value specifies the number of login failures allowed before the user is disabled. A disabled password can be re-enabled by the administrator using the "RESET Command" on page 225.

<class>
the performance class of the user. Valid values are in the range 1–3. The value specifies whether the user is in a Low (1), Medium (2), or High (3) performance

class. SPD Server can be configured to provide different server parameters, based on the user's performance class setting.

AUTHORIZE Command

Authorizes a user to modify the password database.

Syntax

```
authorize <username> <passwd>
```

Alias: AUTH

Arguments

<username>
an existing SPD Server user ID.

<passwd>
a valid user password.

Details

The password database is owned by the operating system ID of the psmgr executable. The owner of the password database is normally the only person with authorized access to the database.

The following conditions must apply if you want to allow another system user to update the password database using the psmgr utility:

- The user must have operating system rights to execute the psmgr utility.
- The user must have operating system Read and Write access to the password database.
- The user must have AUTHORIZE access to update the password database. This requires an SPD Server user ID that has a special privilege level of 4-7.

Example

Assume that you are an operating system user and your user account meets the preconditions of operating system rights to execute the psmgr utility. You also have operating system Read and Write access to the password database. Your SPD Server administrator has given you a special user ID of **`user=auxadmin, password=admin2`**. To enable authorization to update the password database, use the following command:

```
authorize auxadmin admin2
```

CHGAUTH Command

Changes the authorization level for a user.

Syntax

```
chgauth <username> <authlevel>
```

Arguments

<username>
an existing SPD Server user ID.

<authlevel>
an authorization level for the user, in the range 0–7. The authorization level number assigns access privileges to the user.

The numbers 0–3 are equivalent. Use the numbers 0–3 to specify a regular, non-privileged user.

The numbers 4–7 are equivalent. Use the numbers 4–7 to specify a user with special privilege. Users with special privilege can update the password database and override any ACL restrictions on server tables.

CHGEXPIRE Command

Changes the expiration date for the specified user's password. By default, a new user ID is created with an expired password.

Syntax

```
chgexpire <username> <exptime>
```

Arguments

<username>
an existing SPD Server user ID.

<exptime>
the length of time, in days, after which the user must change his password. The time is measured from the day that you change the expiration time.

CHGIP Command

Changes the IP address from which the user must connect to SPD Server. The IP address on which the SAS, ODBC, JDBC, or SQL client software is running must match the IP address that is entered in the password database.

Syntax

```
chgip <username> "<new-ip-address>"
```

Arguments

<username>
an existing SPD Server user ID.

"<new-ip-address>"
the new IP address from which the user must connect to the SPD Server host. The IP address must be specified numerically using the format *xxx.xxx.xxx.xxx*. The IP address is not verified. Invalid and incorrect IP addresses are noted as errors in the SPD Server log and will cause that user's future logon attempts to fail. The default value is blank.

CHGTIMEOUT Command

Changes the logon time-out date for a user's password.

Syntax

```
chgtimeout <username> <timeout>
```

Arguments

<username>
an existing SPD Server user ID.

<timeout>
a password logon time-out period, specified in days. The time-out period specifies the number of days that the account can be inactive before the password becomes invalid.

CHGPASS Command

Changes the password for a user to a permanent password.

Syntax

```
chgpass <username> <oldpwd> <newpwd> <newpwd>
```

Arguments

<username>
an existing SPD Server user ID.

<oldpwd>
the user's old password.

<newpwd>
the new password for the user. If you are prompted for the new password, you are prompted again to re-enter it for accuracy. The new password must be different from the last six passwords. The new password must also contain at least six characters, with at least one numeric character and with at least one alphabetic character. The password cannot contain the user ID.

CHGPERFCLASS Command

Changes the performance class of a user.

Syntax

```
chgperfclass <username> <class>
```

Arguments

<username>
an existing SPD Server user ID.

<class>
a performance class for the user, in the range 1–3. The value specifies whether the user is in a Low (1), Medium (2), or High (3) performance class. The SPD Server server can be configured to provide different server parameters, based on the user's performance class setting.

DELETE Command

Deletes a user from the password database.

Syntax

```
delete <username>!
```

Alias: DEL

Arguments

<username>
the user ID of an existing SPD Server user.

!
verifies that you intend to delete the user ID from the password database. If you do not specify !, you are prompted to verify the deletion.

EXPORT Command

Exports the current password database into a flat file.

Syntax

```
export <textfile>
```

Arguments

<textfile>
name of the flat file to create that will contain the contents of the current password database.

Details

The EXPORT command generates a single line in the flat file for each record in the password database. User passwords are encrypted in the table.

The contents of the flat file are a representation of what is stored in the password database. When you are making changes that affect many users, it might be easier to edit the flat file than to use the psmgr utility. After you make the changes in the file, you can use the IMPORT command to construct a new, modified password database.

GROUPDEF Command

Defines a new ACL group entry in the password database.

Syntax

```
groupdef <groupname>
```

Alias: GPDEF

Arguments

<groupname>
: the name of a new ACL group. The name must be unique and is restricted to eight characters. All characters must be alphanumeric or underscores. This argument verifies that the groups that are specified on the GROUPMEM command are valid.

GROUPDEL Command

Deletes an ACL group entry from the password database.

Syntax

```
groupdel <groupname> !
```

Alias: GPDEL

Arguments

<groupname>
: the name of an existing ACL group.

!
: verifies that you intend to delete the group from the password database. If you do not specify !, you are prompted to verify the deletion.

GROUPMEM Command

Updates the ACL group list for a user.

Syntax

```
groupmem username groupname | ! | - | " "
```

Alias: GPMEM

Arguments

<groupname>
: specify the name of an existing ACL group. If multiple groups are specified, separate each ACL group name with a space. The first ACL group name that you specify becomes the default ACL group for the user.

–
: specify a hyphen (-) as a group list argument to serve as a place holder for a group name that cannot be changed.

" "
: specify an empty quoted argument (" ") as a group list argument to clear the group in that position.

!
: specify an exclamation point (!) to indicate to SPD Server that there are no more groups in your group name declaration.

Details

The password database can be configured to support from 5 to 32 groups per user. The utility will prompt you for additional ACL group names, up to the number of groups that are configured at your site. In batch mode, the utility will expect you to enter the configured number of group names. To specify fewer than the configured number of group names, enter an exclamation point (!) to indicate to SPD Server that there are no more groups in your group name declaration.

To add names to the group list and leave existing group names unchanged, specify a hyphen (-) to indicate the groups in the group list that you want unchanged. To remove a group from the group list, specify the " " argument.

Example

The following example uses the GROUPMEM command to change the groups for user DSWider so that the user's previously assigned groups in the first, second, and fourth positions remain the same; the group in the user's third position is to be deleted; and the fifth group position is assigned to the group Fondo. The exclamation mark indicates to psmgr that no additional group memberships are to be declared for this user.

```
groupmem dswider - - "" - fondo !
```

GROUPS Command

Lists the ACL groups that are in the password database.

Syntax

```
groups
```

HELP Command

Displays general or command-specific help for the psmgr utility.

Syntax

```
help [<command>]
```

Alias: ?

Arguments

<command>
the name of a psmgr command. If you specify a command, a short description of the command is displayed. If you issue a HELP command without an operand, a list of all available psmgr commands is displayed.

IMPORT Command

Imports user information from a flat file to the password database. The flat file must have been created with the EXPORT command.

Syntax

```
import <textfile>
```

Arguments

<textfile>
the name of the flat file to import. This flat file contains the user definitions to add to the password database.

Details

The IMPORT command reads the flat file, interpreting each single line as a record in the password database. Typically, the flat file is created from a submitted EXPORT command that was issued on the same password database or another password database.

If the psmgr utility encounters an identical user ID in the password database during the import process, it skips the line. The psmgr utility displays a message that states that the line was skipped.

LIST Command

Lists the contents of the password database or information about a specific user.

Syntax

```
list [<username>]
```

Alias: LS

Arguments

<username>
an existing SPD Server user ID. If you do not specify a user ID, the entire password database is listed.

Example

This following is an example of the LIST command and its output:

```
list bar


USER AUTHORIZATION IP ADDRESS
---- ------------- -----------
 bar       7
```

RESET Command

Resets a password for a user to a new temporary, one-time password.

Syntax

```
reset <username> <newpwd> <newpwd>
```

Arguments

<username>
the user ID of an existing SPD Server user.

<newpwd>
a new password for the user. The new password can be up to eight characters in length. The new password must contain at least six characters. At least one character

must be numeric, and at least one character must be alphabetic. The argument is repeated to verify the password for accuracy.

Details

The RESET command can be used to reset a user's password after three consecutive failed attempts to connect to a server. After the third failed attempt, the user ID is no longer enabled. After the password has been reset, the user must change the password before connecting to a server by using either the NEWPASSWD= or the CHANGEPASSWD= LIBNAME option.

Example

This command resets the password for Tom:

```
reset tom abc123 abc123
```

QUIT Command

Terminates the psmgr session.

Syntax

```
quit
```

Alias: Q

Chapter 26

SPDO Procedure

Overview: SPDO Procedure **228**

Concepts: SPDO Procedure **229**
ACL Definition and Management 229
Row-Level Security Definition and Management 229
Proxy Management 229
Server Management 230
Cluster Table Definition and Management 230
General Table Management 230

Syntax: SPDO Procedure **230**
PROC SPDO Statement 231
ADD ACL Statement 232
CLUSTER ADD Statement 234
CLUSTER CREATE Statement 235
CLUSTER DESTROY Statement 237
CLUSTER FIX Statement 237
CLUSTER LIST Statement 238
CLUSTER MODIFY Statement 239
CLUSTER REMOVE Statement 239
CLUSTER REPLACE Statement 240
CLUSTER UNDO Statement 241
CONSTRAINT ADD Statement 242
CONSTRAINT DESCRIBE Statement 243
CONSTRAINT REMOVE Statement 243
DELETE ACL Statement 244
LIST ACL Statement 246
LIST USERS Statement 248
LIST USERS/LOCKING Statement 248
MODIFY ACL Statement 249
OPER Statement 252
REFRESH DOMAINS Statement 253
REFRESH PARMS Statement 253
SET ACLTYPE Statement 254
SET ACLUSER Statement 254
SET MODE OPER Statement 255
SET USER Statement 256
SET USER/LOCKING Statement 257
SHOWLIBNAME Statement 257
SPDSCMD Statement 258
SPDSMAC Statement 259
TRUNCATE Statement 260

Examples: SPDO Procedure **262**
Example 1: Using PROXY Statements to Get Information about Non-Locking SPD Server Users 262
Example 2: Using PROXY Statements to Get Information about Locking SPD Server Users 264
Example 3: Refreshing SPD Server Domains 266
Example 4: Using PROC SPDO to Back Up and Restore Tables 266
Example 5: Using PROC SPDO to Truncate a Table 267

Overview: SPDO Procedure

The SPDO procedure is the operator interface for SPD Server. It is a SAS procedure and runs only on systems where SAS is installed.

Operator tasks for the server fall into the following categories:

- ACL definition and management
- row-level security definition and management
- proxy management
- server management
- cluster table definition and management
- general table management.

Here are examples of tasks that you might perform in each category:

ACL definition and management:

- add, modify, list, and delete ACLs for server resources

Row-level security definition and management:

- define, describe, and remove WHERE constraints on tables

Proxy management:

- list all active users or all locking users
- set a user for the user proxy process or user locking proxy thread
- list the active librefs and open tables for the user
- set operator mode and issue privileged operator commands.

Server management:

- refresh domains and server parameters on a running server.

Cluster table definition and management:

- create, list, modify, destroy, and undo dynamic cluster tables
- add, remove, replace, and fix cluster table members.

General table management:

- issue system commands on server nodes
- truncate a table.

Many PROC SPDO statements require that the ACLSPECIAL= option is enabled in the SASSPDS engine LIBNAME statement. To enable ACLSPECIAL=, a server user ID must first have been granted special privilege in the password database. Then, the user can request special access for a specific server connection. For more information about SPD Server authorization levels, see Chapter 25, “Password Database Utility,” on page 213.

Concepts: SPDO Procedure

ACL Definition and Management

The statements for defining and managing ACLs are as follows:

- “ADD ACL Statement” on page 232
- “DELETE ACL Statement” on page 244
- “LIST ACL Statement” on page 246
- “MODIFY ACL Statement” on page 249.
- “SET ACLTYPE Statement” on page 254
- “SET ACLUSER Statement” on page 254.

Row-Level Security Definition and Management

The statements for defining and managing row-level security on tables are as follows:

- “CONSTRAINT ADD Statement” on page 242
- “CONSTRAINT DESCRIBE Statement” on page 243
- “CONSTRAINT REMOVE Statement” on page 243

Proxy Management

To issue proxy commands, you must first select a user proxy. PROC SPDO provides the following interactive statements to enable you to list and select from the available user proxies. Separate statements are available for non-locking and locking user proxies.

- “LIST USERS Statement” on page 248
- “SET USER Statement” on page 256
- “LIST USERS/LOCKING Statement” on page 248
- “SET USER/LOCKING Statement” on page 257
- “SHOWLIBNAME Statement” on page 257

The following interactive statements are provided to act on selected proxies:

- “SET MODE OPER Statement” on page 255
- “OPER Statement” on page 252.

Server Management

The server management statements are as follows:

- "REFRESH DOMAINS Statement" on page 253.
- "REFRESH PARMS Statement" on page 253.

Cluster Table Definition and Management

The statements for defining and managing cluster tables are as follows:

- "CLUSTER ADD Statement" on page 234
- "CLUSTER CREATE Statement" on page 235
- "CLUSTER DESTROY Statement" on page 237
- "CLUSTER FIX Statement" on page 237
- "CLUSTER LIST Statement" on page 238
- "CLUSTER MODIFY Statement" on page 239
- "CLUSTER REMOVE Statement" on page 239
- "CLUSTER REPLACE Statement" on page 240
- "CLUSTER UNDO Statement" on page 241

For more information about cluster tables, including examples of how the cluster management statements are used, see the *SAS Scalable Performance Data Server: User's Guide*.

General Table Management

The general table management statements are as follows:

- "SPDSCMD Statement" on page 258
- "TRUNCATE Statement" on page 260

Syntax: SPDO Procedure

Requirements: You must have special privilege to issue proxy management and server management statements.

You must be the resource owner, have ACL access, or have special privilege in order to issue ACL definition and management statements and table management statements.

PROC SPDO LIB=*libref*;

In alphabetical order, optional statements are:

ADD ACL *acl1*[, *acl2*...][C=*cat* T=*type*][/*options*];

CLUSTER ADD *cluster-tablename* MEMBER=*member-name1*[, MEMBER=*member-name2*...];

CLUSTER CREATE *cluster-tablename* MEMBER=*member-name1*[, MEMBER=*member-name2*...][*options*];

CLUSTER DESTROY *cluster-tablename*;

CLUSTER FIX *member-table-name*;

CLUSTER LIST *cluster-tablename* [/VERBOSE][OUT=*SAS-dataset-name*];

CLUSTER MODIFY *cluster-tablename* MINMAXVARLIST=(*var1 var2* ...);

CLUSTER REMOVE *cluster-tablename* MEMBER=*member-name1* [, MEMBER=*member-name2*...];

CLUSTER REPLACE *cluster-tablename* OLDMEMBER=*member-name* NEWMEMBER=*member-name*;

CLUSTER UNDO *cluster- tablename*;

CONSTRAINT ADD *table-name*; where *where-clause-expression*;

CONSTRAINT DESCRIBE *table-name*;

CONSTRAINT REMOVE *table-name*;

DELETE ACL [*options*];

LIST ACL [*options*];

LIST USERS;

LIST USERS/LOCKING;

MODIFY ACL [*options*];

OPER [*options*];

QUIT;

REFRESH DOMAINS;

REFRESH PARMS;

SET ACLTYPE *type*;

SET ACLUSER *user*;

SET MODE OPER;

SET USER *user-ID* [*port-number*];

SET USER/LOCKING [*user-ID* threadID=*number*];

SHOWLIBNAME [*options*];

SPDSCMD '*command*';

TRUNCATE *table-name*;

PROC SPDO Statement

Invokes the operator interface for SPD Server.

Syntax

PROC SPDO LIBRARY=*libref*

Required Argument

LIBRARY=
specifies the libref that represents the SPD Server domain whose files you want to manage. The LIBNAME statement that assigned the libref must specify SPD Server connection parameters. For more information, see "SPD Server LIBNAME Statement" in *SAS Scalable Performance Data Server: User's Guide*.

Alias LIB=

ADD ACL Statement

Creates new ACL entries.

Requirement:	You must be the resource owner, have ACL access to a resource, or have special privilege in order to add an ACL to a resource. When using special privilege, set the ACLSPECIAL= option in the LIBNAME statement.
Interactions:	Before using ADD ACL, you must set the context for the request with the SET ACLUSER statement. For more information, see "SET ACLUSER Statement" on page 254. If you are creating an ACL for a resource other than a domain or a table, set the ACL type before using ADD ACL. For more information, see "SET ACLTYPE Statement" on page 254.
See:	For detailed information about how the server supports ACLs, see "ACL Security Model" on page 144.

Syntax

ADD ACL *acl1*[*acl2*...] [C=*cat* T=*type*] [/*options*];

Optional Arguments

acl
specifies the name of the ACL. Use a one-part name to identify resources except domains and table columns. You do not need to specify a name when creating a domain ACL. The domain is inferred from the libref. Use a two-part name (*table.column*) to identify an ACL for a table column. You can specify multiple names. Each name creates a separate ACL entry.

ALTER
grants universal ALTER access to the resource.

C=*cat*
identifies the specified ACL names as the names of catalog entries in the catalog *cat*. You pair this value with the T= option.

/GENERIC
specifies that the ACL name is a generic ACL.

Note If you specify /GENERIC when defining a table column ACL, the /GENERIC applies to the table name, not to the column name.

GROUPALTER
grants group ALTER access to the resource.

GROUPREAD
grants group READ access to the resource.

GROUPWRITE
grants group WRITE access to the resource.

/LIBNAME
: creates an ACL for the domain. You can control access permissions to an entire domain with this option.

MODEL
: specifies the name of another ACL. This option requests the software to copy all the access permissions and access list entries from this ACL.

/PERSIST
: specifies that the ACL (or ACLs) is a persistent ACL. A persistent ACL entry is an ACL that is not removed from the ACL tables when the resource is deleted.

READ
: grants universal READ access to the resource.

T=*type*
: identifies the catalog entry type to associate with the specified ACLs names. This option is required when you specify the C= option.

WRITE
: grants universal WRITE access to the resource.

Details

SPD Server uses access control lists (ACLs) to secure resources. ACL permissions affect all server resources, including domains, tables, table columns, catalogs, catalog entries, and utility files. By default, only the owner (creator) of a resource has access to a resource. Resource owners can grant others access to their resources, including their ACLs. Resource owners can grant ACL permissions to specific users, to specific groups of users (called an ACL group), or to all SPD Server users or all groups (universal permissions). Users with special privilege can also modify ACLs. For more detailed information about how the server supports ACLs, see “ACL Security Model” on page 144.

The examples below show the ADD ACL statements that are necessary to define various types of ACLs. For examples of how ACLs might be used to secure resources at a customer site, see “ACL Examples” on page 148.

Examples

Example 1: Add Domain ACL

This creates a domain ACL that grants universal READ and group WRITE access.

```
add acl/LIBNAME
read
groupwrite;
```

Example 2: Add Resource ACL

This ACL for the resource MINE_AUG2016 grants universal READ and WRITE access.

```
add acl mine_aug2016/read write;
```

Example 3: Add Generic ACL

This generic ACL for MINE* grants universal READ access.

```
add acl mine/generic read;
```

Example 4: Add Column ACL

This ACL for the column MINE_AUG2016.SALARY grants group READ access and denies access to all others.

```
add acl mine_aug2016.salary/groupread;
```

Example 5: Add Generic Column ACL

This ACL for the column MINE*.SALARY grants group READ access and denies access to all others.

```
add acl mine.salary/generic
groupread;
```

Example 6: Add Catalog ACL

This ACL for the MYCAT catalog grants universal READ and group READ/WRITE access.

```
set acltype catalog;
add acl mycat/read
groupread
groupwrite;
```

Example 7: Add Generic ACL for Catalog Entries

This ACL for catalog entries, MYCAT.MY*.CATAMS, grants universal READ and group READ access.

```
set acltype catalog;
add acl my
c=mycat
t=catams/generic
read
groupread;
```

CLUSTER ADD Statement

Adds a member to an existing cluster table.

Requirements: You must have Control access to both the table and the cluster in order to add the table to the cluster.

The candidate table must be in the same domain as the cluster and have the same structure as other tables in the cluster. See "Member Table Requirements for Creating Dynamic Cluster Tables" in *SAS Scalable Performance Data Server: User's Guide*.

Syntax

CLUSTER ADD *cluster-tablename* MEMBER=*member-name1*[, MEMBER=*member-name2*...];

Arguments

cluster-tablename
specifies the name of an existing cluster table.

MEMBER=*member-name*
specifies the name of one or more new member tables. To specify multiple tables, use multiple MEMBER= statements. The new member tables are appended to the end of the specified cluster table's member list.

Alias	MEM=

Details

The CLUSTER ADD statement is used to add members to a cluster table that was previously created with the CLUSTER CREATE statement. For more information, see "CLUSTER CREATE Statement" on page 235.

When a member table is added to a cluster, users that currently have that cluster open for reading will not see the new member. Changes to the cluster are not reflected until the cluster is reopened or opened after the CLUSTER ADD processing completes.

Example

This CLUSTER ADD example adds two member tables to the cluster named Sales_History.

```
proc spdo library=libref;
cluster add Sales_History
mem=sales201607
mem=sales201608
quit;
```

CLUSTER CREATE Statement

Creates a cluster table.

Requirements:	You must have Control access to all candidate member tables. All of the tables must be in the same domain and have the same structure. See "Member Table Requirements for Creating Dynamic Cluster Tables" in *SAS Scalable Performance Data Server: User's Guide*.

Syntax

CLUSTER CREATE *cluster-tablename* MEMBER=*member-name1*
[MEMBER=*member-name2*...][*options*];

Required Arguments

cluster-tablename
specifies the name of the cluster to be created.

MEMBER=*member-name*
specifies the name of a member table. To specify multiple tables, use multiple MEMBER= statements.

Alias	MEM=

Optional Arguments

DELETE= YES | NO
specifies whether the cluster and all its members can be permanently deleted with the CLUSTER DESTROY statement. The default is NO.

MAXSLOT= *number*
specifies the maximum number of slots, or member tables, to be allocated for the cluster. The default setting for the MAXSLOT= parameter is -1. This value configures the server to permit dynamic growth of the number of member tables in a cluster, up to the specified system maximum value. The system maximum value for the number of slots is specified by the MAXGENNUM parameter in the spdsserv.parm configuration file. If there is a known maximum number of slots to be enforced for a particular cluster table, it is more efficient to specify the limitation using the MAXSLOT= argument.

UNIQUEINDEX=YES|NO
specifies whether the unique indexes that are defined in the member tables will be validated and marked as unique in the cluster table. The default setting is YES.

Note: The processing that is required to validate the unique indexes depends on the number of rows in the tables. Processing can take a considerable amount of time for larger tables. If you choose to use the validation process but the indexes are not unique, the CLUSTER CREATE statement fails.

Details

SPD Server supports virtual table structures called cluster tables. A cluster table is a table that consists of numerous similar tables called members. In order to qualify as a member of a cluster table, the tables must share the same table, variable, and index attributes. They must also have the same organization. Cluster tables use the attributes to manage the data that is contained in the members. The cluster table structure provides architecture that enables flexible loading and rapid storage and processing for very large data tables. Cluster tables provide organizational features and performance benefits that traditional SAS tables and regular SPD Server tables do not. For more information about the benefits of and requirements for cluster tables, see “Creating and Using Dynamic Cluster Tables” in *SAS Scalable Performance Data Server: User’s Guide*.

Example

This CLUSTER CREATE example creates a cluster named Sales_History, which contains member tables sales201601, sales201602, sales201603, sales201604, and sales201606. The cluster table records sales results for the first six months of 2016. The cluster has a limit of 24 member tables.

```
proc spdo library=libref;
cluster create Sales_History
mem=sales201601
mem=sales201602
mem=sales201603
mem=sales201604
mem=sales201605
mem=sales201606;
maxslot=24;
```

CLUSTER DESTROY Statement

Deletes an existing cluster table and all of its members.

Requirement:	You must have Control access to a cluster in order to destroy the cluster.
Interaction:	The CLUSTER DESTROY statement is valid only when used on clusters that were created with the DELETE=YES option.
See:	For detailed information about cluster tables, see “Dynamic Cluster Tables” in *SAS Scalable Performance Data Server: User’s Guide*.

Syntax

CLUSTER DESTROY *cluster-tablename*;

Argument

cluster-tablename
specifies the name of the cluster table to be deleted.

Details

To simply revert a cluster table back to its unbound tables, use the CLUSTER UNDO statement. See “CLUSTER UNDO Statement” on page 241.

CLUSTER FIX Statement

Restores removed or replaced members of a cluster table to a writable state.

See:	For detailed information about cluster tables, see “Creating and Using Dynamic Cluster Tables” in *SAS Scalable Performance Data Server: User’s Guide*.

Syntax

CLUSTER FIX *member-table-name*;

Argument

member-table-name
specifies the name of the member table that you want to repair.

Details

When a cluster member table is removed with the CLUSTER REMOVE statement or replaced with the CLUSTER REPLACE statement, the table becomes visible as a separate table, but the table remains in a read-only state. The CLUSTER FIX statement restores the member table to a writable state.

CLUSTER LIST Statement

Lists information about the members in a cluster table.

See: For detailed information about cluster tables, see “Dynamic Cluster Tables” in *SAS Scalable Performance Data Server: User’s Guide*.

Syntax

CLUSTER LIST *cluster-tablename* [/VERBOSE] [OUT=*SAS-data-set-name*];

Required Argument

cluster-tablename
specifies the name of an existing cluster table.

Optional Arguments

OUT= *SAS-data-set-name*
(optional) specifies the name of an output SAS data set to store the output. Specify the data set name in this form:

```
libref.data-set-name
```

In the specification, *libref* is a Base engine libref. When the OUT= option is omitted, the CLUSTER LIST statement writes output to STDOUT.

The output data set has two columns, named Cluster Name and Member Name, unless the CLUSTER LIST statement includes the /VERBOSE option.

/VERBOSE
specifies to write detail information about each member table. When OUT= is also specified, it adds columns named Variable Name, Minimum Value, and Maximum Value to the OUT= data set. When the OUT= option is omitted, the CLUSTER LIST statement writes output to STDOUT.

Interaction The /VERBOSE argument must precede the OUT= argument in the CLUSTER LIST command.

Details

Apart from identifying the members in a cluster table, CLUSTER LIST can be used to determine the member number of a particular table for use with the MEMNUM= table option. The MEMNUM= table option limits a query or Read operation to a specific table in the cluster.

Examples

Example 1

This CLUSTER LIST statement specifies to list the members in cluster table Sales_History in the SAS log:

```
cluster list Sales_History;
```

Example 2

This CLUSTER LIST statement specifies to write detail information about the members of cluster table Sales_History to a SAS data set named MyLib.Outfile.

```
cluster list Sales_History /verbose out=mylib.outfile;
```

CLUSTER MODIFY Statement

Modifies an existing cluster table's metadata to include minimum and maximum values for one or more columns.

Requirement: You must have Control access and exclusive access to the cluster table in order to execute the CLUSTER MODIFY statement.

See: For detailed information about cluster tables, see "Creating and Using Dynamic Cluster Tables" in *SAS Scalable Performance Data Server: User's Guide*.

Syntax

CLUSTER MODIFY *cluster-tablename* MINMAXVARLIST=(*var1 var2 ...*);

Arguments

cluster-tablename
specifies the name of an existing cluster table.

MINMAXVARLIST= (*var1 var2* ...)
specifies the name of one or more columns whose minimum and maximum values you want to add to metadata. The column names that you specify must exist in the cluster tables. The columns must not have a previous MINMAXVARLIST setting.

Details

SPD Server permits you to store the maximum and minimum values for a column in the metadata. The CLUSTER MODIFY statement enables you to add this metadata.

When the server runs the CLUSTER MODIFY statement, it unclusters the cluster table and makes the column modifications to the individual member tables. It then re-creates the cluster table after the column modifications have completed. If an error occurs while the CLUSTER MODIFY statement is running, the cluster table can only be re-created manually by issuing the CLUSTER CREATE statement.

The server performs a full table scan to initialize the MINMAXVARLIST values in each member table. As a result, the processing time required for the CLUSTER MODIFY statement is directly related to the sizes of the tables that belong to the cluster table.

CLUSTER REMOVE Statement

Removes members from a cluster table.

Requirement: You must have Control access to the cluster table in order to remove a member table.

Syntax

CLUSTER REMOVE *cluster-tablename* MEMBER=*member-name1*
[, MEMBER=*member-name2*...];

Arguments

cluster-tablename
specifies the name of an existing cluster table.

member-name
specifies the name of one or more member tables to remove from the cluster. To specify multiple tables, use multiple MEMBER= statements.

Alias	MEM=

Details

When a cluster member table is removed or replaced, users that currently have that particular cluster open for Read access will not see the change until a subsequent reopen or open of the table after the remove command has completed.

A cluster member table that has been removed from a cluster becomes visible as a separate table, but the table is in a Read-only state. If there is a need to update the table that was removed, use the CLUSTER FIX statement to restore the table to a writable state. For more information, see “CLUSTER FIX Statement” on page 237.

Example

This CLUSTER REMOVE statement removes member tables that contain sales information for the second half of 2014 from the cluster table Sales_History.

```
cluster remove sales_history
mem=sales201407
mem=sales201408
mem=sales201409
mem=sales201410
mem=sales201411
mem=sales201412';
```

CLUSTER REPLACE Statement

Specifies a replacement member for a member of a cluster table.

Requirements: You must have Control access to the cluster table in order to replace a table in a cluster.

The target cluster must either have been created with UNIQUEINDEX=NO or there must not be any unique indexes defined on the cluster.

Syntax

CLUSTER REPLACE *cluster-tablename* OLDMEMBER=*member-name*
NEWMEMBER=*member-name*;

Arguments

cluster-tablename
specifies the name of an existing cluster table.

OLDMEMBER=*member-name*
specifies the name of the member table to remove from the member list.

Alias OLDMEM=

NEWMEMBER=*member-name*
specifies the name of the new member table to replace the old table with.

Details

The CLUSTER REPLACE statement specifies a replacement member table for a single member table in a cluster. The new member table is inserted in the same slot (or cluster position) as the table that was removed.

CLUSTER REPLACE does not validate unique indexes. Therefore, it will not work if a cluster has UNIQUEINDEX=YES defined, which is the default setting for CLUSTER CREATE. You can do a CLUSTER REMOVE followed by a CLUSTER ADD to replace a member in a cluster that was created with unique indexes. The CLUSTER ADD operation will validate unique indexes. CLUSTER ADD appends new tables to the end of the cluster member list.

When CLUSTER REPLACE fails, the SAS macro variable SYSERR is set to an appropriate value. When the CLUSTER REPLACE command succeeds, SYSERR is set to 0 (zero).

When a cluster member table is replaced in the cluster, users that currently have that particular cluster open for Read access will not see the change until a subsequent reopen or open of the table after the replace command has completed. The replaced table becomes visible as a separate table, but the table is in a Read-only state. If there is a need to update the table, use the CLUSTER FIX statement to restore the table to a writable state. For more information, see "CLUSTER FIX Statement" on page 237.

Example

This CLUSTER REPLACE statement replaces member table sales201601 with new member table sales201601 in cluster table Sales_History's member list.

```
cluster replace sales_history
oldmem=sales201601 newmem=sales201601;
```

CLUSTER UNDO Statement

Reverts a cluster table back to its unbound tables.

Requirement: You must have Control access to the cluster table to undo it.

Syntax

CLUSTER UNDO *cluster- table-name*;

Arguments

cluster-tablename
specifies the name of an existing cluster table to undo.

Details

Use CLUSTER UNDO when you want to destroy the metadata that is associated with a cluster, but want to keep the source tables.

Example

This CLUSTER UNDO statement reverts the cluster Sales_History back to its original tables. Undoing the cluster frees you to create a new cluster table named Sales_History with a new set of member tables. Meanwhile, the original tables continue to exist.

```
proc spdo library=libref;
   cluster undo Sales_History;
quit;
```

CONSTRAINT ADD Statement

Adds a table WHERE constraint to a table.

Requirements: You must have Control permission to add a WHERE constraint on a table.
You must set the context for the request with the SET ACLUSER statement.

Syntax

CONSTRAINT ADD *table-name*; where *where-clause-expression*;

Arguments

table-name
specifies the name of the table on which to apply the table WHERE constraint.

where-clause-expression
specifies a WHERE clause. A table constraint WHERE clause can be any WHERE clause that the server can parse and interpret. The WHERE clause can use any function that SPD Server supports. Generally, the WHERE clause uses symbolic substitution to create row-level security that is based on the values for user ID, group, or special privilege.

Details

Row-level security is created by defining WHERE constraints on tables that specify symbolic substitutions. The symbolic substitutions filter table rows based on the values for user ID, group ID, or special privilege. For information to create the WHERE clauses, see Chapter 15, “Defining WHERE Constraints on Tables,” on page 169. For information about symbolic substitution, see “Symbolic Substitution” on page 173.

Before applying a WHERE constraint to a table, note that the SPD Server SQL extensions COPY TABLE and UPDATE TABLE do not function on a table that has a table constraint WHERE clause. If you try to use COPY TABLE or UPDATE TABLE, the destination table contains rows that were filtered by the WHERE constraints on the source table. The destination table does not inherit the WHERE constraints of the source table. To retain the WHERE clause, you must use a different method to copy the table, such as PROC COPY or SQL CREATE TABLE as SELECT. Table owners should grant Update access only to users who can modify any row of the table, delete any row of the table, or append new rows to the table.

Table WHERE constraints do not affect normal table ACL settings, such as Read, Write, and Column access.

Example

For an example of how the CONSTRAINT ADD statement is used, see "Example Table Constraint WHERE Clauses" on page 170.

CONSTRAINT DESCRIBE Statement

Describes the table WHERE constraint on a table.

Requirements: You must have Read permission to describe a WHERE constraint on a table.

You must set the context for the request with the SET ACLUSER statement.

Syntax

CONSTRAINT DESCRIBE *table-name*;

Arguments

table-name
specifies the name of the table whose table WHERE constraint you want to describe.

Example

For an example of how the CONSTRAINT DESCRIBE statement is used, see "Example Table Constraint WHERE Clauses" on page 170.

CONSTRAINT REMOVE Statement

Removes a table WHERE constraint from a table.

Requirements: You must have Control permission to remove a WHERE constraint from a table.

You must set the context for the request with the SET ACLUSER statement.

Syntax

CONSTRAINT REMOVE *table-name*;

Arguments

table-name
specifies the name of the table whose table WHERE constraint you want to remove.

Example

For an example of how the CONSTRAINT REMOVE statement is used, see "Example Table Constraint WHERE Clauses" on page 170.

DELETE ACL Statement

Deletes existing ACLs for resources.

Requirement: You must be the resource owner, have ACL access to a resource, or have special privilege in order to delete an ACL. When using special privilege, set the ACLSPECIAL= option in the LIBNAME statement.

Interactions: Before using DELETE ACL, you must set the context for the request with the SET ACLUSER statement. The user name that is specified in SET ACLUSER must be that of the ACL owner. For more information, see "SET ACLUSER Statement" on page 254.

If you are deleting an ACL for a resource other than a domain or a table, set the ACL type before using DELETE ACL. For more information, see "SET ACLTYPE Statement" on page 254.

See: For information about how the server supports ACLs, see "ACL Security Model" on page 144.

Syntax

DELETE ACL *acl1* [*acl2...*] [C=*cat* T=*type*] [/*options*];

DELETE ACL _ALL_ [C=*cat* T=*type*] [/*options*];

Optional Arguments

ALL
deletes all existing resource ACLs for which ACLUSER has Control access.

C=*cat*
identifies the selected ACLs as names of catalog entries from the catalog *cat*. The C= option must be paired with the T= option.

/GENERIC
identifies the specified ACLs as generic ACLs.

Note If you specify /GENERIC when deleting table column ACLs, the /GENERIC applies to the table name, not to the column name. You cannot use wildcards with column names.

/LIBNAME
identifies the domain ACL.

T=*type*
identifies the catalog entry type used to qualify the selected ACLs. This option is required when the C= option is specified.

Details

Use the DELETE ACL _ALL_ syntax to delete all existing resource ACLs for which ACLUSER has Control access.

Use the DELETE ACL *acl1*[*acl2*] syntax to delete specified existing resource ACLs that are owned by ACLUSER. Specify _ALL_ as the table identifier in a two-part name to delete all tables for which the given column is matched. If you specify _ALL_ as the column identifier in a two-part name, you delete all columns for which the given table is matched.

Use the other options to uniquely identify the ACL that you want to delete. For example, use the /GENERIC option to identify a generic ACL. Use the /LIBNAME option to identify a domain ACL. Use the C= and T= options to identify an ACL for a catalog entry.

Examples

Example 1: Delete a Domain ACL

This deletes a domain ACL.

```
delete acl/LIBNAME;
```

Example 2: Delete All ACLs for Current ACL Type

This deletes all the ACLs for the current ACL catalog.

```
set acltype catalog;
delete acl _all_;
```

Example 3: Delete a Resource ACL

This deletes ACL MINE_AUG2016.

```
delete acl mine_aug2016;
```

Example 4: Delete a Generic ACL

This deletes a generic ACL MINE*.

```
delete acl mine/generic;
```

Example 5: Delete a Column ACL

This deletes a column ACL on MINE_AUG2016.SALARY.

```
delete acl mine_aug2016.salary;
```

Example 6: Delete All Column ACLs on a Table

This deletes all column ACLs on table KBIKE.

```
delete acl kbike._all_;
```

Example 7: Delete All Column ACLs on All Tables

This deletes all column ACLs on all tables.

```
delete acl _all_._all_;
```

Example 8: Delete a Catalog ACL

This deletes an ACL on the catalog RBIKE.

```
set acltype catalog;
delete acl rbike;
```

Example 9: Delete a Generic ACL on Catalog Entries

This deletes a generic ACL on the catalog entries MYCAT.MY*.CATAMS.

```
set acltype catalog;
delete acl my
c=mycat
t=catams/generic;
```

LIST ACL Statement

Lists information about ACLs.

Requirement: You must be the resource owner, have ACL access to a resource, or have special privilege in order to list ACLs. When using special privilege, set the ACLSPECIAL= option in the LIBNAME statement.

Interaction: Before using LIST ACL, you must set the context for the request with the SET ACLUSER statement. For more information, see "SET ACLUSER Statement" on page 254.

Syntax

LIST ACL *acl1* [*acl2*...] [/*options*];

LIST ACL _ALL _ [/*options*];

Optional Arguments

ALL
lists all existing resource ACLs for which ACLUSER has Control access.

C=*cat*
identifies the selected ACLs as names of catalog entries from the catalog *cat*. The C= option must be paired with the T= option.

/GENERIC
identifies the specified ACLs as generic ACLs.

Note If you specify /GENERIC when listing table column ACLs, the /GENERIC applies to the table name, not to the column name. You cannot use wildcards with column names.

/LIBNAME
identifies the domain ACL.

T=*type*
identifies the catalog entry type used to qualify the selected ACLs when the C= option is specified.

/VERBOSE
performs the requested table ACL listing, followed by the column ACLs for a specified table or tables. This is equivalent to a LIST ACL *table* followed by a LIST ACL *table*._ALL_.

Details

Use the LIST ACL _ALL_ syntax to list all existing resource ACLs for which ACLUSER has Control access.

Use the LIST ACL *acl1* [*acl2*] syntax to list specified ACL entries owned by ACLUSER. The ACL entries can be one-part resource names or two-part (*table.column*) names. Specify _ALL_ as the table identifier in a two-part name to list all tables for which the given column is matched. Specify _ALL_ as the column identifier in a two-part name to list all columns for which the given table is matched. Other optional arguments further filter the listing, except VERBOSE. The VERBOSE option expands the listing.

Examples

Example 1: List All ACL Entries
This lists all ACL entries for the current ACL type setting.

```
set acltype data;
list acl _all_;
```

Example 2: List a Generic ACL
This lists a generic ACL entry for MINE*.

```
list acl mine/generic;
```

Example 3: List All Column ACLS for a Table
This lists all column ACLs for table MINE_AUG2016.

```
list acl mine_aug2016._all_;
```

Example 4: List All Column ACLs for All Tables
This lists all column ACLs for all tables.

```
list acl _all._all_;
```

Example 5: List the Column ACL for a Specific Column
This lists the column ACL for MINE_JAN2006.SALARY.

```
list acl mine_jan2006.salary;
```

Example 6: List All ACL Data for a Table
This provides all ACL information for table MINE_AUG2016.

```
list acl mine_aug2016/verbose;
```

Example 7: List All ACLs for Catalogs
This lists all ACLs for the ACL type 'catalog'.

```
set acltype catalog;
list acl _all_;
```

Example 8: List All ACLs for a Catalog
This lists all ACLs for catalog MYCAT.?.CATAMS.

```
set acltype catalog;
 list acl _all_ c=mycat t=catams;
```

LIST USERS Statement

Lists the non-locking users that are active in the connected domain.

Requirement:	You must have special privilege and specify ACLSPECIAL=YES in the LIBNAME statement in order to use this statement.
Interaction:	Used in conjunction with the SET USERS, SHOWLIBNAME, SET MODE OPER, and OPER statements to issue proxy commands.
See:	SPD Server proxies are described in detail in Chapter 23, “Managing SPDSBASE Processes,” on page 205.

Syntax

LIST USERS;

Details

A proxy command is a command issued by one user on behalf of another user ID or user group. To issue proxy commands, you must first select a user proxy. PROC SPDO provides several interactive statements to enable you to list and select from the available user proxies. Separate statements are available for non-locking and locking user proxies. The LIST USERS statement is used in conjunction with the SET USER statement to set the context for executing a proxy command. The LIST USERS statement identifies the non-locking user proxies that are active in the connected domain so that the SET USER statement can be used to connect to one.

See “Example 1: Using PROXY Statements to Get Information about Non-Locking SPD Server Users” on page 262 to see how this statement is used with other statements to issue a proxy command.

LIST USERS/LOCKING Statement

Lists the locking users that are active in the connected domain.

Requirement:	You must have special privilege and specify ACLSPECIAL=YES in the LIBNAME statement in order to use this statement.
Interaction:	Used in conjunction with the SET USER/LOCKING, SHOWLIBNAME, SET MODE OPER, and OPER statements to issue proxy commands.
See:	SPD Server proxies are described in detail in Chapter 23, “Managing SPDSBASE Processes,” on page 205.

Syntax

LIST USERS/LOCKING;

Details

A proxy command is a command issued by one user on behalf of another user ID or user group. To issue proxy commands, you must first select a user proxy. PROC SPDO provides several interactive statements to enable you to list and select from the available user proxies. Separate statements are available for non-locking and locking user proxies. The LIST USERS/LOCKING statement is used in conjunction with the SET USER/LOCKING statement to set the context for executing a proxy command. The LIST USERS/LOCKING statement identifies the locking user proxies that are active in the connected domain so that the SET USER/LOCKING statement can be used to connect to one.

See "Example 2: Using PROXY Statements to Get Information about Locking SPD Server Users" on page 264 to see how this statement is used with other statements to issue a proxy command.

MODIFY ACL Statement

Modifies existing ACLs for resources.

Requirements: You must be the resource owner, have ACL access to a resource, or have special privilege in order to modify an ACL. When using special privilege, set the ACLSPECIAL= option in the LIBNAME statement.

Before using MODIFY ACL, you must set the context for the request with the SET ACLUSER statement. The user name specified in SET ACLUSER must be that of the ACL owner. See "SET ACLUSER Statement" on page 254.

When modifying ACLs for resources other than DATA, also set the ACL type. For more information, see "SET ACLTYPE Statement" on page 254.

Syntax

MODIFY ACL *acl1* [*acl2...*] [C=*cat* T=*type*] /[*acl-type*][*permission*];

MODIFY ACL *acl* | _ALL_ /[*acl-type*][*permission*] [*GroupList*][*UserList*];

Arguments

ALL
modifies all existing resource ACLs for which ACLUSER has Control access.

acl
specifies the name of the ACL entry that you want to modify. You can specify more than one ACL.

acl-type

/GENERIC
identifies the specified ACLs as generic ACLs.

Note: If you specify /GENERIC when modifying a table column ACL, the /GENERIC applies to the table name, not to the column name. You cannot use wildcards with column names.

/LIBNAME
identifies the domain ACL.

C=*cat*
identifies the selected ACLs as the names of catalog entries from the catalog *cat*. The C= option must be paired with the T= option.

permissions
specifies to grant or remove one or more of the following permissions:

ALTER
grants universal ALTER access to the resource.

GROUPALTER
grants group ALTER access to the resource.

GROUPREAD
grants group READ access to the resource.

GROUPWRITE
grants group WRITE access to the resource.

NOALTER
removes universal ALTER access to the resource.

NOGROUPALTER
removes group ALTER access to the resource.

NOGROUPREAD
removes group READ access to the resource.

NOGROUPWRITE
removes group WRITE access to the resource.

NOREAD
removes universal READ access to the resource.

NOWRITE
removes universal WRITE access to the resource.

READ
grants universal READ access to the resource.

WRITE
grants universal WRITE access to the resource.

GroupList
specifies the permissions that you want to assign or modify for an ACL group in the form:

```
groupname = (Y/N,Y/N,Y/N,Y/N)
```

groupname is a group name that is registered in the password database.

The comma-separated Y or N indicates a grant or denial for (READ, WRITE, ALTER, or CONTROL).

T=*type*
identifies the catalog entry type used to qualify the selected ACLs. This option is required when the C= option is specified.

UserList
specifies the permissions that you want to assign or modify for a user in the form:

```
username = (Y/N,Y/N,Y/N,Y/N)
```

username is a user name that is registered in the password database.

The comma-separated Y or N indicates a grant or denial for (READ,WRITE,ALTER,CONTROL).

Details

Use the MODIFY ACL _ALL_syntax when you want to modify all existing ACLs for which ACLUSER has Control access.

Use the MODIFY ACL *acl1* [*acl2*] syntax to modify specified ACLs that are owned by ACLUSER. If you specify _ALL_ as the table identifier in a two-part name, you modify all tables for which the given column is matched. If you specify _ALL_ as the column identifier in a two-part name, you modify all columns for which the given table is matched.

Use the *UserList* or *GroupList* option when you want to grant a user or group Control access to a resource. These options are the only way to define permissions for specific users and ACL groups.

Examples

Example 1: Modify Domain ACL

This modifies the domain ACL to set READ and WRITE access for a given user.

```
modify acl/LIBNAME
ralph=(y,y,n,n);
```

Example 2: Modify ACL MINE

This modifies ACL MINE_AUG2016 to deny universal WRITE access and add user-specific permissions.

```
modify acl mine_aug2016/nowrite
 bolick=(y,n,n,n)
johndoe=(n,n,n,n);
```

Example 3: Modify Generic ACL

This modifies a generic ACL MINE* to add user-specific permissions.

```
modify acl mine/generic
 tom=(y,y,y,n);
```

Example 4: Modify All ACLs

This modifies all ACLs to grant READ access to a given user.

```
modify acl _all_/gene=(y,,,);
```

Example 5: Modify a Column ACL

This modifies column ACL MINE_AUG2016.SALARY to add explicit READ and WRITE access for a given user.

```
modify acl mine_aug2016.salary/ralph=(y,y,n,n);
```

Example 6: Modify a Generic Column ACL

This modifies generic column ACL, MINE*.SALARY, to add explicit READ and WRITE access for a given user.

```
modify acl mine.salary/generic
 debby=(y,y,n,n);
```

Example 7: Modify ACL for a Catalog

This modifies catalog MYCAT to remove universal READ and group WRITE access.

```
set acltype catalog;
modify acl mycat/noread nogroupwrite;
```

Example 8: Modify Generic ACL for Catalog Entries

This modifies a generic ACL for catalog entries, MYCAT.MY*.CATAMS, to remove universal READ access.

```
set acltype catalog;
modify acl my
c=mycat
t=catams/generic noread;
```

OPER Statement

Specifies a privileged action to perform on the connected proxy.

Requirements: You must have special privilege and specify ACLSPECIAL=YES in the LIBNAME statement in order to use this statement.

Before you can issue this statement, you must have connected to a proxy with the SET USER or SET USER/LOCKING statement and invoked operator privilege with the SET OPER MODE statement .

See: SPD Server proxies are described in detail in Chapter 23, “Managing SPDSBASE Processes,” on page 205.

Syntax

OPER *action*;

OPER CANCEL [/DUMP];

| OPER DISCONNECT;

| OPER INTERRUPT;

Arguments

CANCEL [/DUMP]

cancels and exits the user proxy. If you specify the /DUMP option for a non-locking user proxy, the proxy exits with an ABORT function call, which produces a core file. If you are the operator of a locking user proxy, the /DUMP option is ignored.

CANCEL initiates a hard exit of the user proxy. Hard exits might leave tables opened for UPDATE access, which is an inconsistent and unusable state. In this case, you can submit the PROC DATASETS REPAIR statement to restore the tables to a usable state.

DISCONNECT

drops the control socket from the user proxy to the client. This action causes the user proxy to terminate the next time it tries to communicate with the client.

This termination initiates a hard exit of the user proxy. Hard exits might leave tables opened for UPDATE access, which is an inconsistent and unusable state. In this case, you can submit the PROC DATASETS REPAIR statement to restore the tables to a usable state.

DISCONNECT differs from CANCEL in that DISCONNECT continues the user proxy until it detects that the control socket connection has been dropped. As a result, DISCONNECT has the potential to complete. However, the point at which the user proxy detects that the control socket has been disconnected varies, which can produce different results.

INTERRUPT

sets a soft interrupt flag in any open tables that belong to the user proxy. During certain long-running operations such as large table sorts, table scans with a WHERE clause, or index creations, the user proxy periodically checks for an interrupt flag in all of the open tables that are involved in the operation. If the user proxy detects an interrupt flag, it terminates the operation and any previously opened tables are closed.

Unlike CANCEL and DISCONNECT, INTERRUPT initiates a soft exit of the user proxy. The user receives a message in the SAS log that states that the job has been interrupted. If the job did not finish, the results might be incomplete. However, the user libref remains intact, and the user proxy is still viable.

TIP To determine whether a job can be interrupted, submit a SHOWLIBNAME *libref* / DATA=_ALL_ statement before you submit the OPER INTERRUPT statement to see all of the open tables. Then, submit the SHOWLIBNAME *libref/* DATA=_ALL_ statement again after you submit the OPER INTERRUPT statement to see whether the open tables were closed. If the tables are still open after you submit the interrupt, wait a few moments and then check again. If the tables need to be closed immediately, issue an OPER CANCEL statement to cancel the user proxy.

REFRESH DOMAINS Statement

Reloads the libnames.parm parameter file in the server session.

Requirements: You must have special privilege or ACL access in order to use this statement. When using special privilege, set the ACLSPECIAL= option in the LIBNAME statement.

You must set the context for the request with the SET ACLUSER statement.

Example: "Example 3: Refreshing SPD Server Domains" on page 266

Syntax

REFRESH DOMAINS;

Details

Refreshing the libnames.parm parameter file with the REFRESH DOMAINS statement activates new settings in the file without having to restart the server.

REFRESH PARMS Statement

Reloads the spdsserv.parm parameter file in the server session.

Requirements: You must have special privilege and specify ACLSPECIAL=YES in the LIBNAME statement in order to use this statement.

You must set the context for the request with the SET ACLUSER statement.

Syntax

REFRESH PARMS;

Details

The REFRESH PARMS statement activates new settings in the spdsserv.parm parameter file without having to restart the server.

The statement cannot be used to change user authentication options. To change user authentication options, you must re-start the server.

SET ACLTYPE Statement

Sets the member type for subsequent ACL operations.

Interaction: Used in conjunction with the ADD ACL, DELETE ACL, and LIST ACL statements to indicate the resource type to which the ACL actions apply.

Syntax

SET ACLTYPE *type*;

Required Argument

type

specifies the type of resource on which the ACL action is to be applied. Valid values are DATA, CATALOG, and VIEW. When this statement is omitted, the default is DATA.

Example

See "Example 6: Add Catalog ACL" on page 234.

SET ACLUSER Statement

Sets the user scope for subsequent ACL operations.

Interaction: Used in conjunction with the ADD ACL, DELETE ACL, LIST ACL, and MODIFY ACL statements to set the context for the ACL request.

Syntax

SET ACLUSER *name*;

Required Argument

name

specifies a user name or group name. The name must be registered in the password database.

Details

To perform an ACL operation on a resource, you must meet these criteria:

- be the resource owner
- be the ACL entry owner

- have Control access to the ACL entry
- have special privilege in the password database and specify ACLSPECIAL=YES in the LIBNAME statement.

In addition, you must specify the name of the ACL owner in the SET ACLUSER statement. The SET ACLUSER statement sets the context for the all subsequent ACL requests. If the SET ACLUSER statement is executed without specifying a user name, the default user is the USER= specified in the LIBNAME statement.

Examples

Example 1: Assign the Scope to the Libref User

```
libname mylib sasspds 'd2'
server=zztop.5162
user='prod1'
password='spds123'
IP=YES ;

proc spdo library=mylib ;
/* assign the scope to the libref user */
set acluser prod1 ;
```

Example 2: Assign the Scope to Another User to Modify ACLs

In this example, the ACLSPECIAL= option invokes special privilege for user Admin1 to enable Admin1 to modify an ACL owned by user Prod1.

```
libname admin1d2 sasspds 'd2'
server=zztop.5162
user='admin1'
password='spds123'
ACLSPECIAL=YES
IP=YES ;
proc spdo library=admin1d2 ;
set acluser prod1 ;
```

SET MODE OPER Statement

Enables you to run privileged operator commands.

Requirement:	You must have special privilege and specify ACLSPECIAL=YES in the LIBNAME statement in order to use this statement.
Interaction:	Used in conjunction with the LIST USERS[/LOCKING], SET USERS[/LOCKING], SHOWLIBNAME, and OPER statements to issue proxy commands.
See:	SPD Server proxies are described in detail in Chapter 23, “Managing SPDSBASE Processes,” on page 205.

Syntax

SET MODE OPER;

Details

The SET MODE OPER statement sets you as the operator of the user proxy that you are currently set to. It is typically executed after the SET USER or SET USER/LOCKING statement.

A user proxy can have only one operator at any time. If you submit the SET MODE OPER statement when someone is already established as operator of the user proxy, you get the following message:

```
ERROR: Operator mode owned by another connection.
Cannot grant this request.
```

After you have successfully set yourself as the operator, you can submit the OPER statement. See "OPER Statement" on page 252.

SET USER Statement

Connects to a non-locking proxy for the specified user ID.

Requirement: You must have special privilege and specify ACLSPECIAL=YES in the LIBNAME statement to use this statement.

Interaction: Used in conjunction with the LIST USERS, SHOWLIBNAME, SET MODE OPER, and OPER statements to issue proxy commands.

See: SPD Server proxies are described in detail in Chapter 23, "Managing SPDSBASE Processes," on page 205.

Syntax

SET USER *user-ID* [*port-number*];

Arguments

user-ID
specifies a user ID that is returned by the LIST USERS statement. The user ID is required.

port-number
specifies a proxy operator port number. The port number is optional. The proxy operator port number enables you to distinguish between two proxies that share the same user ID.

Details

The SET USER statement is used after the LIST USERS statement in preparation for executing a proxy command. The LIST USERS statement identifies the non-locking users that are active in the connected domain. The SET USER statement enables you to connect to one of the listed user proxies.

After you have established a user connection with SET USER, you can issue the SHOWLIBNAME statement to list the active librefs for the user and to get information about tables that are accessible or open in specific librefs. You can also use the SET MODE OPER and OPER statements to cancel, disconnect from, or interrupt the user proxy.

See “Example 1: Using PROXY Statements to Get Information about Non-Locking SPD Server Users” on page 262 to see how this statement is used with other statements to issue a proxy command.

SET USER/LOCKING Statement

Connects to the locking proxy for the specified user ID and thread ID.

Requirement:	You must have special privilege and specify ACLSPECIAL=YES in the LIBNAME statement to use this statement.
Interaction:	Used in conjunction with the LIST USERS/LOCKING, SHOWLIBNAME, SET MODE OPER, and OPER statements to issue proxy commands.
See:	SPD Server proxies are described in detail in Chapter 23, “Managing SPDSBASE Processes,” on page 205.

Syntax

SET USER/LOCKING *user-ID* THREAD=*thread-ID*;

Arguments

user-ID
specifies a user ID returned by the LIST USERS/LOCKING statement.

THREAD=*thread-ID*
specifies the thread ID associated with the user ID.

Details

The SET USER/LOCKING statement is used after the LIST USERS/LOCKING statement in preparation for executing a proxy command. The LIST USERS/LOCKING statement identifies the locking users that are active in the connected domain. The SET USER/LOCKING statement enables you to connect to one of the listed user proxies.

After you have established a user connection with SET USER/LOCKING, you can issue the SHOWLIBNAMES statement to list the active librefs for the user and to get information about tables that are accessible or open in specific librefs. You can also use the SET MODE OPER and OPER statements to cancel, disconnect from, or interrupt the user proxy.

See “Example 2: Using PROXY Statements to Get Information about Locking SPD Server Users” on page 264 to see how this statement is used with other statements to issue a proxy command.

SHOWLIBNAME Statement

Lists libref and table information for a connected user proxy.

Requirement:	You must have special privilege and specify ACLSPECIAL=YES in the LIBNAME statement to use this statement.
Interaction:	Used in conjunction with the LIST USERS, LIST USERS/LOCKING, SET USER, SET USER/LOCKING, SET MODE OPER, and OPER statements to issue proxy commands.

Syntax

SHOWLIBNAME *options*;

SHOWLIBNAME *libref* | _ALL_;

SHOWLIBNAME *libref* / DATA=[_ALL_| *table-name*];

SHOWLIBNAME *libref* / DUMP=[_ALL_| *table-name*];

Details

- Specify the SHOWLIBNAME statement with the _ALL_ option to see every currently assigned libref for the selected user proxy.
- Specify the SHOWLIBNAME statement with a *libref* to list information about that specific libref.
- Specify the SHOWLIBNAME statement with a *libref* and the /DATA=_ALL_ option to list open tables in the server connection.
- Specify the SHOWLIBNAME statement with a *libref* and the /DATA=*table-name* option to list information about a specific table in the server connection.
- Specify the SHOWLIBNAME statement with a *libref* and the /DUMP=_ALL_ option to list accessible tables in the libref's server connection.

For more information, see "Example 1: Using PROXY Statements to Get Information about Non-Locking SPD Server Users" on page 262 and "Example 2: Using PROXY Statements to Get Information about Locking SPD Server Users" on page 264.

SPDSCMD Statement

Sends SPD Server a command to run.

Requirement: You must have special privilege and specify ACLSPECIAL=YES in the LIBNAME statement to use this statement.

Example: "Example 4: Using PROC SPDO to Back Up and Restore Tables" on page 266.

Syntax

SPDSCMD '*command*';

Required Argument

command
specifies a host command or a server command.

Details

SPD Server uses the host system() call to run the specified command. The command is executed according to the user ID rights of the individual or administrator that started the server session. The server returns the system() call value to PROC SPDO. PROC SPDO echoes the exit status of the command that was run on the host.

Here are examples of commands that can be issued from the SPDSCMD statement:

- List the WORKPATH directory:

```
/* UNIX */
spdscmd 'ls /spdswork/*.spds9';

/* Windows */
spdscmd 'dir d:\spdswork\*.spds9';
```

- Clean up WORKPATH files:

```
/* UNIX */
spdscmd 'rm /spdswork/*.spds9';

/* Windows */
spdscmd 'del d:\spdswork\*.spds9';
```

The SPD Server backup and restore utilities (spdsbkup and spdsrstr), index utility (ixutil), and table list utility (spdsls) can also be executed in the SPDSCMD statement. Executing the utilities remotely via PROC SPDO circumvents the need to set local environment variables for the utilities. See "Example 4: Using PROC SPDO to Back Up and Restore Tables" on page 266 to see how the utilities are invoked.

When you use the SPDSCMD statement to perform a system command, the procedure sets the SAS SYSRC macro variable to the value that was returned by the system command.

Note: System commands executed on Linux do not reliably return the status of the command being executed. Therefore, PROC SPDO cannot set the SYSRC macro variable.

If the server cannot run the system command for any reason, PROC SPDO sets the SAS SYSERR macro to 1012.

SPDSMAC Statement

Prints the default values for SPD Server macro variables.

Syntax

spdsmac;

Example

After assigning a libref that specifies the SASSPDS engine and connects to an SPD Server domain, enter the following in your SAS session:

```
proc spdo lib=libref;
spdsmac;
```

The server returns information similar to the following in the SAS log:

```
                SPD Macro Variables list/settings.
        SPDSACWH=NO     Controls whether to use Where clause filtering for
HADOOP processing.
        SPDSAUNQ=NO     Controls appends to tables with unique indexes.
        SPDSBNEQ=NO     Controls BYNOEQUALS for BY processing.
        SPDSBSRT=YES    Controls Sorting for BY processing.
        SPDSCLJX=NO     Controls whether SAS will consider doing an index
                        join to a SPD cluster table index with an
                        unknown number of values.
        SPDSCOMP=YES    Controls Network Data Compression.
        SPDSCMPF=0      Number of bytes free in each compress block
                        for later updates. Default is 0 bytes.
        SPDSDCMP=NO     Controls Data Compression.
        SPDSEINT=YES    Controls network interrupts for SQL pass-through.
        SPDSEOBS=0      Ending observation number.
        SPDSEV1T=1      EVAL1 method type(0=sync,1=async).
        SPDSEV2T=1      EVAL2 method type(0=sync,1=async).
        SPDSFSAV=NO     Controls whether output files are deleted.
        SPDSHPRD=NO     Controls whether to use parallel reads for HADOOP
processing.
        SPDSIASY=NO     Controls whether indexes are created in parallel.
        SPDSIPDB=NO     Controls whether Implicit SQL Pass-Through errors
appear in the SAS log.
        SPDSIRAT=0      Percentage of segments on which to perform pre-
evaluation.
        SPDSNBIX=NO     Controls use of indexes during a BY sort.
        SPDSNETP=4096   Network Packet Size.
        SPDSNIDX=NO     Controls whether indexes are used during Where clause
                        planning or by processing.
        SPDSRSSL=NO     Controls SSL communication with the server.
        SPDSSADD=NO     Controls whether record updates are synchronous.
                        Important when using Unique indexes and duplicate values
                        are detected.
        SPDSSIZE=0      File Partition size (in M units).
        SPDSSOBS=0      Starting observation number.
        SPDSSQLR=NO     SQL reset options specified in implicit pass-through.
        SPDSSTAG=NO     Controls whether to use tagged sorting.
        SPDSTCNT=0      Number of threads to use during Where clause
evaluations.
        SPDSUSAV=NO     Controls whether records with duplicate
                        keys are saved in a SAS Data Set.
        SPDSUSDS=       Name of the SAS Data Set which has the
                        records with duplicate keys.
        SPDSVERB=NO     Controls verbose listing of data set contents.
        SPDSWCST=NO     Controls whether to use dynamic Where clause costing.
        SPDSWDEB=NO     Controls whether WHERE clause planning tree is printed.
        SPDSWSEQ=NO     Controls whether to override Where clause costing.
```

TRUNCATE Statement

Removes data from a table without deleting the table structure or metadata.

Requirements: You must be the table owner or have been granted WRITE access to the table by the table owner in order to use this statement.

You must set the context for the TRUNCATE statement with the SET ACLUSER statement.

Syntax

TRUNCATE *table-name*;

Details

Use the TRUNCATE statement when you want to remove old data from a table and leave the table structure in place so that new data can be appended.

The TRUNCATE statement can be executed in the SQL explicit pass-through facility, or with PROC SPDO.

Example

The following code shows how the TRUNCATE statement is executed in PROC SPDO.

```
%let host=kaboom ;
%let port=5191 ;
%let domain=path2 ;

libname &domain sasspds "&domain"
   server=&host..&port
   user='siteusr1'
   password='mypasswd'
   ip=YES ;

/* create a table */
data &domain..staceys_table ;

   do i = 1 to 100 ;
   output ;
end ;
run ;

/* verify the contents of the created table */

proc contents data=&domain..staceys_table ;
run ;

/* SPDO Truncate statement deletes the table  */
/* data but leaves the table structure in   */
/* place so new data can be appended        */

proc spdo lib=&domain ;
set acluser ;
truncate staceys_table ;

quit ;

* verify that no rows or data remain in    */
/* the structure of staceys_table           */
proc contents data=&domain..staceys_table ;
run ;
```

Examples: SPDO Procedure

Example 1: Using PROXY Statements to Get Information about Non-Locking SPD Server Users

Features:	LIST USER statement
	SET USER statement
	SHOWLIBNAME statement with _ALL_ option
	SHOWLIBNAME statement with /DUMP option
	SHOWLIBNAME statement with /DATA option

Details

Here are examples of proxy statements that an administrator might submit to get information about non-locking users. They are presented in an order that would be typical for an administrator who is gathering user proxy information before issuing proxy commands on the user's behalf.

1. List the users on the host that you are interested in. The following code requests to list all of the users for the SPD Server host named Sunburn.6200. The host is identified in the HOST= LIBNAME option. The code also specifies the ACLSPECIAL= LIBNAME option.

```
libname example sasspds
  host='sunburn'
  serv='6100'
  user='sassyl'
  passwd='abc123'
  aclspecial=YES;

proc spdo lib=example;
list users;
```

The following output is written to the log:

```
Users Currently Connected to SPD Server
UserName Pid   Portno
---------------------------------------
SASSYL   17704 58382
SASSYL   17614 58298
SASSYL   17613 58293
ANONYMOU 17611 58288
ANONYMOU 17610 58283
```

2. Set the user. The following code sets the user to ANONYMOU and specifies process ID (PID) 17610:

```
set user anonymou 17610;
```

The following message is written to the log:

```
NOTE: User ANONYMOU connected to proxy operator port with pid=17610.
```

3. List the active librefs for the user with the SHOWLIBNAMES statement. This code submits the SHOWLIBNAME statement with the _ALL_ option to list every libref for user ANONYMOU for this proxy.

```
showlibname _all_;
```

The following output is written to the log:

```
LIBREF(FOO):Pathname assigned=/bigdisk/test/qabig1_dev/
LIBREF(FOO):ACL Owner=
LIBREF(FOO):ACL Defaults(R,W,A,C)=(Y,Y,Y,Y)
```

4. List the open tables in a libref. This code submits the SHOWLIBNAME statement with libref FOO and the DATA= _ALL_ option to show all of the open tables in libref FOO.

```
showlibname FOO/data=_all_;
```

The code might return a message similar to the following in the log:

```
NOTE: No data sets currently opened for LIBREF FOO.
```

5. List the accessible tables in the libref. This code submits the SHOWLIBNAME statement with libref FOO and the /DUMP option to show all of the accessible tables in libref FOO.

```
showlibname FOO/dump=_all_;
```

The following output is written to the log:

```
LIBREF(FOO):Dataset name=BIGX
LIBREF(FOO):ACL Owner=ANONYMOU
LIBREF(FOO):ACL Defaults(R,W,A,C)=(N,N,N,N)
LIBREF(FOO):Dataset name=X
LIBREF(FOO):ACL Owner=ANONYMOU
LIBREF(FOO):ACL Defaults(R,W,A,C)=(N,N,N,N)
```

6. Submit a request and list the open tables in the libref FOO. User ANONYMOU issues a WHERE clause on the table BIGX. Submit the SHOWLIBNAME statement on libref FOO again with the /DATA option.

```
showlibname FOO/data=_all_;
```

The following output is written to the log:

```
LIBREF(FOO):Dataset name=BIGX
LIBREF(FOO):ACL Owner=ANONYMOU
LIBREF(FOO):ACL Defaults(R,W,A,C)=(N,N,N,N)
LIBREF(FOO):WHERE clause read thread active
```

7. Get detailed information. Submit the SHOWLIBNAME statement on libref FOO again and specify table BIGX in the /DATA= option.

```
showlibname FOO/data=bigx;
```

The following output is written to the log:

```
LIBREF(FOO):Dataset name=BIGX
LIBREF(FOO):ACL Owner=ANONYMOU
LIBREF(FOO):ACL Defaults(R,W,A,C)=(N,N,N,N)
LIBREF(FOO):WHERE clause read thread active
LIBREF(FOO):Type=
LIBREF(FOO):Label=
LIBREF(FOO):Number observations=5000000
LIBREF(FOO):Observation length=41
LIBREF(FOO):Wire blocksize=32718
LIBREF(FOO):Wire block factor=798
LIBREF(FOO):Data port number=58392
LIBREF(FOO):Active data socket=33
LIBREF(FOO):Metafile=/bigdisk/test/qabig1_dev/bigx.mdf.0.0.0.spds9
LIBREF(FOO):Metafile size=31
LIBREF(FOO):Datafile=
  /spds02/test/qabig1_dev/bigx.dpf._bigdisk_test_qabig1_dev.0.1.spds9:
  /spds03/test/qabig1_dev/bigx.dpf._bigdisk_test_qabig1_dev.1.1.spds9:
  /spds04/test/qabig1_dev/bigx.dpf._bigdisk_test_qabig1_dev.2.1.spds9:
  /spds01/test/qabig1_dev/bigx.dpf._bigdisk_test_qabig1_dev.3.1.spds9:
  /spds02/test/qabig1_dev/bigx.dpf._bigdisk_test_qabig1_dev.4.1.spds9:
  /spds03/test/qabig1_dev/bigx.dpf._bigdisk_test_qabig1_dev.5.1.spds9:
  /spds04/test/qabig1_dev/bigx.dpf._bigdisk_test_qabig1_dev.6.1.spds9:
  /spds01/test/qabig1_dev/bigx.dpf._bigdisk_test_qabig1_dev.7.1.spds9:
  /spds02/test/qabig1_dev/bigx.dpf._bigdisk_test_qabig1_dev.8.1.spds9:
  /spds03/test/qabig1_dev/bigx.dpf._bigdisk_test_qabig1_dev.9.1.spds9:
  /spds04/test/qabig1_dev/bigx.dpf._bigdisk_test_qabig1_dev.10.1.spds9:
  /spds01/test/qabig1_dev/bigx.dpf._bigdisk_test_qabig1_dev.11.1.spds9:
  /spds02/test/qabig1_dev/bigx.dpf._bigdisk_test_qabig1_dev.12.1.spds9
LIBREF(FOO):Datafile size=200196
LIBREF(FOO):Number of Indexes=0
```

Example 2: Using PROXY Statements to Get Information about Locking SPD Server Users

Features:	LIST USERS/LOCKING statement
	SET USER/LOCKING statement
	SHOWLIBNAMES statement

Details

Here are examples of proxy statements that an administrator might submit to get information about locking users. They are presented in an order that would be typical for an administrator who is gathering user proxy information before issuing proxy commands on the user’s behalf.

1. List all locking users for the server Sunburn 6100. This example uses the same libref as the previous example and submits the LIST USERS/LOCKING statement.

```
libname example sasspds
  host='sunburn'
  serv='6100'
  user='sassyl'
  passwd='abc123'
  aclspecial=YES;

proc spdo lib=example;
list users/locking;
```

Information similar to the following is written to the log:

```
Users Currently Connected to the Record Level Proxy
SPDUserName  Client Login   Thread Id
----------------------------------------
ANONYMOU     SASTEST        7
TEST         SASTEST        8
```

2. Set the user and specify the thread ID. The code submits the SET USER/ LOCKING statement to set the user to ANONYMOU and the thread ID to 7.

```
set user/locking anonymou threadid=7;
```

A message similar to the following is written to the log:

```
NOTE: User ANONYMOU connected to record
level proxy operator port with thread=7.
```

3. List the active librefs for the locking user with the SHOWLIBNAMES statement.

```
showlibname _all_;
```

The following output is written to the log:

```
LIBREF(LOCKING):Pathname assigned=/bigdisk/test/qabig1/
LIBREF(LOCKING):ACL Owner=
LIBREF(LOCKING):ACL Defaults(R,W,A,C)=(Y,Y,Y,Y)
```

4. List the open tables in the locking libref. This code submits the SHOWLIBNAME LOCKING statement with the /DATA= _ALL_ option. Note that there is no need to specify a libref in the SHOWLIBNAME statement.

```
showlibname LOCKING/data=_all_;
```

A message similar to the following might be written to the log. There are currently no tables open for the locking user ANONYMOU.

```
NOTE: No data sets currently opened for LIBREF LOCKING.
```

Example 3: Refreshing SPD Server Domains

Features: REFRESH DOMAINS statement

Details

This example shows how to activate a new domain without having to stop and restart the server. In this example, domain RefTest is a pre-existing domain. The new domain is RefTest2. Both domains have OWNER=ADMIN set in their definitions in the libnames.parm parameter file.

Connect to the server and execute PROC SPDO with the REFRESH DOMAINS statement using an existing domain. In the LIBNAME statement, specify the ACLSPECIAL= LIBNAME option. Set the user context to the domain owner by using the SET ACLUSER statement.

```
libname mylib sasspds 'reftest'
server=d8488.5180
user='admin'
password='spds123'
aclspecial=YES;

proc spdo library=mylib;
SET acluser admin;
REFRESH DOMAINS;
quit;
```

Reconnect to the server and specify the RefTest2 domain. Domains that have an OWNER= option such as RefTest2 must be reconnected to the domain.

```
libname mylib2 sasspds 'reftest2'
server=d8488.5180
user='admin'
password='spds123';
```

Example 4: Using PROC SPDO to Back Up and Restore Tables

Features: SPDSCMD statement with SPDSBKUP and SPDSRSTR commands

Details

This example shows how to execute the server backup and restore utilities, spdsbkup and spdsrstr, via the PROC SPDO SPDSCMD statement. In order to issue the SPDSCMD statement, you must have special privilege in the password database.

When you execute commands using the PROC SPDO SPDSCMD statement, the current working directory is the root directory of the server. Messages generated by the commands are echoed to the SAS log. In this example, the incremental backup and restore utilities reside in the SPD Server directory. The incremental backup and restore files are saved in the server directory **`/spdsadm/bkup`**.

Note: There is a limitation when you use the -AFORCE option with PROC SPDO to restore data on Windows. The -AFORCE option fails if ACLs exist and there are active connections to the domain that were specified by using the -D option during the restore process. ACLSPECIAL= connections to a libref must specify a domain that is separate from the domain in which you are attempting to restore the ACLs (if the ACLs currently exist). If you make ACLSPECIAL= libref connections that specify the domain in which you are attempting to restore the ACLs, then the ACL restore operation fails.

Issue a LIBNAME statement that specifies special privilege. This example creates the libref Backup for the domain Test on the host machine Sunny. The port number of the name server is 5150. The connection specifies the Admin user ID and password.

```
libname backup sasspds 'test'
host='sunny'
serv='5150'
user='admin'
passwd='admin'
ACLSPECIAL=YES;
```

Invoke PROC SPDO for the libref and specify the SPDSBKUP command in the SPDSCMD statement. The following code performs a full SPD Server backup of the domain Tstdomn. The backup is made on February 3, 2016. The command creates the backup file **`/spdsadm/bkup/test_BK_03Feb2016_233000.0.0.spds9`** and the table of contents file **`/spdsadm/bkup/test_TC_03Feb2016_233000`**.

```
proc spdo lib=backup;
spdscmd 'spdsbkup -a -full -d tstdomn -h sunny -s 5150 -f /spdsadm/bkup/test';
```

To restore SPD Server from a backup, invoke PROC SPDO and specify the SPDSRSTR command in the SPDSCMD statement. The restore utility restores the domain to its last full backup state.

```
spdscmd 'spdsrstr -aforce -d tstdomn -h sunny -s 5150 -e /spdsadm/bkup/test';
```

Example 5: Using PROC SPDO to Truncate a Table

Features: TRUNCATE statement

Details

The following code creates and then truncates a sample table to illustrate use of the TRUNCATE statement.

```
%let host=kaboom;
%let port=5400;

libname mylib sasspds "path"
host="&host"
```

```
serv="&port"
user='anonymous'
ip=YES;

/* create a table */
data mylib.sample_table
do i = 1 to 100;
output;
end;
run;
/* verify the contents of the created table */
proc contents data=mylib.sample_table;
run;

/* truncate the table */
proc spdo lib=mylib
set acluser;
truncate sample_table;
quit;

/* verify that no rows or data remain in */
/* the structure of sample_table */
proc contents data= mylib.sample_table
run;
```

Chapter 27
Backing Up and Restoring SPD Server Data

Overview of the Backing Up and Restoring SPD Server Data **270**

Pros and Cons of Each Backup Approach **270**
SPD Server Utilities 270
System Backup Utilities 271

Combining Backup Approaches **271**

Introduction to the SPD Server Backup and Restore Utilities **272**
Overview 272
Path Requirements for SPD Server Utilities 272
Compatibility with Previous Versions 272
Privileged Access Protection 272
Invoking the Backup and Restore Utilities 273

SPD Server Backup Utility (spdsbkup) **273**
Backup Requirements 273
Backup Process 274
Content of a Full Backup 274
Content of an Incremental Backup 275
Backup Data File 275
Backup Table of Contents File 275

Using spdsbkup **276**
Syntax for the spdsbkup Utility 276
Options for the spdsbkup Utility 277
Return Values for the spdsbkup Utility 279
Log Messages for the spdsbkup Utility 279

SPD Server Restore Utility (spdsrstr) **279**
Restore Requirements 279
Usage Overview 280

Using spdsrstr **280**
Syntax for the spdsrstr Utility 280
Options for the spdsrstr Utility 281
Return Values for the spdsrstr Utility 282
Log Messages for the spdsrstr Utility 282

Sample Usage Scenarios **283**
Overview 283
Performing Full and Incremental Backups with spdsbkup 283
Using a System Utility to Perform Full Backups 284
Restore a Single SPD Server Table 285

Restore an SPD Server Domain 286
Back Up and Restore Table Indexes Using SPD Server Full Backups 286
Back Up and Restore Table Indexes Using System Full Backups 287

Overview of the Backing Up and Restoring SPD Server Data

It is important that you make regular backups of your server tables. You can use the standard file system backup and restore facilities that native operating systems provide to back up and restore your server tables. Or you can use utilities provided by SPD Server to back up and restore your server tables. Each approach has trade-offs. Therefore, you might want to consider using a combination of the two approaches, depending on the size of your server tables and how frequently they are updated.

Server tables can be enormous in size, surpassing the file size limits maintained by some operating environments. SPD Server is also dependent on the operating environment's ability to detect a modification to a table, such as adding, deleting, or modifying a record.

A change in a table is typically a signal to ensure that the newly modified file is backed up. When a standard utility subsequently performs an incremental backup, it processes the file change by backing up the entire table. If the table is very large, the backup time can be lengthy. In addition, the processing can consume considerable resources. Administrators frequently struggle with a dilemma of whether incremental backups of large tables are worth the resources that they consume.

The SPD Server backup and restore utilities alleviate these problems by backing up and restoring only changed rows. Instead of backing up an entire table, the backup utility backs up only the rows that changed after the previous backup date. If a subsequent restore of the table becomes necessary, the restore utility can incrementally restore the table to its last backup state. By backing up and restoring only the changed rows, SPD Server conserves valuable system resources. The software also gives you the option to perform periodic full backups with your system's full backup and restore facilities with SPD Server incremental backups.

Pros and Cons of Each Backup Approach

SPD Server Utilities

Benefits

- Incremental backups copy only changed rows, instead of the entire table, which conserves disk space.
- Incremental restores replace specific rows, rather than an entire table, which can save time.
- The utilities will not back up tables that are being updated.

Limitations

The SPD Server backup utility does not back up the following:

- cluster tables. In order to back up a cluster table, you must undo the cluster, back up the individual member tables, and then remake the cluster. Likewise, to restore a damaged cluster table, you must undo the cluster, restore the member tables individually, and remake the cluster.
- table WHERE constraints.
- index files. The backup utility backs up only the definition of the index from the metadata. The index is re-created by using the metadata when a full restore is performed.

Incremental backups can take a long time. The utilities use SQL to push queries to the tables being backed up and restored. To accommodate this, three extra columns are added to each table—Last Date Modified, Deleted, and Modified—which are hidden to the user. Backup queries for full backups read the entire table. Backup queries for increments must push a query to set the rows modified since the last backup using the values in the Last Date Modified column. The Last Date Modified column is not indexed. This can be a very long query for a large table.

System Backup Utilities

Benefits

- System utilities do not parse the data set. Therefore, this type of backup usually runs faster than the SPD Server utility when you are doing a full backup.
- System utilities back up table indexes.

Limitations

- System utilities do not coordinate with tables being modified. They just read the table parts and back them up, so keeping them in sync could be an issue.

Combining Backup Approaches

To optimize resources, consider the following:

- Do full backups with a system utility.
- Use the server utilities to do incremental backups of tables that have minimal change. The server utilities will back up only rows that have changed.
- For tables that get heavy or frequent updates, do incremental backups with a system utility. The incremental system updates will back up files that have changed. Because SPD Server uses multiple data partition files for table data, only the data partition files that have changed will need incremental backup from a system utility.

Introduction to the SPD Server Backup and Restore Utilities

Overview

SPD Server provides the spdsbkup and spdsrstr utilities for backing up and restoring your server tables:

- spdsbkup

  performs a full or incremental backup of a server table or domain, storing the information in a backup data file.
- spdsrstr

  performs a full or incremental restore of a server table or domain using the backup data file that was created by the spdsbkup utility.

Path Requirements for SPD Server Utilities

SPD Server provides National Language Support (NLS) for multiple languages and character sets in database operations. As a result, all SPD Server utilities require access to the ***InstallDir*/bin** directory. You must ensure that the ***InstallDir*/bin** directory is included in your SPD Server library path specification.

Here is an example of a statement that specifies the necessary path:

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:InstallDir/bin
export LD_LIBRARY_PATH
```

Compatibility with Previous Versions

SPD Server 5.3 backup and restore utilities cannot restore SPD Server backup files created with SPD Server 3.x and earlier. You must use your SPD Server 3.x utilities to restore SPD Server 3.x backup files, and then archive the restored files using the SPD Server 5.3 utilities.

Privileged Access Protection

Running the SPD Server backup and restore utilities is a privileged operation. For a user to have access to the SPD Server backup and restore utilities, one of the following statements must be true:

- The user ID that starts the SPD Server session must be identical to the user ID that runs the backup and restore utilities. By definition, the user ID that starts the SPD Server session is a privileged user.
- The user has ACLSPECIAL=YES privileges.

Access to the backup and restore utilities is granted to the special user SPDSBKUP. You can use the user (-u) and password (-p) options for the utilities to give a specific privileged user access to the utilities.

Invoking the Backup and Restore Utilities

spdsbkup and spdsrstr can be invoked as stand-alone utilities, or they can be invoked in PROC SPDO by using the SPDSCMD statement. For more information about the PROC SPDO SPDSCMD statement, see "SPDSCMD Statement" on page 258.

The following environment variables must be established in order for the spdsbkup utility to be invoked as a stand-alone utility. Note that these are not needed if the utility is invoked in PROC SPDO.

For UNIX:

```
INSTDIR=<!replace with SPD Server 5.3 Installation Directory!>
 SASVER=9.4
 SASHOME=<!replace with SPD Server 5.# Installation Directory!>
 LD_LIBRARY_PATH=$INSTDIR/bin:$SASHOME/SASFoundation/$SASVER/sasexe
 export LD_LIBRARY_PATH
 LIBPATH=$INSTDIR/bin:$SASHOME/SASFoundation/$SASVER/sasexe
 export LIBPATH
 TKPATH=$INSTDIR/bin:$SASHOME/SASFoundation/$SASVER/sasexe:$SASHOME/SASFoundation
 /$SASVER/utilities/bin
 export TKPATH
 MSGPATH=$INSTDIR/misc/spds
 export MSGPATH
```

For Windows:

```
SET SASROOT=<!replace with SPD Server 5.3 Installation Directory!>
SET INSTALLDIR=<!replace with SPD Server 5.3 Installation Directory!>
SET MSGPATH=@INSTALLDIR@\spds\sasmisc
SET TKPATH=@INSTALLDIR@\bin;@SASROOT@\core\sasext
```

SPD Server Backup Utility (spdsbkup)

Backup Requirements

Client Access to an SPD Server Domain

The client that performs the backup does not have to execute on the same machine as the SPD Server. However, the client must be able to access the physical path of the server domain that is being backed up. The client can access the physical path of the domain directly or through a network connection.

Domain Preparation

In order for a table or domain to be eligible for backup, the domain BACKUP=YES option must be set in the libnames.parm parameter file. Here are examples of two domain definitions from a data server's libnames.parm parameter file:

```
LIBNAME=nobackup pathname=/usr/foo/test;

LIBNAME=canbackup pathname=/usr/foo/test BACKUP=YES;
```

The domain definition called NoBackup creates tables in the directory **`/usr/foo/test`**, but it does not specify the BACKUP= option. Therefore, tables that are created through this domain definition are ineligible for backup. In contrast, the definition for the domain CanBackup, which creates tables in the same directory, specifies the BACKUP=YES option. As a consequence, tables that are created through this domain are eligible for backup.

When spdsbkup performs a backup, it checks every table in **`/usr/foo/test`**. However, based on the parameter file entries in this example, spdsbkup backs up only the eligible tables in CanBackup. When you create client connections using pass-through or LIBNAME statements, you can use the BACKUP=NO LIBNAME option to override default backup settings.

Incremental Backups Require a Prior Full Backup

Before you can do an incremental backup of a server table, you must do a full backup of the table. You can perform the full backup in two ways:

- Use the system's full backup utility, and then inform spdsbkup of the system's last full backup date.
- If a full backup has not been done before, use spdsbkup to perform a full backup.

After you have performed a full backup on a server table, you can then proceed with an incremental backup strategy. The first incremental backup saves all table changes that were made after the last full backup date. Each successive incremental backup saves the changes that were made after the previous incremental backup.

Backup Process

The backup utility spdsbkup performs a full or incremental backup of a server table or domain. It creates a backup file that contains full backups of newly created server tables or incremental backups of tables that have been backed up before.

During the backup process, the spdsbkup utility performs the following tasks:

- connects to a specified SPD Server
- uses the SPD Server pass-through facility to generate SQL queries on server domain tables
- backs up the records that the query returned
- compresses the record data
- stores the data in a flat data file so that the restore utility can use it later when it restores the tables
- creates a table of contents file that contains summary information about the backup.

Content of a Full Backup

When you do a full backup of a server table, all of the table rows and attributes (indexes, partition size, compression, and sorted) are backed up. ACL files must be in the same physical directory as the domain. If any ACL file does not meet this requirement, the ACLs are not backed up, and a warning message is sent to the log. The spdsbkup utility continues to back up all specified tables.

When you perform an SPD Server full backup of a table, the SPD Server full backup utility does not save the index data. It saves only the information necessary to re-create the indexes when the table is restored. Therefore, when you back up table indexes, the information that is saved does not require additional overhead or a lot of additional space. If you must fully restore a table later, the SPD Server restore utility, spdsrstr, can re-create the indexes when the table is created. Or you can use the -n option in the restore. The -n option suppresses index creation. After the table is fully restored, you can use PROC DATASETS or PROC SQL to re-create the indexes.

Content of an Incremental Backup

When you perform an incremental backup of a server table, only changes that were made to the table rows since the last full backup are included in the backup. Changes to the table attributes are not backed up. When you restore an SPD Server incremental backup, the incremental changes to the rows are applied. Only attributes that were associated with the table at the time of the last full backup (indexes, partition size, compression, and sorted) are applied to the restored rows.

Backup Data File

Backup Data File Nomenclature

The spdsbkup utility stores backup data in a file named *filename*`_BK_ddmmmyyyy_hhmmss`.0.0.0.spds9. The suffix, which is added to the filename, generates a unique backup file that indicates when the backup was performed. Because the suffix is unique, you can use the same filename for successive backups of a domain or a table, without overwriting an existing file.

Backup Data File Extensions

If the backup file exceeds the system file size limit, spdsbkup automatically extends the file and stores the excess data in additional files. The software identifies these files with a file extension that follows the date/time. (The SPD Server file extension is the 0.0.0 portion of the filename.) Although the extensions for the files vary, the date/time is the same on all the files. You must have a backup file complete with file extension before you can begin a restore session.

Backup Table of Contents File

The backup table of contents file has a name in the form *filename*`_TC_ddmmmyyyy_hhmmss`. The TC in the filename identifies it as a table of contents file. If the table of contents file is created in the same SPD Server operation, the timestamp for the backup file and the table of contents file are identical. The table of contents file does not have an SPD Server file extension. Unlike the backup file, the table of contents file is a regular system file and cannot be extended. The table of contents file size is constrained only by the native operating system's file size limit.

The table of contents file contains the following information for each table that is backed up:

- Columns 1–32 contain the table name. If the file is a domain ACL file, these columns contain the ACL name.
- Columns 33–232 contain the backup filename.
- Columns 233–250 contain the last full backup date, using the SAS datetime18. format.

- Columns 251–258 contain the incremental backup sequence number, since the last full backup. For example, the value 2 indicates that this is the second incremental backup since the last full backup.
- Columns 259–268 contain the number of rows that were backed up.
- Column 269 contains F for a full SPD Server backup, or I for an incremental backup.
- Columns 270–277 contain the number of indexes that were backed up during a full SPD Server backup. This field contains the value 0 if an incremental backup is being performed.
- Column 278 is a Boolean ACL file indicator. Column 278 contains a T if a domain ACL file is being backed up, or an F if a table is being backed up. If the ACL file indicator is set to T, columns 1–32 are configured for ACL names.

The table of contents file is formatted so that it can be used as a table of contents for a SAS backup file. The table of contents file uses the following SAS format:

```
format lastfull datetime18.;
 input @1 table $32. @33 bk_file $200.
 @233 lastfull datetime18. @251 inc_seq 8.
 @259 rows 10. @269 bk_type $1.
 @270 num_idx 8.,
 @278 acls $1.;
```

After you perform each SPD Server backup, you should append the resulting table of contents file to the table of contents file for the previous backup. This step saves the backup history and will assist you when you restore the tables.

For example, if you want to determine which backup files you need to restore a specific table, you could create the following SQL query, using the date of the last full backup:

```
select bk_file from foo.bkup_toc
where table = "dset"
and datepart(lastfull) >= 'ddmmmyyyy'd;
```

Using spdsbkup

Syntax for the spdsbkup Utility

```
spdsbkup -d <domain> -f <file> -h <host> [-hash][-s <service>][-u <user>]
[-p <password>][-t <mm/dd/yyyy:hh:mm:ss>][-fibfact <n>] [-r <count>]
[-a | -aonly][-n] [-q] [-v] [-nv6warn] [table...]
```

```
spdsbkup -full -d <domain> -f <file> -h <host > [-hash][-s <service>][-u <user>]
[-p <password>][-t <mm/dd/yyyy:hh:mm:ss>][-fibfact <n>][-r <count>]
[-a | -aonly][-n][-q][-v][-nv6warn][table...]
```

```
spdsbkup -inc -d <domain> -f <file> -h <host> [-hash][-s <service>][-u <user>]
[-p <password>][-t <mm/dd/yyyy:hh:mm:ss>][-fibfact <n>] [-r <count>]
[-a | -aonly][-n][-q][-v][-nv6warn][table...]
```

spdsbkup
: performs an incremental backup or full backup of server tables, depending on whether a full backup already exists.

  - If a table does not have a pre-existing full backup, spdsbkup performs a full backup of the table and sets the last full backup date.

- If the table does have a pre-existing full backup, spdsbkup performs an incremental backup. The incremental backup uses the latest full or incremental backup date as the beginning point for the incremental content change for the table.

spdsbkup -full
performs a full backup of server tables. All of the table rows and attributes (indexes, definitions, partition size, compression, and sorted) are backed up. After each full table backup, spdsbkup -inc resets the last full backup date for the table. See the -n option for dependencies.

spdsbkup -inc
performs only incremental backups of server tables.

- If a full backup (required for an incremental) is unavailable, spdsbkup -inc prints a warning message and the table is not backed up.
- If a full backup is available, spdsbkup -inc performs an incremental backup of the table using the later of the two dates: the date of the last full backup or of the last SPD Server incremental backup for the table.

Attribute changes to the table are not backed up in an incremental backup. Only the incremental changes for the rows since the last backup are backed up.

Options for the spdsbkup Utility

-a
specifies to include the domain ACL files in the backup.

-aonly
specifies to back up only the domain ACL files. No tables are backed up.

-d <domain>
specifies the server domain that you want to back up.

Note: The system that performs the backup must be able to access the physical path for the domain locally or through a network connection.

-f <file>
specifies the prefix filename for the backup data file. This filename is concatenated with _BK_`ddmmmyyyy_hhmmss`.0.0.0.spds9. The complete name identifies the file as an SPD Server backup file. If the backup file exceeds the system's file size limit, spdsbkup automatically extends the file and separates it into multiple backup files. Each file has a unique file extension (the 0.0.0 portion of the filenames is different).

-fibfact <n>
increases the File Information Block (FIB) metadata space by a factor of *n*, where *n* is greater than or equal to 2. The -fibfact option is necessary only if a backup fails due to insufficient FIB metadata space (fibspace). FIB metadata space shortages occur when the domain that is being backed up contains an unusually large number of tables.

-h <host>
specifies the host name of the name server. If you do not specify this option, the default value is SpdsName.

-hash
prints the hash sign (#) to STDOUT for each 256K block that is compressed and written to the backup file.

-help
prints the command-line usage syntax and option switch list for the spdsbkup utility.

-n
specifies that index information for server tables should not be saved when performing full backups. When the table is restored, the restore utility does not create indexes. The index itself is not backed up; only the definition of the index is backed up.

-nv6warn
suppresses the warning `Cannot back up v6 data set.` spdsbkup can back up only tables that were created with SPD Server 5.1 or later. If you try to back up earlier versions of server tables, a warning message is issued unless you specify the -NV6WARN option.

-p <password>
specifies the user password. Used in conjunction with the -u option.

-proj <directory>
specifies the domain project directory.

-q
runs spdsbkup in quiet mode, which includes only error messages and warning messages in the output during a backup operation.

-r <count>
specifies the number of times spdsbkup retries accessing a table that is not available because it is being updated. A table that is being updated cannot be backed up. The spdsbkup utility pauses 5 seconds, and then retries the table if it was unavailable during the previous access attempt. The default retry count is 1.

-s <service>
specifies the port number of the name server. If you do not specify this option, the default port number of the name server is 5400.

-t <datetimevalue>
specifies the date/time of the last full system backup for the table that is to be backed up.

When you specify the -t option with spdsbkup, the utility performs a full backup if no previous backup was made with the utility. Otherwise, the utility performs an incremental backup on the table from the date/time that the last backup was made or the date/time in the -t option, whichever is later.

When you specify the -t option with spdsbkup -inc, the utility prints a warning message if it encounters a table that was created after the specified date/time. The message states that the table cannot be backed up incrementally until a full backup of the table is completed. If no conflict is found, the utility performs an incremental backup on the table from the date/time the last backup was made or the date/time in the -t option, whichever is later.

You cannot use the -t option with spdsbkup -full.

-u <user>
specifies the SPD Server user ID of the user. Used in conjunction with the -p option.

-v
includes the full name of the backup file and the backup's table of contents file as part of a spdsbkup note. For an incremental backup, the spdsbkup note indicates the date/time from which the incremental backup on the table was made.

[table ...]
specifies a list of tables in the domain that you want to include in the backup. If you do not specify any tables, all of the eligible tables in the domain are backed up.

Note: The list of tables to be backed up must be the last option that you specify.

Return Values for the spdsbkup Utility

When spdsbkup exits, it generates a return value.

0
indicates that the backup was successful.

1
indicates that one or more tables were not backed up. Examine your SAS log for warning messages.

2
indicates that a critical error caused the process to terminate early. Examine your SAS log for warning and error messages.

Log Messages for the spdsbkup Utility

These spdsbkup messages can appear in your SAS log:

- `Successful Backup`

  If spdsbkup successfully backs up a table, it writes notes to STDOUT, unless the -Q option is specified. The notes include summary information such as the name of the table that was backed up, the number of rows that were backed up, and whether a full backup or an incremental backup was performed.

- `Warning: Table Cannot Be Backed Up`

  If spdsbkup cannot back up a table, it prints a warning message that states why the table could not be backed up.

- `Failed Backup: Error and Program Aborts`

  If the spdsbkup utility detects a serious failure condition, it stops the backup process and prints an error message that states the reason for the failure.

SPD Server Restore Utility (spdsrstr)

Restore Requirements

You must meet the following requirements before you can use the spdsrstr utility to restore a table:

- The client that performs the restore must have access to the server domain.
- The table to be restored must be identical to the table that was backed up. The name and creation date of the table to be restored must match the name and creation date of the backed up table.
- You must perform incremental table restores in the same order as the incremental backups were performed.

- The table must not have been modified between the incremental restore dates to ensure that the table is returned to the exact state at time of backup.
- The backup file (regardless of its file extension type) must be available.

If a table does not meet all of the criteria, spdsrstr prints a warning message to STDOUT and does not restore the table. If spdsrstr is restoring multiple tables, it restores only the tables that meet the restore criteria.

Usage Overview

The spdsrstr command can be used to validate SPD Server backup files in advance of performing a restore. It can also be used to create a backup file table of contents in addition to restoring files from a backup.

When validating backup files or creating a backup file table of contents, you need to provide only the source backup files as input. There is no need to connect to SPD Server. Use the -f or -e option to identify the backup files.

When performing a restore, specify options to connect to a server domain in addition to the backup files. To restore (or validate the restore) of a single table or specified tables, include the table option.

When performing a restore, spdsrstr re-creates indexes by default. Specify the -n option if you want to suppress index creation, and re-create the indexes with PROC DATASETS or PROC SQL after the table is fully restored instead.

Domain ACL files are not restored unless you specify the -a or -aforce option. Specify -aonly to restore only the domain ACL files, and nothing else.

Using spdsrstr

Syntax for the spdsrstr Utility

```
spdsrstr -d <domain> -h <host> {-f <fullfile>|-e <extfile>}[-hash][-r <count>]
[-a |-aforce][-aonly][-n][-q][-s <service>][-u <user>][-p <password>]
[-proj <directory>][table ...]
```

```
spdsrstr -t {-f <fullfile>|-e <extfile>}[table...]
```

```
spdsrstr -v -d <domain> -h <host> {-f <fullfile>|-e <extfile>}[-s <service>]
[-u <user>][-p <password>][-proj <directory>][table ...]
```

spdsrstr
restores all or selected tables from a backup file.

spdsrstr -t
prints a table of contents for a backup file. This table of contents file reports when the backup file was created and the type of backup that was performed. If a full backup was performed, the file includes the number of indexes. For each table that was backed up, the file specifies the name, backup sequence, and the number of columns and records that are in the table.

spdsrstr -v
verifies that all or selected tables from a backup file can be restored, but does not do the actual restore.

Options for the spdsrstr Utility

-a
restores the backed up domain ACL (access control list) files if they do not already exist.

-aforce
restores the backed up domain ACL files if they do not exist or overwrites the current files if they do exist.

Note: To ensure that the domain ACL files are consistent with the last file that was restored, use this option when you are restoring multiple files with the -e option.

-aonly
restores only the domain ACL files, and nothing else.

-d <domain>
specifies the server domain that you want to restore.

Note: The system that performs the restore must be able to access the physical path for the domain locally or through a network connection.

-e <extfile>
specifies the backup filename prefix as specified in spdsbkup that you use to restore all backup files in the directory with the name <*extfile*>_BK_**`ddmmmyyyy_hhmmss`**. 0.0.0.spds9. The backup files are restored in order from oldest to newest as determined by the **`ddmmmyyyy_hhmmss`** component of the filename.

-f <fullfile>
specifies the name of the backup file that contains the tables to restore.

Note: The filename must be the full filename (including its extension) that was created by the SPD Server backup utility.

-h <host>
specifies host name of the name server. If you do not specify this option, the default value is SpdsName.

-hash
specifies to print a hash sign (#) to STDOUT for each 256K compressed block that is read from the backup file.

-n
specifies that indexes should not be created for a full restore of a table that was backed up with index information.

-p <password>
specifies the user password.

-proj <directory>
specifies the domain project directory.

-q
runs spdsrstr in quiet mode, which includes only error and warning messages in the output during a restore operation.

-r <count>
specifies the number of times spdsrstr retries accessing tables that are not available during a restore operation because they were in Query or Update mode. The spdsrstr utility cannot restore a table if that table is in Query or Update mode when spdsrstr accesses it. The utility pauses 5 seconds, and then retries the table if it was in Query or Update mode. The default retry count is 1.

-s <service>
specifies the port number of the name server. If you do not specify this option, the default port number of the name server is 5400.

-u <user>
specifies the SPD Server user ID of the user. Used in conjunction with the -P option.

[table ...]
specifies the list of tables to restore from the backup file. If you do not specify any tables, all the tables in the file are restored.

Note: The list of tables must be the last option that you specify in your spdsrstr command.

Return Values for the spdsrstr Utility

When spdsrstr exits, it generates a return value.

0
indicates the restore was successful.

1
indicates one or more tables could not be restored. Examine your SAS log for warning messages.

2
indicates a critical error caused the process to terminate early. Examine your SAS log for warning and error messages.

Log Messages for the spdsrstr Utility

These spdsrstr messages can appear in your SAS log:

- `Successful Restore`

  If spdsrstr successfully restores a table, it writes notes to STDOUT, unless the -Q option is specified. The notes include summary information such as the name of the table that was restored, the number of rows that were restored, and whether the restore that was performed was a full restore or an incremental restore.

- `Warning: Table Cannot Be Restored`

  If spdsrstr cannot restore a table, it prints a warning message that states why the table could not be restored. No tables are restored after the failure.

- `Failed Restore`

  If the spdsrstr utility detects a serious failure condition, it stops the restore process and prints an error message that states the reason for the failure.

Sample Usage Scenarios

Overview

This section provides examples of potential backup scenarios and how to restore them. All of the scenarios use Sunday, February 3, 2016, as the starting date for the backup cycle. The weekly schedule includes the following backups:

- a full backup of the server domain Test every Sunday at 23:30
- an incremental backup of the domain at 23:30 on the remaining days of the week

One scenario shows how to perform both the full system backup and the incremental backups with spdsbkup. Another shows how to perform the full system backups with an operating system full backup utility and the incremental backups with spdsbkup.

Restore examples show how to restore a single server table, a server domain, and server indexes.

All of the examples use the command-line interface. For an example of how spdsbkup and spdsrstr are submitted from PROC SPDO, see “Example 4: Using PROC SPDO to Back Up and Restore Tables” on page 266.

Performing Full and Incremental Backups with spdsbkup

This scenario uses spdsbkup to perform a full backup of your domain once a week, and to perform incremental backups the rest of the week. The incremental backups also fully back up any newly created tables. The backups are performed on a server domain named Test.

1. On Sunday, February 3, 2016, at 23:30, issue the following spdsbkup command to do a full backup of the domain:

   ```
   spdsbkup -full -a -d test -h spdsname -s 5150 -f backup
   ```

   In the command, the following occurs:

   - The -full option specifies to perform a full backup.
   - The -a option specifies to include the domain ACL files in the backup.
   - The -d option specifies the domain to back up.
   - The -h option specifies the name server host.
   - The -s option specifies the name server port number.
   - The -f option specifies the prefix to use for the backup data file.

   The backup creates the backup data file backup_BK_05Feb2006_233000.0.0.0.spds9 and the backup table of contents file backup_TC_03Feb2016_233000. The backup file contains the full SPD Server backup for each table and for any ACL files in the domain. The table of contents file contains information about each table that was backed up.

2. Archive the SPD Server backup file and the source that are in the table of contents file into a SAS table of contents table.

3. On Monday night through Saturday night, use this spdsbkup command to perform incremental backups:

```
spdsbkup -a -d test -h spdsname -s 5150 -f backup
```

In this command, the following occurs:

- The spdsbkup command is specified without the -full or -inc options to let the utility determine whether a full or incremental backup is appropriate for each table in domain Test.
- The -a option specifies to include the domain ACL files in the backup, in case there are changes to it.
- The -d option identifies the domain.
- The -h and -s options identify the name server.
- The -f option identifies the prefix of the target backup file. Use the same prefix that you used for the full backup. Although the same filename prefix is specified each night, spdsbkup saves each night's backup to a different file and appends the date and time of the backup to the filename.

The command performs incremental backups of tables that were previously backed up, performs a full backup of tables that were created after the previous night's backup, and backs up any ACL files that are in the domain.

4. Archive the incremental data file and source in the table of contents file into a SAS table of contents table.

Using a System Utility to Perform Full Backups

This scenario performs a full system backup of the domain Test by using operating system utilities once a week. It then uses SPD Server to perform a domain back up on the remaining nights.

Advantages to using system full backups over the SPD Server utility to do a full backup are as follows:

- A system utility does not parse the data set. Therefore, this type of backup usually runs faster than the SPD Server utility when you are doing a full backup.
- System utilities often write directly to tape storage media. In contrast, the SPD Server utility first writes backup data to a file on the hard drive, and then the backup file is usually backed up to tape.

1. On Sunday, February 3, 2016, at 23:30, run the SPD Server list utility SPDSLS with the -L option to produce a listing of the tables that belong to the domain Test.

   ```
   spdsls -l -a <physical_path_of_domain>
   ```

2. Perform the full backup with the system utility.
3. On Monday, February 4, 2016, at 23:30, run spdsbkup to perform the incremental backups as follows:

   ```
   spdsbkup -d test -h spdsname -s 5150 -t 02/04/2008:23:30:00 -f backup
   ```

   In the command, the following occurs:

   - The spdsbkup command is specified without the -full or -inc options to let the utility determine whether a full or incremental backup is appropriate for each table in domain Test.
   - The -d option identifies the domain.
   - The -h and -s options identify the name server.

- The -t option sets the last full backup date to the previous night for the Test tables. The utility performs an incremental backup of tables that have changed since the last full system backup. And it performs a full backup of tables that were created after the last full system backup was performed.
- The -f option specifies a prefix for the target backup file.

The utility creates the backup data file backup_BK_04Feb2016_233000.0.0.0.spds9 and a backup table of contents file backup_TC_04Feb2016_233000. The backup file contains incremental changes for tables that were modified after 23:30:00 on February 3, 2016, and full backups of tables created after 23:30:00 on February 4, 2016. Only the tables that were modified or created since the date specified in the -t option are included in the backup file. The table of contents file contains information about each table that was either incrementally or fully backed up.

4. Archive the SPD Server backup file and source in the table of contents file into a SAS table of contents table.
5. On Tuesday night through Saturday night, use the SPD Server backup utility to do incremental backups of previously backed up tables and full backups of the newly created tables with the following command:

   ```
   spdsbkup -d test -h spdsname -s 5150 -f backup
   ```

   In the command, note that a last full backup date is not specified for the remaining week's incremental backups. The SPD Server backup utility performs incremental backups of tables that were previously backed up and full backups of tables that were created since the previous night's backup. Although the same filename prefix is specified each night, spdsbkup saves each night's backup to a different file and appends the date and time of the backup to the filename.
6. Archive the incremental data file and source in the table of contents file into a SAS table of contents table.

Restore a Single SPD Server Table

Use the following steps to restore a table that was accidentally deleted from the domain Test on Friday, February 8, 2016.

1. If the table was backed up fully by the operating system backup utility, use the system restore utility to restore the table. (Restore the table to its last full backup image, which was taken on February 3, 2016.) If the table was backed up fully by the spdsbkup, skip this step.
2. Run a SAS query on the backup table of contents table. In this example, the table is named Bkup.toc.

   ```
   select bk_file from foo.bkup_toc
   where domain = "test"
   and table = "results"
   and dttime >= '03Feb2016:23:30:00'd;
   ```

   The query results indicate which SPD Server backup files are required to restore the table to its last full backup state.
3. Gather the archived SPD Server backup files and any extensions that are required to restore the table.
4. Run spdsrstr on each sequential SPD Server backup file to restore the table. The order runs from the oldest backup date to the most recent backup date. The example table was backed up fully using the SPD Server backup utility on Sunday, February

3, 2016. The table was then backed up incrementally on Tuesday, February 5, and on Thursday, February 7. The following order of the statements is required to restore the table:

```
spdsrstr -d test -h spdsname -s 5150 -f backup_BK_03Feb2016_233000.0.0.0.spds9
results

spdsrstr -d test -h spdsname -s 5150 -f backup_BK_05Feb2016_233000.0.0.0.spds9
results

spdsrstr -d test -h spdsname -s 5150 -f backup_BK_07Feb2016_233000.0.0.0.spds9
results
```

In the commands, note the following:

- Each command specifies a different backup filename in the -f option. The files are uniquely identified by date.
- “Results” is the name of the table that is being restored.

TIP You can also use the -e option of spdsrstr and restore all of the files with one command:

```
spdsrstr -d test -h spdsname -s 5150 -e backup results
```

Note: When you restore a single table, you do not need to restore the ACL files, because they were not deleted.

Restore an SPD Server Domain

Use the following steps to restore a server domain named Test. This domain was lost due to a system media failure that occurred on Friday, February 15, 2016.

1. If the domain was backed up fully using the system backup utility, use the system restore utility to restore the domain Test to its state at the last full backup date of February 10, 2016. If the domain was backed up fully using the SPD Server utility, then skip this step.
2. Use SAS to run a query on the backup table of contents table Bkup_toc.

   ```
   select bk_file from foo.bkup_toc
   where domain = "test"
   and dttime >= '10FEB2016:23:30:00'd;
   ```

   The query results identify which SPD Server backup files are required to restore the domain.
3. Gather the archived SPD Server backup files required to restore the domain.
4. Use the SPD Server restore utility to restore the domain Test:

   ```
   spdsrstr -aforce -d test -h spdsname -s 5150 -e backup
   ```

   The -aforce option causes the domain ACLs to be updated for each restore file. Therefore, the latest backup of the ACLs is restored.

Back Up and Restore Table Indexes Using SPD Server Full Backups

When you perform an SPD Server full backup of a table, spdsbkup does not save the index data. It saves only the information necessary to re-create the indexes when the table is restored. Therefore, when you back up table indexes, the information that is saved does not require additional overhead or a lot of additional space.

If you must fully restore a table later, you can use one of the following two methods to restore the indexes:

- spdsrstr can re-create the indexes when the table is created (default behavior). In this method, the index is updated dynamically as each observation is added to the table.
- Use the -n option with spdsrstr. The -n option suppresses the creation of an index. After the table is fully restored, you can use PROC DATASETS or PROC SQL to re-create the indexes.

Back Up and Restore Table Indexes Using System Full Backups

Restoring the Index Dynamically

To restore a table's index from a full system backup dynamically, you must include the table index files in the full backup and restore of the table. To determine which index files to include, use spdsls with the -i *index* option. The output lists component files for each table in the domain that is to be fully backed up.

When you restore a table, you must first restore the table metadata, data, and index files from the last full backup archive. Then use spdsrstr to perform incremental restores. As the tables are restored, the indexes are dynamically updated to include any new or modified records.

This method trades the additional resources that are required for full backup of the table index files, which can be very large, against the potentially short time that might be required to restore them. You can restore indexes for a table that has not had any incremental changes after the system full backup by using a system full restore.

Re-creating the Index after the Table Is Restored

If you choose to re-create the index after the table is restored, you do not need to include the index files in the full backup of the table. When you run spdsls to list the component files for each table in the domain that you intend to back up, omit the -i *index* option. The spdsls utility includes a list in the output that excludes index files.

Note: If you do not save index information, you can experience problems when you attempt to fully restore the table. The table's metadata contains information about the index files that might be missing or out of date. As a result, the metadata no longer mirrors the contents of the table.

Before you can perform an incremental restore of the table, you must first repair the table metadata. To repair the metadata, use PROC DATASETS to modify the table and delete all of the indexes. Then run spdsrstr to restore the table. After the table is restored, use PROC DATASETS again to modify the table and create the indexes.

Chapter 28
Index Utility

Overview **289**

Invoking ixutil **289**

Syntax for the ixutil Utility **290**

Examples **292**
Create an Index 292
Retrieve Disk Usage and Fragmentation Statistics 292
Retrieve Index Distribution Statistics 293
Reorganize the Index 294
Review Disk Usage Statistics 295
Create a Join Index 295
Generate Join Index Statistics 295
Delete the Join Index 296

Overview

Because SPD Server tables can be very large, index creation is a CPU-intensive process. The index utility ixutil is provided to help you manage your indexes. The ixutil utility enables you to reorganize an SPD Server hybrid index to improve query performance and minimize disk space. The utility also prints the disk usage statistics and the contents of indexes. Using ixutil, you can also create, list, and delete join indexes.

Note: The ixutil utility is currently not supported for server tables in Hadoop domains.

Invoking ixutil

The ixutil utility can be invoked as a stand-alone utility, or it can be invoked in PROC SPDO by using the SPDSCMD statement. For more information about the PROC SPDO SPDSCMD statement, see“SPDSCMD Statement” on page 258.

The following environment variables must be established in order to invoke ixutil as a stand-alone utility. These environment variables are not needed if the utility is invoked in PROC SPDO.

For UNIX:

```
 INSTDIR=<!replace with SPD Server 5.3 Installation Directory!>
 SASVER=9.4
 SASHOME=<!replace with SPD Server 5.3 Installation Directory!>
 LD_LIBRARY_PATH=$INSTDIR/bin:$SASHOME/SASFoundation/$SASVER/sasexe
 export LD_LIBRARY_PATH
 LIBPATH=$INSTDIR/bin:$SASHOME/SASFoundation/$SASVER/sasexe
 export LIBPATH
 TKPATH=$INSTDIR/bin:$SASHOME/SASFoundation/$SASVER/sasexe:$SASHOME/SASFoundation
 /$SASVER/utilities/bin
 export TKPATH
 MSGPATH=$INSTDIR/misc/spds
 export MSGPATH
```

For Windows:

```
SET SASROOT=<!replace with SPD Server 5.3 Installation Directory!>
SET INSTALLDIR=<!replace with SPD Server 5.3 Installation Directory!>
SET MSGPATH=@INSTALLDIR@\spds\sasmisc
SET TKPATH=@INSTALLDIR@\bin;@SASROOT@\core\sasext
```

Syntax for the ixutil Utility

The ixutil utility can be used to perform several tasks, depending on the arguments that you specify:

```
ixutil -crejidx <dataset,column> <dataset,column> -libpath <physical-path>
-joinparts <number>

ixutil -deljidx <dataset,column> <dataset,column> -libpath <physical-path>

ixutil -lstjidx -libpath <physical-path> [-verbose]

ixutil -reorg <index1,index2,... | _all_> -dsn <dataset> -libpath <physical-path>

ixutil -runstats <index1,index2,... | _all_> -dsn <dataset> -libpath <physical-path>
[-maxruns <number>]

ixutil -statjidx <dataset,column> <dataset,column> -libpath <physical-path>

ixutil -stats <index1,index2,... | _all_> -dsn <dataset> -libpath <physical-path>
[-dist]

ixutil -help
```

ixutil -crejidx <dataset,column> <dataset,column> -libpath <physical-path> -joinparts <number>
creates a join index for a pair of tables that are in the same domain. The join index can be used to optimize parallel range joins. The columns must already be indexed. The -libpath argument specifies the physical path of the domain that contains the table. The -joinparts argument specifies the number of parallel join work units. Parallel join threads join the work units concurrently and then merge their partial results into the final result. The recommended number of parallel join work units is two times the number of processors.

ixutil -deljidx <dataset,column> <dataset,column> -libpath <physical-path>
deletes a join index. The -libpath argument specifies the physical path of the domain that contains the table.

ixutil -lstjidx -libpath <physical-path> [-verbose]
lists the join indexes that are in a domain. The -libpath argument specifies the physical path of the domain that contains the table.

ixutil -reorg <index1,index2,... | _all_> -dsn <dataset> -libpath <physical-path>
reorganizes the specified set of indexes in a table to reclaim wasted disk space and to aggregate the per-value segment lists. Reorganizing an index results in optimal disk usage and query performance. The -dsn argument specifies the table that contains the index. The -libpath argument specifies the physical path of the domain that contains the table.

ixutil -runstats <index1,index2,... | _all_> -dsn <dataset> -libpath <physical-path> [-maxruns <number>]
prints the run statistics for each index for the specified set of indexes that belong to a given table. Run statistics provide an indication of how the values of a particular index are sorted in relation to their observation numbers. By default, the ixutil utility run statistics display the ten longest runs in the table. A run is defined as the number of successive rows that contain the same index value. The -dsn argument specifies the table that contains the index. The -libpath argument specifies the physical path of the domain that contains the table. In addition, you can use the -maxruns argument to change the default setting of 10 runs to any integer in the range 1–100. You can use run statistics to construct more efficient BY and WHERE clause constructs for the table.

ixutil -statjidx <dataset,column> <dataset,column> -libpath <physical-path>
gathers statistics about the join index parts. The average join row percentage indicates the average number of rows that a parallel join work unit reads. For example, a row percentage of 75 indicates that the parallel join work unit uses 75% of the rows that it must read. The closer the percentage is to 100, the better the performance. The percentage increases as the distribution of the data for the join column becomes more sorted. The -libpath argument specifies the physical path of the domain that contains the table.

ixutil -stats <index1,index2,... | _all_> -dsn <dataset> -libpath <physical-path> [-dist]
prints the disk usage statistics and segment list fragmentation statistics for the specified set of indexes that belong to a given table. Each value in the index has a segment list. A value's segment list can become fragmented when the index is updated. An index that is highly fragmented can degrade query performance and waste disk space. The -dsn argument specifies the table that contains the index. The -libpath argument specifies the physical path of the domain that contains the table. Use the -dist option to include the distribution statistics with the index statistics.

TIP To improve performance and reclaim the wasted disk space, the index should be reorganized using the -reorg argument.

ixutil -help
prints a list containing the command-line usage and option switches for the ixutil utility.

Examples

Create an Index

Assume that a domain named MyDomain is assigned to the directory **`/spds`** on a machine named Spot. The SAS program creates a table named Test, and creates an index for column X on the table named Test.

```
libname my_data sasspds 'mydomain'
   server=spot.spdsname
   user='anonymous';

data my_data.test(index=(x));
   do i = 1 to 30000;
     x = mod(i,3);
     output;
   end;
run;

data my_data.test1;
   do i = 1 to 10000;
    x = mod(i,2);
    output;
   end;
run;

proc append
   base=my_data.test
   data=my_data.test1;
run;

proc sql;
   delete from my_data.test
     where x=1;
quit;
```

Retrieve Disk Usage and Fragmentation Statistics

Use the -stats argument to get the disk usage and segment list fragmentation statistics for the index.

```
> ixutil -stats test -dsn test -libpath /spds

SAS Scalable Performance Data Server 5.3(TS M0)
Build(Apr 26 2013, 11:50:08)
Index File Utility
Copyright (c) 1996-2013 by SAS Institute Inc, Cary NC 27513 USA

Statistics for Index X:
---------------------------------------------------
```

```
+--segment_size         = 8192
+--n_segments_in_tbl    = 5
+--n_values_in_index    = 2
+--n_vdeleted_values    = 1
+--percent_vdeleted     = 33.33
+--n_seglist_values     = 2
+--n_seglist_chunks     = 3
+--avg_chunks_per_list  = 1.00
+--idx_file_bytes       = 13304
+--idx_garbage_bytes    = 4272
+--percent_idx_garbage  = 32.11


Ixutil completed successfully
```

The statistics include the following information:

- the segment size of the index.
- the number of segments in the table.
- the number of distinct values for the index.
- the number of virtually deleted values (values that are no longer recognized by query indexes).
- the percentage of virtually deleted values.
- the number of values that require segment lists (a value that is in only one segment does not require a segment list).
- the number of segment list chunks for all values of the index.
- the average number of chunks for any value in the index.
- the size of the .idx file for the index. The .idx file maintains the value segment lists and bitmaps.
- the number of garbage bytes in the .idx file. Garbage bytes are the space in the file that was thrown away and cannot be reclaimed. Garbage bytes can result from deleting values, updating values, or appending values.
- the percentage of garbage bytes in the .idx file.

The average number of chunks for a segment list is a good indicator of the fragmentation level of the index. As this value increases, query performance can degrade when SPD Server must retrieve per-value information by making multiple reads of the index. If the average number of chunks exceeds 10, you should consider reorganizing the index to consolidate the segment lists.

The number of garbage bytes and the percentage of garbage bytes indicate the amount of unused disk space that is being consumed by the index. To conserve and consolidate disk space, consider reorganizing the index. Reorganizing the index frees up disk space when the garbage content is high.

Retrieve Index Distribution Statistics

Using the -stats argument, include the -dist option to get the index distribution statistics for the index:

```
> ixutil -stats x -dsn test -libpath /spds -dist
SAS Scalable Performance Data Server
```

```
   5.3 (TS M0) Build(Apr 26 2013, 11:50:08)
Index File Utility
Copyright (c) 1996-2013 by SAS Institute Inc, Cary NC 27513 USA

Statistics for Index X:
---------------------------------------------------
+--segment_size         = 8192
+--n_segments_in_tbl    = 5
+--n_values_in_index    = 2
+--n_vdeleted_values    = 1
+--percent_vdeleted     = 33.33
+--n_seglist_values     = 2
+--n_seglist_chunks     = 3
+--avg_chunks_per_list  = 1.00
+--idx_file_bytes       = 13304
+--idx_garbage_bytes    = 4272
+--percent_idx_garbage  = 32.11

+--Distribution Stats for Non Unique Values
+----minimum segments for all values = 4
+----maximum segments for all values = 5
+----average segments of any value = 4
+----average percentage of segments of any value = 90.00

Ixutil completed successfully
```

The distribution statistics include the following information:

- the number of unique values in the index
- the number of nonunique values in the index
- the minimum number of segments that any value is divided into
- the maximum number of segments that any value is divided into
- the average number of segments that all values are divided into
- the average percentage of segments that all values are divided into

You can use the distribution statistics to determine the effectiveness of the index. The index performs better if the distribution of the index values is clustered in a minimum number of segments. In general, the lower the average percentage of segments that all values are divided into, the better the index performs.

Reorganize the Index

Use the -reorg argument to reorganize the index to consolidate segment lists and retrieve unused disk space.

```
> ixutil -reorg x -dsn test -libpath /spds
SAS Scalable Performance Data Server
   5.3 (TS M0) Build(Apr 26 2013, 11:50:08)
Index File Utility
Copyright (c) 1996-2013 by SAS Institute Inc, Cary NC 27513 USA

Reorg for Index x:
Reorg successfully completed
Ixutil completed successfully
```

If you run the index utility program again to get the statistics, you will find that the segment lists for all of the values have been aggregated (the avg_chunks_per_list is 1.0) and that the unused disk space has been freed (the idx_garbage_bytes is 0). This outcome results in a proportional decrease in the size of the index file.

Aggregating the segment lists and compacting the index file minimizes the reads on the index for a query. The locality of segment data for an index key also increases. The combination of these processes results in the best query performance for the index.

Review Disk Usage Statistics

Use the -stats argument to review the index and segment list data, in order to view the improved performance statistics.

```
> ixutil -stats x -dsn test -libpath /spds
SAS Scalable Performance Data Server
   5.3 (TS M0) Build(Apr 26 2013, 11:50:08)
Index File Utility
Copyright (c) 1996-2013 by SAS Institute Inc, Cary NC 27513 USA

Statistics for Index X:
-----------------------------------------
+--segment_size        = 8192
+--n_segments_in_tbl   = 5
+--n_values_in_index   = 2
+--n_vdeleted_values   = 0
+--percent_vdeleted    = 0.00
+--n_seglist_values    = 2
+--n_seglist_chunks    = 2
+--avg_chunks_per_list = 1.00
+--idx_file_bytes      = 9008
+--idx_garbage_bytes   = 0
+--percent_idx_garbage = 0.00
```

Create a Join Index

Assume that server tables are in a domain in the directory **`/tmp`**. A user has created two tables, Table1 and Table2, that can be joined on the column ID. An index exists on the column ID for both tables. A join index is created on the tables to allow a parallel range join on column ID.

Use the -crejidx argument to create the join index.

```
> ixutil -crejidx Table1,ID Table2,ID
  -libpath /tmp
  -joinparts 4;
```

Generate Join Index Statistics

Obtain statistics on the join index that you created by using the -statjidx argument. The statistics are printed for each join range of the index and for the overall index. The range statistics identify each range (sobs=starting observation, eobs=ending observation), the number of unique join keys that exist in the range, and the number of keys that are joined in the range for each table.

```
> ixutil -statjidx Table1,ID Table2,ID
  -libpath /tmp

SAS Scalable Performance Data Server
   5.3 (TS M0) Build(Apr 26 2013, 11:50:08)
Index File Utility
Copyright (c) 1996-2013 by SAS Institute Inc, Cary NC 27513  USA


Stat of Join Index Table1.jdxid.table2.jdxid.0.0.0.spds9: Nranges=4
----------------------------------------------------------------------
+-Range 0
+----<Table1,ID>: sobs=1 eobs=25000 (Sorted)
+-------unique_keys=25000, max_occurance=1
+-------obs=25000, joinobs=25000, rangepct=100.00
+----<Table2,ID>: sobs=1 eobs=10000 (Sorted)
+-------unique_keys=10000, max_occurance=1
+-------obs=10000, joinobs=10000, rangepct=100.00
+-Range 1
+----<Table1,ID>: sobs=25001 eobs=50000 (Sorted)
+-------unique_keys=25000, max_occurance=1
+-------obs=25000, joinobs=25000, rangepct=100.00
+----<Table2,ID>: sobs=-1 eobs=0
+-------unique_keys=0, max_occurance=0
+-------obs=2, joinobs=0, rangepct=  0.00
+-Range 2
+----<Table1,ID>: sobs=50001 eobs=75000 (Sorted)
+-------unique_keys=25000, max_occurance=1
+-------obs=25000, joinobs=25000, rangepct=100.00
+----<Table2,ID>: sobs=-1 eobs=0
+-------unique_keys=0, max_occurance=0
+-------obs=2, joinobs=0, rangepct=  0.00
+-Range 3
+----<Table1,ID>: sobs=75001 eobs=100000 (Sorted)
+-------unique_keys=25000, max_occurance=1
+-------obs=25000, joinobs=25000, rangepct=100.00
+----<Table2,ID>: sobs=-1 eobs=0
+-------unique_keys=0, max_occurance=0
+-------obs=2, joinobs=0, rangepct=  0.00

Table Table1, Column ID average range join row pct=100.00
Table Table2, Column ID average range join row pct= 25.00

Ixutil completed successfully
```

Delete the Join Index

Use the -deljidx argument to delete the join index.

```
> ixutil -deljidx Table1,ID Table2,ID
-libpath /tmp
```

```
SAS Scalable Performance Data Server
   5.3 (TS M0) Build(Apr 26 2013, 11:50:08)
Index File Utility
Copyright (c) 1996-2013 by SAS Institute Inc, Cary NC 27513  USA

Ixutil completed successfully
```

Parallel join work units are based on the ranges of the join keys. For example, range 0 joins ranges 1 through 100, range 1 can join range 101 to 200, and so on. Ranges can overlap observations if the tables are not sorted by the join key. Join keys result in table sorting. The nature of the join key determines how much sorting is performed on the table. The more extensive the table sorting performed on behalf of the join key, the fewer rows a range work unit needs to read in order to gather all of the rows in its range. The overall performance of the parallel join index depends on how well the table is sorted by the join key. The stronger the join key sort, the better the performance. If a range work unit has a range percentage of 0 for either table, then there are no rows in the table for that range, and that range is discarded by a parallel work thread.

Chapter 29
Domain List Utility

Overview **299**

Invoking the spdsls Utility **299**

Syntax for the spdsls Utility **300**

Options for the spdsls Utility **301**

Examples **302**
Use spdsls to List All Components in a Domain 302
Use spdsls to List Sizing Information and Table Information for a Domain 302
Use spdsls to List Related Files for a Specified Table File 303
Use spdsls to List Related Files for a Table File Whose
Domain Primary Path Has an Underscore 303

Overview

The domain list utility spdsls lists the contents of an SPD Server domain directory, or lists all other component files for a given SPD Server table component file. The list utility has three purposes:

- furnish a complete list of tables for each server domain that you want to include in a backup
- for a specified damaged or questionable component file, provide a list of all other component files for the table that might be affected
- provide information about the size of server tables.

Note: spdsls is currently not supported for server tables in Hadoop domains.

Invoking the spdsls Utility

The spdsls utility can be invoked as a stand-alone utility or with PROC SPDO by using the SPDSCMD statement. For more information about the PROC SPDO SPDSCMD statement, see “SPDSCMD Statement” on page 258.

The following environment variables must be established in order for this utility to be invoked as a stand-alone utility. Note that these environmental variables are not needed if the utility is invoked in PROC SPDO.

For UNIX:

```
INSTDIR=<!replace with SPD Server 5.3 Installation Directory!>
 SASVER=9.4
 SASHOME=<!replace with SPD Server 5.3 Installation Directory!>
 LD_LIBRARY_PATH=$INSTDIR/bin:$SASHOME/SASFoundation/$SASVER/sasexe
 export LD_LIBRARY_PATH
 LIBPATH=$INSTDIR/bin:$SASHOME/SASFoundation/$SASVER/sasexe
 export LIBPATH
 TKPATH=$INSTDIR/bin:$SASHOME/SASFoundation/$SASVER/sasexe:$SASHOME/SASFoundation
 /$SASVER/utilities/bin
 export TKPATH
 MSGPATH=$INSTDIR/misc/spds
 export MSGPATH
```

For Windows:

```
SET SASROOT=<!replace with SPD Server 5.3 Installation Directory!>
SET INSTALLDIR=<!replace with SPD Server 5.3 Installation Directory!>
SET MSGPATH=@INSTALLDIR@\spds\sasmisc
SET TKPATH=@INSTALLDIR@\bin;@SASROOT@\core\sasext
```

Syntax for the spdsls Utility

The spdsls utility can be used to perform several tasks, depending on the arguments that you specify:

```
spdsls -l [-o][-i][-a][-s][-v][-v8][-v6][-aonly] <LibraryPath> [Table...]
```

```
spdsls -c [-i][-o][-a][-v][-diresc <char>] <ComponentPath>
```

```
spdsls -info [-o][-v][-s][-v8][-v6][-verbose] <LibraryPath> [Table...]
```

```
spdsls -help
```

spdsls -l
lists all component files for the specified table in the specified server domain. If no table is specified, all tables in the server domain are listed. You can use the output list with any system full backup utility.

spdsls -c
for a specified SPD Server index or data partition component file (which is identified by a full path), lists all other component files for the table that contain the specified file. If you have a table file that is corrupted or that has been deleted, use this option to find all related component files that might be affected. For information about component filenames, see "Component File Pathnames" on page 198.

Note: If any directory names in the primary path of the domain contain an underscore, you must escape the underscore in the path specification. Otherwise, spdsls -c will not work. For more information, see "Use spdsls to List Related Files for a Table File Whose Domain Primary Path Has an Underscore" on page 303.

spdsls -info
lists information about the specified table in the specified server domain. This option provides one line of information about the table as a whole rather than a listing for each component of the table.

spdsls -help
prints a list containing the command-line usage and option switches for the spdsls utility.

Options for the spdsls Utility

-a
specifies to include the domain ACL files in the listing. The files contain the access control lists (ACLs) for any server table in the domain.

-aonly
specifies to include only the domain ACL files in the listing.

-diresc <char>
specifies a directory escape character.

-i
specifies to list the index files.

-n
specifies to list the number of component files.

-o
specifies to list the file owner.

-s
lists the size of the component file, in bytes. When you use this option with spdsls -info, the size of the accumulated component file is listed, in bytes.

-v
includes the version number in the listing.

-verbose
when specified with spdsls -info, the -verbose option includes detailed information about a server table. The information includes the number of observations in the table, the length of the observation, the size of the index segment, the partition size, and whether the table is compressed, encrypted, or is a cluster member.

-v6
lists only SAS 6.*x* data sets.

-v8
lists only SAS 8.*x* or SAS 9.*x* data sets.

<LibraryPath>
full path of an SPD Server LIBNAME directory.

<ComponentPath>
full path of a specified table component file.

[Table...]
specifies one or more tables to list. (If no table is specified, all tables in the server domain are listed.)

Examples

Use spdsls to List All Components in a Domain

This sample command executes spdsls with the -l, -i, -o, and -s options to display owner and size information for component files in the domain **`/bigdisk/sas/data/public`**.

```
spdsls -l -i -o -s /bigdisk/sas/data/public
```

```
SAS Scalable Performance Data Server 5.3  Build(Thu Feb  4 21:05:05 EST 2016) -
Domain List Utility
Copyright (c) 1996-2011 SAS Institute Inc., Cary, NC, USA, All Rights Reserved

ANONYMOU     31196 /bigdisk/sas/data/public/cars.mdf.0.0.0.spds9
               648 /bigdisk/sas/data/public/cars.dpf._bigdisk_sas_data_public.0.1.spds9
ANONYMOU     30588 /bigdisk/sas/data/public/a.mdf.0.0.0.spds9
              8000 /bigdisk/sas/data/public/a.dpf._bigdisk_sas_data_public.0.1.spds9
ANONYMOU     31956 /bigdisk/sas/data/public/trx.mdf.0.0.0.spds9
          16774912 /bigdisk/sas/data/public/trx.dpf._bigdisk_sas_data_public.0.1.spds9
          16774912 /bigdisk/sas/data/public/trx.dpf._bigdisk_sas_data_public.1.1.spds9
          16774912 /bigdisk/sas/data/public/trx.dpf._bigdisk_sas_data_public.2.1.spds9
          16774912 /bigdisk/sas/data/public/trx.dpf._bigdisk_sas_data_public.3.1.spds9
          16774912 /bigdisk/sas/data/public/trx.dpf._bigdisk_sas_data_public.4.1.spds9
          16774912 /bigdisk/sas/data/public/trx.dpf._bigdisk_sas_data_public.5.1.spds9
          16774912 /bigdisk/sas/data/public/trx.dpf._bigdisk_sas_data_public.6.1.spds9
           9430190 /bigdisk/sas/data/public/trx.dpf._bigdisk_sas_data_public.7.1.spds9
ANONYMOU     47328 /bigdisk/sas/data/public/ida.mdf.0.0.0.spds9
                80 /bigdisk/sas/data/public/ida.dpf._bigdisk_sas_data_public.0.4.spds9
              8192 /bigdisk/sas/data/public/ida.idxx._bigdisk_sas_data_public.0.4.spds9
             24576 /bigdisk/sas/data/public/ida.hbxx._bigdisk_sas_data_public.0.4.spds9
```

Use spdsls to List Sizing Information and Table Information for a Domain

This spdsls command specifies the -info, -s, and -verbose options to get size and verbose descriptive information about tables in the domain **`/bigdisk/sas/data/public`**.

```
spdsls -info -s -verbose /bigdisk/sas/data/public
```

```
SAS Scalable Performance Data Server 5.3  Build(Thu Feb  4 21:15:09 EST 2016) -
Domain List Utility
Copyright (c) 1996-2011 SAS Institute Inc., Cary, NC, USA, All Rights Reserved

 DPF_SIZE IDX_SIZE MDF_SIZE NUMOBS OBSLEN SEGSIZE   PARTSIZE CMP ENC CLM TABLE
      648        0    31196      9     72    8192 1073740032  NO  NO  NO CARS
     8000        0    30588   1000      8    8192 1073741824  NO  NO  NO A
126854574        0    31956 489786    259    8192   16774912  NO  NO  NO TRX_RESULT_SEG1
       80    32768    47328     10      8    8192   16777216  NO  NO  NO IDA
```

Use spdsls to List Related Files for a Specified Table File

This spdsls command specifies the -c and -i options and specifies the full path to a data partition file to list component files that are related to the specified file. The full path to the data partition file is **`/users/tuser/testdata/tbl1.dpf._users_tuser_test.2.1.spds9`**. In the path, **`/users/tuser/testdata`** is the actual path of the component file, **`Tbl1`** is the table name, and **`.dpf._users_tuser_test.2.1.spds9`** is an extension that SPD Server generates to associate related component files. The file extension specifies the file type and the domain primary path, replacing the directory delimiters with underscores, among other things.

```
spdsls -c -i /users/tuser/testdata/tbl1.dpf._users_tuser_test.2.1.spds9
```

```
SAS Scalable Performance Data Server 5.3  Build(Thu Mar  3 10:07:09 EST 2016) -
Domain List Utility
Copyright (c) 1996-2011 SAS Institute Inc., Cary, NC, USA, All Rights Reserved

/users/tuser/test/tbl1.mdf.0.0.0.spds9
/users/tuser/testdata/tbl1.dpf._users_tuser_test.0.1.spds9
/users/tuser/testdata/tbl1.dpf._users_tuser_test.1.1.spds9
/users/tuser/testdata/tbl1.dpf._users_tuser_test.2.1.spds9
/users/tuser/testdata/tbl1.dpf._users_tuser_test.3.1.spds9
/users/tuser/test/tbl1.idxy._users_tuser_test.0.1.spds9
/users/tuser/test/tbl1.hbxy._users_tuser_test.0.1.spds9
```

Use spdsls to List Related Files for a Table File Whose Domain Primary Path Has an Underscore

This spdsls command specifies the -c and -i options and specifies the full path to a data partition file that has an underscore in one of the directory names in its domain's primary path. The full path for this component file is **`/users/tuser/testdata/tbl2.dpf._users_tuser_test_tables.3.1.spds9`**. The primary path for the domain is **`/users/tuser/test_tables`**. When a directory in the primary path for a domain contains an underscore, escape the underscore in the component file as follows:

1. Enclose the full pathname in quotation marks.
2. Escape the underscore with a backslash in the *file extension* portion of the name.

In the example below, note the backslash in **`test\_tables.`** near the end of the path.

```
spdsls -c -i
"/users/tuser/testdata/tbl2.dpf._users_tuser_test\_tables.3.1.spds9"
```

```
SAS Scalable Performance Data Server 5.3  Build(Thu Mar  3 10:07:09 EST 2016) -
Domain List Utility
Copyright (c) 1996-2011 SAS Institute Inc., Cary, NC, USA, All Rights Reserved

/users/tuser/test_tables/tbl2.mdf.0.0.0.spds9
/users/tuser/testdata/tbl2.dpf._users_tuser_test_tables.0.1.spds9
/users/tuser/testdata/tbl2.dpf._users_tuser_test_tables.1.1.spds9
/users/tuser/testdata/tbl2.dpf._users_tuser_test_tables.2.1.spds9
/users/tuser/testdata/tbl2.dpf._users_tuser_test_tables.3.1.spds9
/users/tuser/test_tables/tbl2.idxy._users_tuser_test_tables.0.1.spds9
/users/tuser/test_tables/tbl2.hbxy._users_tuser_test_tables.0.1.spds9
```

Chapter 30
Directory Cleanup Utility

Overview of the Directory Cleanup Utility **305**

Wildcards and Pattern Matching with spdsclean **306**

Files Used or Affected by spdsclean **306**

Syntax for the spdsclean Utility **307**

Options for the spdsclean Utility **308**

Examples **309**
Cleaning WORKPATH Files on Your Server 310
Cleaning Residual Temporary Server Domain Files 310
Cleaning Specified Domains 310
Cleaning Other Domain File Classes 310
Cleaning WORKPATH and Server Domain Directories 311
Cleaning Log Files 311
Cleaning WORKPATH, Server Domain, and Log Files 311

Overview of the Directory Cleanup Utility

The directory cleanup utility, spdsclean, is located in the `InstallDir/site` directory of your SPD Server installation. You use spdsclean to perform routine maintenance functions on the following types of directories:

- directories that you use to configure SPD Server storage
- directories that SPD Server uses for working storage
- various system-specific directories that are designated for temporary files.

The spdsclean utility has a simple command-line interface. You can control the level of cleanup performed and the behavior of elements that are used in the utility by specifying options.

CAUTION:

Use the spdsclean command-line utility only when SPD Server is not running. Do not run spdsclean when the SPD Server is running. The directory cleanup utility does not ensure that files in the SPD Server cleanup area are not in use by others. Some cleanup actions can violate SPD Server file integrity. Such a violation can permit concurrent access to file structures that were not designed to support concurrent access.

Wildcards and Pattern Matching with spdsclean

Some spdsclean options, such as -domains, use wildcards and pattern matching functions. The spdsclean utility uses the following wildcard and pattern matching rules:

- Character strings must match the domain name from the libnames.parm parameter file. The match is not case sensitive.
- In the search pattern, the period (.) and question mark (?) characters find a wildcard match to any single character in a domain name in the libnames.parm parameter file.
- The asterisk (*) character terminates the pattern and finds wildcard matches to all remaining characters in the domain name in the libnames.parm parameter file.

For example, the -domains pattern **`?test*`** matches the domains ATEST1, ATEST123, ATESTXYZ, CTEST1, and so on, from a libnames.parm parameter file. The -domains pattern **`test*`** matches only the domain name TEST from the libnames.parm parameter file.

Note: When you use wildcard characters in a -domains pattern, follow the rules for your command shell (such as ksh) to ensure that these characters are passed to the spdsclean command. For example, a ksh command shell user needs to enclose the wildcard pattern in double quotation marks. The double quotation marks ensure that the wildcard pattern matching occurs relative to the spdsclean command. Here is an example:

```
spdsclean -domains "?test*"
```

You can also disable command shell globbing for the execution of the spdsclean command.

Files Used or Affected by spdsclean

The spdsclean utility uses the libnames.parm and spdsserv.parms files as input:

libnames.parm
: the libnames.parm parameter file establishes the names and file storage locations of SPD Server domains that are available on an SPD Server host. The domains are defined in LIBNAME= specifications. At a minimum, a LIBNAME= specification specifies the domain name and a file system in a PATHNAME= attribute. The administrator can also specify path options for SPD Server data files, index files, work files, and metadata files. For more information about the libnames.parm parameter file, see Chapter 8, “Configuring Server Domains,” on page 59.

spdsserv.parm parameter file
: The spdsserv.parm parameter file defines the SPD Server operating parameters. The WORKPATH= statement in this file lists the directories that SPD Server uses for transient or working disk storage.

The spdsclean utility operates on the following files:

ACL files
: When you create SPD Server access control lists (ACLs), hidden ACL files are created in the primary directory of the domain. The hidden files are named .spres11*

and .sppro11*. The hidden ACL files retain the state of the ACLs that were defined for the domain resources. Typically, you should not delete ACL files.

domain state file
: The domain state file is also known as .spdslib11. The domain state file retains the set of directory paths that are configured for the domain. The directory path information is stored as an ordered list for each of the following domain storage classes:

- METAPATH=
- DATAPATH=
- INDEXPATH=

As you make path assignments over the life of the domain, the new directories are appended to the end of the ordered lists for METAPATH=, DATAPATH=, and INDEXPATH= storage classes. The order of directories listed in the .spdslib11 file defines the order of data cycling and overflow sequencing for each of the respective classes.

residual lock file
: When SPD Server accesses a data resource or table that is within a domain, it creates a lock file. The local operating environment uses the locking mechanism to ensure that proper member-level locking is observed by all SPD Server processes that access the named data resource. If a server domainproxy process terminates unexpectedly, the residual lock files remain in the domain. Residual lock files do not cause problems when the files are accessed again because the lock belongs to the operating environment. The lock is cleared when the process terminates and does not depend on the presence of the file itself. However, unused residual lock files can accumulate and create clutter in your primary domain directory.

residual temporary file
: SPD Server creates temporary files when you create a new resource in a domain. If the SPD Server server domainproxy process terminates unexpectedly while you are creating a new file, the residual temporary files remain in the domain directories. These temporary files contain a leading dollar sign character ($) in the name, which prevents the residual temporary files from appearing in a PROC DATASETS directory listing. You should periodically remove old or abandoned residual temporary files that unexpected proxy process terminations created.

system-specific temporary files
: SPD Server uses pre-assigned directories (which vary by operating environment) that are designated for temporary files. The pre-assigned directories hold files, logs, and other temporary entities that SPD Server creates while it is running. SPD Server usually cleans up these temporary files when it exits. If SPD Server terminates abnormally, these temporary files might be left in the temporary directory. In UNIX operating environments, the temporary files usually appear in directories such as **`/tmp`** or **`/var/tmp`**. In Windows operating environments, the temporary files are usually stored in **`C:/TEMP`** (or wherever the user profile is configured to store temporary files).

Syntax for the spdsclean Utility

The spdsclean utility can be used to perform several tasks, depending on the arguments that you specify:

```
spdsclean -libnamefile <path-to-libnames.parm> [-acl][-all][-domains <domain,...>]
[-lib11|+lib11][-tmp|+tmp][-vdomain|+vdomain][-verbose|+ verbose]
```

```
spdsclean -logdir <path-to-logfile-directory> [-logAge <days>]
```

```
spdsclean -parmfile <path-to-spdsserv.parm> [-vwork|+vwork][-verbose|+ verbose]
```

spdsclean -libnamefile <path>
: runs a cleanup on the SPD Server environment that is defined in the libnames.parm parameter file. You must specify the full pathname to the libnames.parm parameter file as input. By default, the spdsclean utility deletes residual lock files that were left behind in the domain directory and removes any temporary (TEMP=yes) directories and files. These files and directories are deleted for all of the domains that are defined in the libnames.parm parameter file. The spdsclean utility does not remove ACL files or the domain state file. Use the -domains option with pattern matching to filter the domains that you want to clean in the libnames.parm parameter file. Options are available to include ACL files and the domain state file in the files to be deleted, and to suppress the default deletion of residual temporary files. For examples of how the -libnamefile option can be used, see "Cleaning Residual Temporary Server Domain Files" on page 310 , "Cleaning Specified Domains" on page 310, and "Cleaning Other Domain File Classes" on page 310.

spdsclean -logdir <path>
: specifies the path for SPD Server to use when cleaning server log files. SPD Server searches the specified log path directories for .spdslog files. When .spdslog files are found, SPD Server checks them for aging criteria. You specify the aging criteria by using the -logage option. When spdsclean finds server log files that have a creation date that is older than the number of days specified in -logage, spdsclean deletes the files. Files that are equal to or less than the specified age are retained. For an example of how the -logdir option is used, see "Cleaning Log Files" on page 311.

spdsclean -parmfile <path>
: runs cleanup on the SPD Server environment that is defined in the spdsserv.parm parameter file. You must specify the full pathname to the spdsserv.parm parameter file as input. The cleanup action empties all directory resources that are defined in the spdsserv.parm parameter file. For example, all files in the WORKPATH= path list are deleted. For an example of how -parmfile is used, see "Cleaning WORKPATH Files on Your Server" on page 310.

Options for the spdsclean Utility

-acl
: deletes ACL files in the domain path list. These files are the ones that have .spres11* and .sppro11* extensions. The files are deleted from all of the domains that you specify with the -server domain option.

Note: Deleting the ACL files does not grant broader access to a given resource. Deleting the ACL files restricts the access to the resource owner.

+acl
: disables the deletion of ACL files in the server domain path list. The default setting is +acl.

-all
: enables deletion of residual temporary files, ACL files, and the domain state file from the domain path list.

Interaction Specifying -all is equivalent to specifying -tmp, -acl, and -lib11.

-domains <dompat1[, dompat2,]>
specifies a list of domains. The list is a comma-separated list of domain names and wildcard matching patterns. The list builds the SPD Server domains from the libnames.parm parameter file when it is processed. Standard pattern matching rules and wildcards apply.

-lib11
enables the deletion of the domain state file, .spdslib11.

+lib11
disables the deletion of the domain state file. This is the default setting for the lib111 variable.

-logage <ageDays>
sets the age threshold, in days, for keeping .spdslog files in the SPD Server log directory when the -logdir option is specified. Files older than the threshold value are deleted. The default value for ageDays is 7.

-tmp
enables deletion of residual temporary files in the domain path list. Deletion is enabled by default.

+tmp
disables the deletion of residual temporary files in the domain path list.

-vdomain
enables logging of the resource cleanup process from domain directories.

+vdomain
disables logging of the resource cleanup process from domain directories. This setting is the default setting.

-verbose
enables logging of the resource cleanup process from WORKPATH, system workspace directories, and domain directories.

Interaction Specifying -verbose is equivalent to specifying -vwork and -vdomain.

+verbose
disables logging of the resource cleanup process from WORKPATH, system workspace directories, and domain directories.

Interaction Specifying +verbose is equivalent to specifying +vwork and +vdomain.

-vwork
enables logging of the resource cleanup process from WORKPATH and system workspace directories.

+vwork
disables logging of the resource cleanup process for WORKPATH and system workspace directories. This is the default setting.

Examples

For the following examples, assume that the ***`InstallDir/`*** value for your SPD Server is the directory **`/opt/spds45`**.

Cleaning WORKPATH Files on Your Server

The following spdsclean command cleans all of the files in the WORKPATH directory list that is designated by **`/opt/spds45/site/spdsserv.parm`**:

```
spdsclean -parmfile /opt/spds45/site/spdsserv.parm
```

If you want spdsclean to log the files that it deletes, add the -verbose option to the command.

```
spdsclean -parmfile /opt/spds45/site/spdsserv.parm -verbose
```

Cleaning Residual Temporary Server Domain Files

The following spdsclean command cleans all of the residual temporary files from all of the domains that are defined in the specified libnames.parm parameter file:

```
spdsclean -libnamefile /opt/spds45/site/libnames.parm
```

If you want spdsclean to log the files that it deletes, add the -verbose option to the command.

```
spdsclean -libnamefile /opt/spds45/site/libnames.parm -verbose
```

Cleaning Specified Domains

The following spdsclean command cleans all residual temporary files from the domain TRIAL99:

```
spdsclean -libnamefile /opt/spds45/site/libnames.parm -domains trial99
```

To add domain UJOE04 to be cleaned also, use the following command:

```
spdsclean -libnamefile /opt/spds45/site/libnames.parm -domains trial99, ujoe04
```

To clean all TRIAL9x domains and all domains that begin with UJOE from the specified server domainfile, use the following command:

```
spdsclean -libnamefile /opt/spds45/site/libnames.parm -domains trial9?, ujoe*
```

To log the domains that spdsclean processed and the files that were deleted from each domain, add the -verbose option to any of these spdsclean commands.

Cleaning Other Domain File Classes

The following spdsclean command cleans only the ACL files from domains that begin with UJOE that are defined in the specified libnames.parm parameter file. Because this command specifies the +tmp option, the deletion of residual temporary files is suppressed. To log the domains that were cleaned and the ACL files that were deleted, add the -verbose option.

```
spdsclean -libnamefile /opt/spds45/site/libnames.parm +tmp -acl -domains ujoe*
```

To clean domain state files from domains TRIAL9*x* for the specified libnames.parm parameter file, issue the following command:

```
spdsclean -libnamefile /opt/spds45/site/libnames.parm
 -domains trial9? -lib11 +tmp
```

To log the domains that were cleaned and the files that were deleted, add the -verbose option.

Cleaning WORKPATH and Server Domain Directories

The following spdsclean command cleans all of the WORKPATH files from the directory list specified by the -parmfile option. This command also cleans residual temporary files from domain directories specified by the -libnamefile option.

```
spdsclean -parmfile /opt/spds45/site/spdsserv.parm
 -libnamefile /opt/spds45/site/libnames.parm -verbose
```

Logging is enabled for the WORKPATH and domain directories and for the files that were deleted from each directory.

Cleaning Log Files

The following spdsclean command cleans the .spdslog files that are more than 7 days old from the specified log path directory:

```
spdsclean -logdir /opt/spds45/log
```

To keep log files that are older than 10 days from the date of creation, use the following command:

```
spdsclean -logdir /opt/spds45/log -logage 10
```

If you want to see the files that were deleted, add the -verbose option to the spdsclean command.

Cleaning WORKPATH, Server Domain, and Log Files

The following spdsclean command cleans WORKPATH files from the directory list specified by the -parmfile option, residual temporary files from domain directories in the libnames.parm parameter file specified by -libnamefile, and .spdslog files that are older than 7 days from the specified log path directory:

```
spdsclean -parmfile /opt/spds45/site/spdsserv.parm
 -libnamefile /opt/spds45/site/libnames.parm
 -logdir /opt/spds45/log -verbose
```

Chapter 31
Debugging Tools

Overview of SPD Server Debugging Tools **313**
SPD Server LIBNAME Statement Debug Option **313**
ALTPATH= 313
spdsserv.parm Debug Option **314**
ALTBINPATH= 314

Overview of SPD Server Debugging Tools

SPD Server includes debugging tools that are useful for system administrators. The debugging tools allow system administrators to create debug images and to evaluate test images that do not interfere with a pre-existing production SPD Server environment. The debugging tools are organized into LIBNAME statement options and spdsserv.parm options.

SPD Server LIBNAME Statement Debug Option

When you issue a LIBNAME statement in SPD Server, the following debug option is available:

ALTPATH=

The ALTPATH= option enables the use of an alternate binary path, which is defined by the ALTBINPATH= option in the spdsserv.parm parameter file. The ALTPATH= option does not search entities in the PATH environment variable. If ALTPATH= does not find the ALTBINPATH= option specified in the file, a login failure error is issued. The ALTPATH= option is useful if you want to load a non-production copy of SPD Server (for example, testing a fix) without replacing the production copy of SPD Server on a user basis.

Syntax

```
ALTPATH= YES | NO
```

Arguments

YES
: enables use of the alternate binary path that is defined on the ALTBINPATH= spdsserv.parm parameter file option.

NO
: disables the alternate binary path for the specified proxy if a binary path is present.

ALTPATH= Example

Issue a LIBNAME proxy that uses the alternate binary path that is defined on the ALTBINPATH= parameter file debug option:

```
LIBNAME mylib sasspds 'spdsdata'
  user='denettee' altpath=yes;
```

spdsserv.parm Debug Option

SPD Server provides the following spdsserv.parm option that you can use as a troubleshooting and debugging tool.

ALTBINPATH=

The ALTBINPATH= option specifies the path to an alternate executable binary file directory. An alternate binary file path enables you to load a non-production copy of SPD Server without replacing the production copy of SPD Server.

Syntax

The ALTBINPATH= server parameter file option is enabled when a LIBNAME statement that contains a valid ALTBINPATH= specification is issued.

```
ALTBINPATH= 'DirPath'
```

Recommended Reading

- *SAS Scalable Performance Data Server: User's Guide*
- *SAS Scalable Performance Data Server: Processing Data in Hadoop*
- SAS Technical Papers — SAS Scalable Performance Data Server

Glossary

access control list (ACL)
a list of users and permission types that each user has for a data resource such as a file, directory, or table.

ACL
See access control list.

authentication
See client authentication.

Base SAS
the core product that is part of SAS Foundation and is installed with every deployment of SAS software. Base SAS provides an information delivery system for accessing, managing, analyzing, and presenting data.

big data
information (both structured and unstructured) of a size, complexity, variability, and velocity that challenges or exceeds the capacity of an organization to handle, store, and analyze it.

block
a group of observations in a data set. By using blocks, thread-enabled applications can read, write, and process the observations faster than if they are delivered as individual observations.

Certificate Revocation List (CRL)
a list of revoked digital certificates. CRLs are published by Certification Authorities (CAs), and a CRL contains only the revoked digital certificates that were issued by a specific CA.

client authentication (authentication)
the process of verifying the identity of a person or process for security purposes. Authentication is commonly used in providing access to software, and to data that contains sensitive information.

component file
any of several file types in a logical file structure that is tracked and indexed as a single table. Each SPD Server table includes a metadata file (.mdf), at least one data file (.dpf), and might also include index files (.hbx or .idx).

compound WHERE expression
a WHERE expression that contains more than one operator, as in WHERE X=1 and Y>3. *See also* WHERE expression.

controller
a computer component that manages the interaction between the computer and a peripheral device such as a disk or a RAID. For example, a controller manages data I/O between a CPU and a disk drive. A computer can contain many controllers. A single CPU can command more than one controller, and a single controller can command multiple disks.

CPU-bound application
an application whose performance is constrained by the speed at which computations can be performed on the data. Multiple CPUs and threading technology can alleviate this problem.

CRL
See Certificate Revocation List.

data partition
a physical file that contains data and which is part of a collection of physical files that comprise the data component of a table. *See also* partition.

data resource
any of a collection of domains, tables, catalogs and other types of data that users (with permissions) can access with SPD Server.

directory cleanup utility (spdsclean)
a component of SPD Server that performs routine maintenance functions on directories.

distinguished name (DN)
a unique identifier of an entry in an LDAP network directory. In effect, a distinguished name is the path to the object in the directory information tree.

distributed data
data that is divided and stored across multiple connected computers.

distributed locking
provides synchronization and group coordination services to clients over a network connection. The service provider is the Apache ZooKeeper coordination service, specifically the implementation of the recipe for Shared Lock that is provided by Apache Curator.

DN
See distinguished name.

domain
for SPD Server, a specific directory of file storage locations. The SPD Server Administrator defines the domain in the libnames.parm parameter file and assigns a name. Users connect to the SPD Server domain by specifying the domain name, for example, in the LIBNAME statement for the SASSPDS engine.

dynamic cluster table
two or more SPD Server tables that are virtually concatenated into a single entity, using metadata that is managed by the SAS SPD Server.

dynamic locking

provides multiple users concurrent access to tables. Users can perform read and write functions, and the integrity of the table contents is preserved. Clients that use dynamic locking connect to a separate SPD user proxy process for each connection in the domain.

explicit pass-through

a form of the SQL pass-through facility that passes the user-written, DBMS-specific SQL query code directly to a particular DBMS for processing. *See also* implicit pass-through.

firewall

a set of related programs that protect the resources of a private network from users from other networks. A firewall can also control which outside resources the internal users are able to access.

format

See SAS format.

function

See SAS function.

I/O-bound application

an application whose performance is constrained by the speed at which data can be delivered for processing. Multiple CPUs, partitioned I/O, threading technology, RAID (redundant array of independent disks) technology, or a combination of these can alleviate this problem.

implicit pass-through

a form of the SQL pass-through facility that translates SAS SQL query code to the DBMS-specific SQL code, enabling the translated code to be passed to a particular DBMS for processing. *See also* explicit pass-through.

informat

See SAS informat.

JAR (Java Archive)

the name of a package file format that is typically used to aggregate many Java class files and associated metadata and resources (text, images, etc.) into one file to distribute application software or libraries on the Java platform.

Java Archive

See JAR.

Java Database Connectivity (JDBC)

a standard interface for accessing SQL databases. JDBC provides uniform access to a wide range of relational databases. It also provides a common base on which higher-level tools and interfaces can be built.

JDBC

See Java Database Connectivity.

LDAP (Lightweight Directory Access Protocol)

a protocol that is used for accessing directories or folders. LDAP is based on the X. 500 standard, but it is simpler and, unlike X.500, it supports TCP/IP.

libnames.parm file
an SPD Server parameter file that defines the domains by establishing the names and file storage locations for data resources. The file also serves as a tool for controlling access to the domains and for managing storage of SPD Server files.

light-weight process thread
a single-threaded subprocess that is created and controlled independently, usually with operating system calls. Multiple light-weight process threads can be active at one time on symmetric multiprocessing (SMP) hardware or in thread-enabled operating systems.

Lightweight Directory Access Protocol
See LDAP.

macro variable (symbolic variable)
a variable that is part of the SAS macro programming language. The value of a macro variable is a string that remains constant until you change it.

name server
an SPD Server server process that converts domain names to data storage locations.

national language support (NLS)
the set of features that enable a software product to function properly in every global market for which the product is targeted.

NLS
See national language support.

ODBC
See Open Database Connectivity.

Open Database Connectivity (ODBC)
an interface standard that provides a common application programming interface (API) for accessing data. Many software products that run in the Windows operating environment adhere to this standard so that you can access data that was created using other software products.

parallel execution
See parallel processing.

parallel I/O
a method of input and output that takes advantage of multiple CPUs and multiple controllers, with multiple disks per controller to read or write data in independent threads.

parallel processing (parallel execution)
a method of processing that divides a large job into multiple smaller jobs that can be executed simultaneously on multiple CPUs. *See also* threading.

parameter file
a document that contains the data that SPD Server needs to perform its functionality. SPD Server uses a libnames.parm parameter file and an spdsserv.parm parameter file.

partition

part or all of a logical file that spans devices or directories. A partition is one physical file. Data files, index files, and metadata files can all be partitioned, resulting in data partitions, index partitions, and metadata partitions, respectively. Partitioning a file can improve performance for very large tables. *See also* data partition.

pass-through facility

See SQL pass-through facility.

password database

registers users to enable access to SPD Server. The database stores each user ID and password, a user's ACL group memberships, authorization level, performance class, account expiration information, and server access records.

password database utility

a component that creates and manages the password database, and that enables users to access SPD Server. It is an interactive command-line utility that begins with the psmgr command.

primary path

the location in which metadata files are stored, and the default location for other component files. Typically, the other component files (data files and index files) are stored in separate storage paths in order to take advantage of the performance boost of multiple CPUs.

RAID (redundant array of independent disks)

a type of interleaved storage system that comprises multiple disks to store large amounts of data inexpensively. RAIDs can have several levels. For example, a level-0 RAID combines two or more hard drives into one logical disk drive. Various RAID levels provide differing amounts of redundancy and storage capability. Also, because the same data is stored in different places, I/O operations can overlap, which can result in improved performance. *See also* redundancy.

record-level locking

locking at the record level in a table or data set. The user who owns the lock has exclusive access to a single record, while other users can access other records in the same table or data set.

redundancy

a characteristic of computing systems in which multiple interchangeable components are provided in order to minimize the effects of failures, errors, or both. For example, if data is stored redundantly (in a RAID, for example), then if one disk is lost, the data is still available on another disk.

redundant array of independent disks

See RAID.

RLS

See row-level security.

row-level security (RLS)

a security feature that controls access to rows in a table in order to prevent users from accessing restricted data.

SAS format (format)
a type of SAS language element that is used to write or display data values according to the data type: numeric, character, date, time, or timestamp.

SAS function (function)
a type of SAS language element that is used to process one or more arguments and then to return a result that can be used in either an assignment statement or an expression.

SAS informat (informat)
a type of SAS language element that is used to read data values according to the data's type: numeric, character, date, time, or timestamp.

SAS Management Console
a Java application that provides a single user interface for performing SAS administrative tasks.

SAS Metadata Server
a multi-user server that enables users to read metadata from or write metadata to one or more SAS Metadata Repositories.

SAS Scalable Performance Data Server (SPD Server)
a server that restructures data in order to enable multiple threads, running in parallel, to read and write massive amounts of data efficiently.

sasroot
a representation of the name for the directory or folder in which SAS is installed at a site or a computer.

SASSPDS
the SAS engine that provides access to the SAS SPD Server.

scalability
the ability of a software application to function well and with minimal loss of performance, despite changing computing environments, and despite changes in the volume of computations, users, or data. Scalable software is able to take full advantage of increases in computing capability such as those that are provided by the use of SMP hardware and threaded processing. *See also* scalable software, server scalability.

scalable software
software that responds to increased computing capability on SMP hardware in the expected way. For example, if the number of CPUs is increased, the time to solution for a CPU-bound problem decreases by a proportionate amount. And if the throughput of the I/O system is increased, the time to solution for an I/O-bound problem decreases by a proportionate amount.

Secure Sockets Layer
See SSL.

serde
an interface that enables serialization or deserialization of one or more file formats.

server scalability
the ability of a server to take advantage of SMP hardware and threaded processing in order to process multiple client requests simultaneously. That is, the increase in

computing capacity that SMP hardware provides increases proportionately the number of transactions that can be processed per unit of time. *See also* threaded processing.

session

a single period during which a software application is in use, from the time the application is invoked until its execution is terminated.

SMP (symmetric multiprocessing)

a type of hardware and software architecture that can improve the speed of I/O and processing. An SMP machine has multiple CPUs and a thread-enabled operating system. An SMP machine is usually configured with multiple controllers and with multiple disk drives per controller.

sort indicator

an attribute of a data file that indicates whether a data set is sorted, how it was sorted, and whether the sort was validated. Specifically, the sort indicator attribute indicates the following information: 1) the BY variable(s) that were used in the sort; 2) the character set that was used for the character variables; 3) the collating sequence of character variables that was used; 4) whether the sort information has been validated. This attribute is stored in the data file descriptor information. Any SAS procedure that requires data to be sorted as a part of its process uses the sort indicator.

spawn

to start a process or a process thread such as a light-weight process thread (LWPT). *See also* thread.

SPD Server

See SAS Scalable Performance Data Server.

SPD Server STARJOIN Facility (STARJOIN Facility)

a component of the SPD Server SQL Planner that optimizes N-way star schema joins for qualified SPD Server tables.

SPDO procedure

the operator interface for SPD Server. The procedure defines and manages SPD Server ACLs, defines row-level security for tables, manages proxies, defines and manages cluster tables, refreshes server parameters and domains, performs table management functions such as truncating tables, and executes SPD Server utilities from a central point.

SPDSBASE process

accesses or creates SPD Server resources for users. Several SPDSBASE processes can be active simultaneously in an SPD Server installation, handling work requests for different users or different SAS sessions. The SPDSBASE process can take on the role of either an SPD Server user proxy, an SPD Server SQL proxy, or an SPD Server SQL user proxy.

spdsclean

See directory cleanup utility.

spdsserv.parm file

an SPD Server parameter file that defines the server configuration and performance parameters that control processing behavior and use of resources.

SQL pass-through facility (pass-through facility)
the technology that enables SQL query code to be passed to a particular DBMS for processing. *See also* record-level locking.

SQL query rewrite facility
examines SQL queries to optimize processing performance. When an SPD Server user submits SQL statements that contain subexpressions, the SQL query rewrite facility optimizes the SQL query when possible.

SSL (Secure Sockets Layer)
an encryption protocol for securing client-server communication. *See also* Transport Layer Security.

star schema
tables in a database in which a single fact table is connected to multiple dimension tables. This is visually represented in a star pattern. SAS OLAP cubes can be created from a star schema.

STARJOIN Facility
See SPD Server STARJOIN Facility.

symbolic variable
See macro variable.

symmetric multiprocessing
See SMP.

thread
the smallest unit of processing that can be scheduled by an operating system.

thread-enabled operating system
an operating system that can coordinate symmetric access by multiple CPUs to a shared main memory space. This coordinated access enables threads from the same process to share data very efficiently.

thread-enabled procedure
a SAS procedure that supports threaded I/O or threaded processing.

threaded I/O
I/O that is performed by multiple threads in order to increase its speed. In order for threaded I/O to improve performance significantly, the application that is performing the I/O must be capable of processing the data rapidly as well. *See also* I/O-bound application, thread.

threaded processing
processing that is performed in multiple threads in order to improve the speed of CPU-bound applications. *See also* CPU-bound application.

threading
a high-performance technology for either data processing or data I/O in which a task is divided into threads that are executed concurrently on multiple cores on one or more CPUs.

time to solution
the elapsed time that is required for completing a task. Time-to-solution measurements are used to compare the performance of software applications in

different computing environments. In other words, they can be used to measure scalability. *See also* scalability.

TLS

See Transport Layer Security.

Transport Layer Security (TLS)

the successor to Secure Sockets Layer (SSL), a cryptographic protocol that is designed to provide communication security. TLS uses asymmetric cryptography for authentication and confidentiality of the key exchange, symmetric encryption for data/message confidentiality, and message authentication codes for message integrity.

WHERE clause

a syntax string that is composed of the keyword WHERE, followed by one or more WHERE expressions. A WHERE clause defines the conditions to be used for selecting observations in a data set. *See also* WHERE expression.

WHERE clause planner

uses factors of cardinality and distribution to calculate relative processor costs of various WHERE clause options. The SPD Server WHERE clause planner avoids computation-intensive operations and uses simple computations where possible.

WHERE expression

is a syntax string within a WHERE clause that defines the criteria for selecting observations. For example, in a membership database, the expression "WHERE member_type=Senior" returns all senior members. *See also* compound WHERE expression, WHERE processing.

WHERE processing

a method of conditionally selecting rows for processing by using a WHERE expression. *See also* WHERE expression.

workspace tables

a list of paths that contain temporary SPD Server work tables and temporary intermediate files that are associated with the declared domain.