SAS® Scalable Performance Data Server 5.1

Administrator's Guide

Second Edition

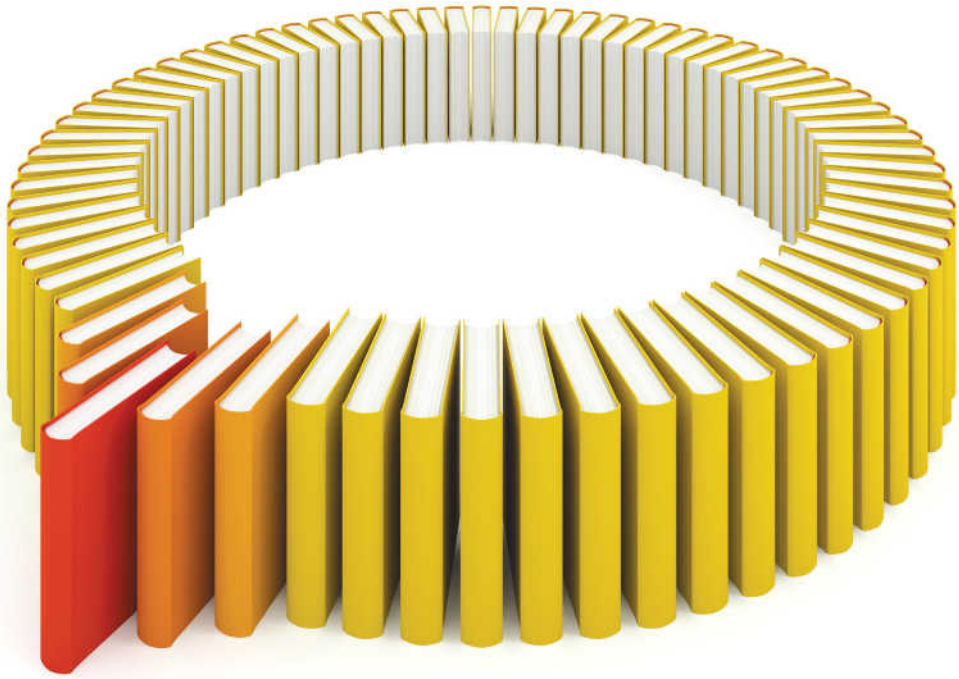
SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2013. *SAS® Scalable Performance Data Server 5.1: Administrator's Guide, Second Edition*. Cary, NC : SAS Institute Inc.

SAS® Scalable Performance Data Server 5.1: Administrator's Guide, Second Edition

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

Electronic book 1, December 2013



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

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

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

Contents

Part 1

Product Notes

Chapter 1

What's New in SAS Scalable Performance (SPD) Server 5.1

What's New in SPD Server 5.1?

SAS 9.4 includes a new SPD Server engine client that can connect with the SPD Server 5.1 server. SPD Server 5.1 also offers expanded support for regulatory, IT, and end user features such as the following:

- Enhanced (AES-256) encryption for data at rest
- Windows 64-bit server support
- SQL performance enhancements
- New SPD Server cluster features introduce enhanced operation, including functions for Cluster Remove, Cluster Replace, and online cluster management.
- New dictionary table types in SQL enables you to view SPD Server cluster and system information.
- Join planner enhancements and hash join optimizations.
- Now supports up to 32 groups per user.
- Validates the paths to the start-up files libnames.parm and spdsserv.parm.
- Enabled prxmatch() support for PERL regular expression pattern matching and support in SPD Server WHERE clauses.
- Information is generated that enables SPD Server administrators to identify the SPD Server user who spawned a given spdsbase processes.
- Binary compression.

Part 2

SAS Scalable Performance Data (SPD) Server Installation

Chapter 2

SAS Scalable Performance Data (SPD) Server Pre-Installation and System Requirements Guide

Operating System Requirements and Tuning for 64-Bit SPD Server

The operating system level requirements for SPD Server 5.1 on a given platform are the same requirements needed for SAS 9.4. For complete information about platform requirements for SAS SPD Server on SAS 9.4, see the information available on SAS Institute's external website at **`www.sas.com/partners/directory/index.html`** .

SPD Server Operating System Resource Configuration

Some systems limit either the number of concurrent processes a user can own at one time. Some systems limit the number of concurrent files a user can have open at one time. If you are using an operating system that enforces one or more of these limits, you should consider the following configuration updates:

Number of Processes per Operating System User ID
: SPD Server installation requires eight concurrent processes. Furthermore, each SPD Server User ID 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 8.

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).

  To calculate the maximum number of open files per process for an SPD Server dynamic cluster, the calculated number of open files per process for a single table is multiplied by the number of members in the dynamic 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 dynamic 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

Before You Install: Precautions and Required Permissions

Review the following precautions and required permissions:

- Use a UNIX user ID other than root to run your production SPD Server environment. The UNIX user ID of the SPD Server installation should be the UNIX user ID of the

SPD Server administrator. For more information and a list of options to use when you are configuring SPD Server, see "Notes for SPD Server Administrators" on page 15.

- Install SPD Server in a location that is adequately mirrored and backed up to assure reliability.
- General familiarity with UNIX is required to install SPD Server. At a minimum, you should be familiar with basic UNIX shell entities (such as sh, csh, and ksh), Bourne shell scripts, and modifying files using a UNIX text editor.
- To create the installation directory for SPD Server, you need appropriate access permissions on the file system on which you install the server software. The owner of the SPD Server installation directory should be the UNIX user ID of the SPD Server administrator.
- If you want SPD Server clients to connect to the SPD Server host using name services instead of specifying port numbers at invocation, you need Write access to your server machine's **`/etc/inet/services`** or **`/etc/services`** file. Name services require you to define registered ports that use the services file appropriate to your machine.
- If your SPD Server clients access the SPD Server host using name services instead of specifying port numbers, you need Write access to the services files on the clients, in the path **`/etc/services`** or **`/etc/inet/services`**.

SPD Server UNIX Installation Packing List

Directory names in the packing list are subdirectories of your SPD Server host installation directory, whose path is represented by **`InstallDir/`**.

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

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

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

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

The **`InstallDir/site/`** directory is a storage directory for a user's site-specific customization of the sample SPD Server start-up and configuration files. No SPD Server files are delivered in this directory; it is for customer use only.

SPD Server Quick UNIX Installation Guide

This section outlines a quick installation of SPD Server to validate and test your installation.

Follow the instructions on your SAS software order. 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 SAS Scalable Performance Data Server product for installation. You also need to include the SAS Scalable Performance

Data Client, the SAS Management Console, and the SAS Scalable Performance Data Server Plug-in for SAS Management Console in your product install selection.

Configuring SPD Server Host Software for Your Site

During installation of SAS Scalable Performance Data Server, you will be prompted to configure the SPD Server connection port, as well as the SPD Server SNET server connection port. The SPD Server Data server connection port connects the SPD Server LIBNAME client engine to the SPD Server Data server. The SPD Server SNET connection connects a JDBC client to the SPD Server SNET server. You use the SPD Server data port to make connections from the SAS SPD Server LIBNAME engine to the SPD Server host, as well as to the SPD Server SNET server port. The SPD Server 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. You can change the default port assignments to port numbers that are available on your system. If you are running within a firewall, see "How do I know which ports must be surfaced through an Internet firewall?" in Appendix 2 of *SAS Scalable Performance Data Server: User's Guide* or see "Using SAS Scalable Performance Data (SPD) Server with an Internet Firewall" on page 175 for more detailed information.

Completing the Quick Start Configuration

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 installation of your server:

1. Verify that the WORKPATH statement in your spdsserv.parm file is valid. If the WORKPATH statement is incorrect, edit your spdsserv.parm file.
2. Verify that the pathname for libname=tmp in the libnames.parm file is valid. If the pathname is not correct, edit the libnames.parm file.
3. Initialize the SPD Server Password Manager Database by invoking the pwdb script by 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 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.
   e. Assign yourself an authorization level of 7 for maximum permissions, and then assign yourself to the admingrp group. Press Enter to use default settings for the remaining prompts.
   f. To change your password to a permanent password, enter **`chgpass`** at the **`command>`** prompt and follow the instructions.
   g. To exit the password manager utility, enter **`quit`** at the **`Enter command>`** prompt.

4. Start SPD Server by executing the rc.spds script. Use the UNIX **PS** command to verify that SPD Server is running.

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

   - spdserv
   - 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, and then restart the software by using the **rc.spds** script.

Testing the SPD Server Quickstart Installation

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

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

```
%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 "Installdir"/samples/verify.sas"

/* include "Installdir"/samples/verptsql.sas */
/* to verify SQL access to SPD Server.       */

%inc "Installdir"/samples/verptsql.sas";
endsdas;
```

:

Potential Problems with Test Verification

Your LIBNAME statement to SPD server 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 SAS 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 5.1 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 5.1, verify that you are running SAS 9.4 with the SAS Scalable Performance Data Server client.

Customizing Your SPD Server Installation

After you complete the SPD Server quick start, you can customize the installation for your site. Shut down SPD Server by using the **`Installdir/site/killrc`** script to begin updating your installation.

Logging

The **`rc.spds`** script assumes that you want to keep the logs from messages that are written to STDOUT or STDERR of the spdsnsrv (SPD Server name server) and spdsserv (SPD Server host) processes. The **`rc.spds`** script variable LOGDIR= defines the directory where these logs are stored.

If you do not want to keep these logs, change the value for LOGDIR=, and the **`rc.spds`** script uses **`/dev/null`**. If you want to keep the logs in another location besides **`"InstallDir"/log`**, change the value of LOGDIR=.

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.

The DSRVFILE= and DSRVTIME= spdsserv options, NSRVFILE= and NSRVTIME= spdsnsrv options, and SNSFILE= and SNSTIME= spdssnet options have the following default values in the rc.spds script to define the log name and recycle time:

DSRVFILE=spdsserv
: specifies the spdsserv process log file prefix

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

NSRVFILE=spdsnsrv
: specifies the spdsnsrv process log file prefix

NSRVTIME=00:00
specifies a recycle time of midnight

SNSFILE=spdsnet
specifies the prefix of the spdssnet process log file

SNSTIME=00:00
specifies the time of midnight

If you want SPD Server to disable automatic log filename generation and cycling, change those settings to empty pointers.

Audit File Facility

The **rc.spds** script enables you to use the SPD Server audit file facility. The audit file facility is not enabled by default. Use the following script variables to configure the SPD Server audit file facility:

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.

Note: When AUDDIR= and AUDFILE= are set, proxy audit file creation is enabled. When AUDDIR= and AUDFILESQL= are set, SQL audit file creation is enabled. If AUDTIME= is set, automatic audit file cycling occurs at the specified time of day.

For more information about SPD Server auditing, see "SAS Scalable Performance Data (SPD) Server Auditing" on page 179.

User Password and Parameter Files

The **rc.spds** script assumes that you keep your spdsserv.parm parameter file and your SPD Server user password file in the **"InstallDir"/site** directory. If you do not keep the files in this location, you need to change your ACLDIR= and PARMDIR= assignments.

Accessing SPD Server through a Registered Port

If you want to access 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, *sp45name*), or you can directly include the SPD Server port number in your SAS statements.

SPD SNET Server

The **rc.spds** script assumes that you want the spdssnet process running for clients that are accessing SPD Server data via a JDBC client. If you do not support this access, use the -nosnet option with the **rc.spds** script to disable start-up of the SPD SNET server.

Adding Users to the SPD Server psmgr Database

See the section "Managing SAS Scalable Performance Data (SPD) Server Passwords and Users" on page 157 for more information about adding users to the SPD Server psmgr database.

Authenticating SPD Server User Passwords

SPD Server user passwords can be authenticated by the SPD Server psmgr utility, or by an LDAP server (such as Microsoft Active Directory, Sun Java System Directory Server, or OpenLDAP).

LDAP authentication integrates with the SPD Server password facility and provides a centralized approach to user ID and password management. SPD Server clients that use LDAP authentication should have user accounts that are managed by the authenticating LDAP server. The user ID and password information must be stored on an LDAP server that the SPD Server can access. The user ID must be entered into the SPD Server password database through the psmgr utility or via the SAS Management Console utility. These user ID requirements exist in order to ensure that all SPD Server user information is recorded and properly propagated.

When a client uses LDAP authentication to connect to SPD Server, the LDAP server that is configured in the SPD Server's parameter file performs the authentication. After the client is verified, SPD Server uses the client's password database record for all other SPD Server operations.

For more information about SPD Server LDAP authentication, see"Overview of LDAP Authentication" on page 167.

Customizing SPD Server Configuration

To customize configuration of the **"Installdir"/site/spdsserv.parm** parameter file and the **"Installdir"/site/libnames.parm** LIBNAME file for your installation, see Part 4, Chapters 6–13, in *SAS Scalable Performance Data Server: Administrator's Guide*.

Notes for SPD Server Administrators

The SPD Server administrator performs the maintenance and configuration functions for SPD 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 SPD Server users do not need UNIX

login IDs. Other users' access to SPD Server data resources is controlled via SPD Server user IDs via the SPD Server password facility.

Administrate your SPD Server environment by using the same UNIX user ID that was used to install SPD Server on the host machine. The user ID should also be the SPD Server administrator's user ID. The common user ID minimizes potential problems with file ownership and system access permissions on the server machine. You add SPD Server access controls to the resources that were created with SPD Server by using SPD Server user IDs and SPD Server ACLs. The SPD Server user IDs and ACLs provide fine-grained access controls for SPD Server data resources.

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

SPD Server User IDs

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

Each SPD User is given their own SPD Server user ID and password. The user ID and password are needed to complete the LIBNAME connection to SPD Server. All resources that a user creates are owned by the user. An SPD Server user can access only resources that that user created, or resources that another SPD Server user grants them access to via SPD 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 SPD Server user.

Troubleshooting

Key information for SPD Server troubleshooting can be found in the SPD Server name server log and in the SPD Server host process log files. Those two log files enable you to reconstruct SAS interactions with SPD 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 -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 -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:

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 5.1 while engine version is <version>.
This error indicates you are trying to connect to SPD Server with an old client. Verify that you are running SAS 9.4 with 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.

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. You must restart SPD Server to use the new license. You must edit your **`rc.spds`** LICFILE= variable name so that it reads the name of the new license file.

Upgrading and Reinstalling SPD Server

If you need to reinstall SPD Server, or if you install a newer release of SPD Server, the installation process does not alter any files that have been modified in the **"InstallDir"/site** directory. Any custom modifications that were made to installation files are retained when you upgrade SPD Server to a subsequent release.

Chapter 4

Installing SAS Scalable Performance Data (SPD) Server on Windows

Before You Install: Precautions and Required Permissions

Review the following precautions and required permissions:

- If you want SPD Server clients to connect to the SPD Server host using name services instead of specifying port numbers at invocation, you need Write access to

your server machine's **...\etc\services** directory. Name services require you to define registered ports that use the services file appropriate to your machine.

- If your SPD Server clients access the SPD Server host by using name services instead of by specifying port numbers, you need Write access to the services files on the clients, in the path **...\etc\services**.

SPD Server Windows Installation Packing List

Directory names in the packing list are subdirectories of your SPD Server host installation directory, whose path is represented by **InstallDir**.

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

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

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

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

The **InstallDir\site** directory is a storage directory for a user's site-specific customization of the sample SPD Server start-up and configuration files. No SPD Server files are delivered in this directory; it is for customer use only.

SPD Server Quick Windows Installation Guide

This section outlines a quick installation of SPD Server to validate and test your installation.

Follow the instructions on your SAS software order. 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 SAS Scalable Performance Data Server product for installation. You also need to include the SAS Scalable Performance Data Client, the SAS Management Console, and the SAS Scalable Performance Data Server Plug-in for SAS Management Console in your product installation selection.

Configuring SPD Server Host Software for Your Site

During installation of SAS Scalable Performance Data Server, you will be prompted to configure the SPD Server connection port, as well as the SPD Server SNET server connection port. The SPD Server Data server connection port connects the SPD Server LIBNAME client engine to the SPD Server Data server. The SPD Server SNET connection connects a JDBC client to the SPD Server SNET server. You use the SPD Server data port to make connections from the SAS SPD Server LIBNAME engine to the SPD Server host, as well as to the SPD Server SNET server port. The SPD Server 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. You can change the default port assignments to port numbers that are available on your system. If you are running within a firewall, see "How do I know which ports must be surfaced through an Internet firewall?" in Appendix 2 of *SAS Scalable Performance Data Server: User's Guide* or see "Using SAS Scalable Performance Data (SPD) Server with an Internet Firewall" on page 175 for more detailed information.

Completing the Quick Start Configuration

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 installation of your server:

1. Verify that the WORKPATH statement in your spdsserv.parm file is valid. If the WORKPATH statement is incorrect, edit your spdsserv.parm file.
2. Verify that the pathname for libname=tmp in the libnames.parm file is valid. If the pathname is not correct, edit the libnames.parm file.
3. Initialize the SPD Server Password Manager Database:
   a. Click the Windows **Start** button and select **All Programs** ⇨ **SAS** ⇨ **SAS SPD Server 5.1** ⇨ **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. 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 manager 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 5.1 Name Server**.
5. Start the SPD Server data server. Click the Windows **Start** button and select **All Programs** ⇨ **SAS** ⇨ **SAS SPD 5.1 Data Server**.
6. Use the Windows Task Manager to verify that SPD Server is running. You should see the following processes running:
   - spdserv
   - spdsbase
   - spdsnsrv

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, and then restart your server.

Testing the SPD Server Quickstart Installation

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;

/* Submit the LIBNAME statements */

/* Open the program verify.sas   */
/* from InstallDir\samples\ dir  */
/* and submit it to verify data  */
/* set access to SPD Server.     */

/* Open the program verptsql.sas */
/* from InstallDir\samples\ dir  */
/* and submit it to verify SQL   */
/* access to SPD Server.         */

:
```

Potential Problems with Test Verification

Your LIBNAME statement to SPD server 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 SAS 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 5.1 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 5.1, verify that you are running SAS 9.4 with the SAS Scalable Performance Data Server client.

Customizing Your SPD Server Installation

After you complete the SPD Server quick start, you can customize the installation for your site. Shut down SPD Server using the Windows Task Manager.

Logging

The **InstallDir\site*.bat** files for the spdsnsrv, spdsserv, and spdssnet assume that you want to keep the logs from messages that are written to STDOUT or STDERR by the SPD Server name server (spdsnsrv), the SPD Server host (spdsserv), or the SPD Server SNET server (spdssnet). These processes are logged to the **InstallDir\log** directory.

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 logfile, 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 specification 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, -logtime 00:00 cycles the log at midnight.

Audit File Facility

The **"Installdir"\site\spdsserv.bat** file enables you to use the SPD Server audit file facility. The audit file facility is not enabled by default. Use the following start-up parameters to the spdsserv process to configure the SPD Server audit file facility:

-auditfile fileSpec

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
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
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 "SAS Scalable Performance Data (SPD) Server Auditing" on page 179.

User Password and Parameter Files

The spdsserv.bat file assumes that you keep your spdsserv.parm parameter file and your SPD Server user password file in the **"InstallDir"\site** directory. If you do not keep the files in this location, you need to change the following start-up parameters:

-parmfile file-spec
specifies an explicit file path for the SPD Server host's 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 file-spec
specifies the name of the file that contains the logical LIBNAME domain definitions that the SPD Server host supports.

.

Accessing SPD Server through a Registered Port

If you want to access 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, *sp45name*) or you can directly include the SPD Server port number in your SAS statements.

Adding Users to the SPD Server psmgr Database

See the section "Managing SAS Scalable Performance Data (SPD) Server Passwords and Users" on page 157 for more information about adding users to the SPD Server psmgr database.

Authenticating SPD Server User Passwords

SPD Server user passwords can be authenticated by the SPD Server psmgr utility, or by an LDAP server (such as Microsoft Active Directory, Sun Java System Directory Server, or OpenLDAP).

LDAP authentication integrates with the SPD Server password facility and provides a centralized approach to user ID and password management. SPD Server clients that use LDAP authentication should have user accounts that are managed by the authenticating LDAP server. The user ID and password information must be stored on an LDAP server that the SPD Server can access. The user ID must be entered into the SPD Server password database through the psmgr utility or via the SAS Management Console utility. These user ID requirements exist in order to ensure that all SPD Server user information is recorded and properly propagated.

When a client uses LDAP authentication to connect to SPD Server, the LDAP server that is configured in the SPD Server's parameter file performs the authentication. After the client is verified, SPD Server uses the client's password database record for all other SPD Server operations.

For more information about SPD Server LDAP authentication, see"Overview of LDAP Authentication" on page 167.

Customizing SPD Server Configuration

To customize configuration of the **`"Installdir"\site\spdsserv.parm`** parameter file and the **`"Installdir"\site\libnames.parm`** LIBNAME file for your installation, see Part 4, Chapters 6–13, in *SAS Scalable Performance Data Server: Administrator's Guide*.

Install SPD Server as a Service

You can use Windows Services to start SPD Server. To install the services that start the SPD Server name server, the SPD Server Data Server, and the SPD Server SNET server, select **Start** ⇨ **All Programs** ⇨ **SAS** ⇨ **Utilities** ⇨ **Install SPD 5.1 as a Service**.

Select **Start** ⇨ **All Programs** ⇨ **SAS** ⇨ **Utilities** ⇨ **Start SPD 5.1 as a Service** to manually start the SPD Server service.

Select **Start** ⇨ **All Programs** ⇨ **SAS** ⇨ **Utilities** ⇨ **Stop SPD 5.1 as a Service** to manually stop the SPD Server service.

After you install SPD Server as a service, you can 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 5.1 services.

Most users want to configure SPD Server to automatically start and stop when they start and stop Windows. The **Automatic** setting starts the name server, data server, and SNET

server without prompting. The same setting also stops the services without prompting when you close Windows.

Here is how to configure SPD Server to automatically start and stop when users start and stop Windows:

1. In the Window Services window, scroll down the **Services** list to find the entry for SPD Server 5.1 Name Server.
2. Select the name server service in the list, right-click on it, and select **Properties**. The Properties window appears.
3. Select **Automatic** from the **Start-up type** list. This setting configures the SPD Server Name server service to automatically start and stop with the Windows operating environment. 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 automatically start and stop with the Windows operating environment.

   Note: The first time you set your SPD Server services to Automatic, you need to manually start them by selecting **Start** ⇨ **All Programs** ⇨ **SAS** ⇨ **Utilities** ⇨ **Start SPD Server 5.1 Services**. After you manually start the services, they will automatically start and stop with Windows.

Notes for SPD Server Administrators

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

SPD Server User IDs

The SPD Server administrator needs to be familiar with the SPD Server psmgr utility.

The SPD Server system uses its own layer of access controls that overlay Windows access permissions. SPD 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 SPD User is given their own SPD Server user ID and password. The user ID and password are needed to complete the LIBNAME connection to SPD Server. All resources that a user creates are owned by the user. An SPD Server user can access only resources that that user created, or resources that another SPD Server user grants them access to via SPD 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 SPD Server user.

Troubleshooting

Key information for SPD Server troubleshooting can be found in the SPD Server name server log and in the SPD Server host process log files. Those two log files enable you to reconstruct SAS interactions with SPD 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 -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 -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:

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 5.1 while engine version is <version>.
: This error indicates you are trying to connect to SPD Server with an old client. Verify that you are running SAS 9.4 with 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.

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. You must restart SPD Server to use the new license. You must edit your rc.spds LICFILE= variable name so that it reads the name of the new license file.

Upgrading and Reinstalling SPD Server

If you need to reinstall SPD Server, or if you install a newer release of SPD Server, the installation process does not alter any files that have been modified in the **"InstallDir"\site** directory. Any custom modifications that were made to installation files are retained when you upgrade SPD Server to a subsequent release.

Part 3

Migration

Chapter 5

Using Earlier Version Tables with SAS Scalable Performance Data (SPD) Server 5.1

Introduction

SPD Server 5.1 uses architectures that enable features like the ability to support data tables that contain more than 2 G of observations. The new architectures provide functionality that today's data marts need, but the current generation of SPD Server tables use metadata architectures that are not backward compatible with SPD Server 3.*x* software.

SPD Server 4.*x* tables use architectures that use SAS 9.4 metadata. SPD Server 3.*x* tables use architectures that use SAS 8 metadata. These two metadata architectures are not compatible. However, SPD Server 5.1 provides a conversion utility SPDSCONV that SPD Server 3.*x* customers can use to convert existing tables into SPD Server 4.*x* format, which SPD Server 5.1 can read and open. If SPD Server 5.1 modifies an SPD Server 4.*x* table, the table will be saved in SPD Server 5.1 format. The SPDSCONV utility is designed to be run by the SPD Server Administrator.

To summarize:

- SPD Server 5.1 can read SPD Server 5.1 and SPD Server 4.*x* tables.
- SPD Server 4.*x* tables that are modified by SPD Server 5.1 become SPD Server 5.1 tables.
- SPD Server 3.*x* and SPD Server 4.*x* cannot read SPD Server 5.1 tables. If you have SPD Server 3.*x* tables, you must convert them to SPD Server 4.*x* tables before SPD Server 5.1 can read or open the tables.

Note: Before you begin the SPDSCONV conversion process, back up all images of the SPD Server 3.*x* data sets that are to be converted.

Converting SPD Server 3.x Tables to SPD Server 4.x Tables

Before You Convert

The SPDSCONV utility changes the format of SPD Server 3.*x* tables to the format used by SPD Server 4.*x* tables. The conversion is not reversible. After you convert a table for use with SPD Server 4.*x*, the table can no longer be read by SPD Server 3.*x*. Before you start converting SPD Server tables, back up all your existing SPD Server 3.*x* tables. If you ever need to revert to your SPD Server 3.*x* tables, you can restore them.

If you need to temporarily use your tables with both SPD Server 3.*x* and SPD Server 4.*x*, you can use PROC COPY to make copies of the tables before you convert them. You can use SPDSCONV on the copied versions to convert them for use with SPD Server 4.*x* (and, by extension, SPD Server 5.1) and maintain archival versions of the older SPD Server tables.

Overview of the SPDSCONV Utility

The SPDSCONV utility converts SPD Server 3.*x* metadata files for use with SPD Server 4.*x*. The conversion process updates the physical structure of the metadata files and renames them. The SPDSCONV utility also updates the data partition files if the SPD Server 3.*x* tables that are being converted contain compressed data.

You can identify SPD Server 3.*x* table files by their filename extension. SPD Server 3.*x* table files end with the filename extension .spds8. SPD Server 4.*x* and SPD Server 5.1 table files end with the filename extension .spds9. All tables that are upgraded to be compatible with SPD Server 4.*x* or SPD Server 5.1 have the filename extension .spds9.

SPD Server 4.*x* and SPD Server 5.1 index files differ from SPD Server 3.x index files. SPD Server 4.*x* and SPD Server 5.1 index files permit greater numbers of observations than SPD Server 3.*x* index files did. SPD Server 3.*x* index files are not compatible with the SPD Server 4.*x* or SPD Server 5.1 environment.

The SPDSCONV table conversion utility does not re-create index files. When you use the SPDSCONV utility to convert tables from SPD Server 3.*x* to SPD Server 4.*x* and SPD Server 5.1 format, the utility automatically deletes physical files that were associated with the old 3.*x* indexes and that are now obsolete. The SPDSCONV utility does offer an option to create a SAS job file that you can run in the SPD Server 4.*x* or SPD Server 5.1 environment to re-create the SPD Server 3.*x* index files for use with SPD Server 5.1.

If you choose to create the SAS job file to re-create SPD Server 3.*x* indexes for use in SPD Server 4.*x* and SPD Server 5.1, the code resembles the following example:

```
%let SPDSIASY=YES;
 PROC DATASETS
  lib=<spdsv4 LIBNAME>;
 modify MYTABLE;
   index create X1 [/Options];
   index create X2 [/Options];
 ...
```

```
quit;
```

You can specify the destination directory for the SAS job file that you create, but the SPDSCONV utility names the job file that you create. The utility generates SAS job filenames by adding the text string **`_v4ix.sas`** to the table name.

TIP You should defer re-creating an index because the process can require intensive computing. In busy computing environments, consider performing this task as an off-peak batch job.

After you convert the indexes, you might notice that SPD Server 4.*x* metadata files are slightly larger than the SPD Server 3.*x* metadata files were. The increase in file size is related to the new structures that enable SPD Server 4.*x* and SPD Server 5.1 to use tables that contain more than 2,000,000 rows of data.

How does SPDSCONV work? When the SPDSCONV utility converts a table, it reads the original SPD Server 3.*x* metadata file and creates a new SPD Server 4.*x* and SPD Server 5.1 metadata file. Both these files are locked during the conversion process. The lock prevents other users from accessing the files while changes are being made. If the conversion process encounters problems, the SPD Server 4.*x* and SPD Server 5.1 metadata file is deleted, and the original SPD Server 3.*x* table remains intact.

SPDSCONV reads the SPD Server 3.*x* metadata file one section at a time, and re-creates each structure in the SPD Server 4.*x* and SPD Server 5.1 metadata file as it is read. After the SPD Server 4.*x* and SPD Server 5.1 metadata file is fully populated, the utility checks the data partition file component to determine whether updates are required.

If SPDSCONV detects the presence of compression block headers, then the data partition file contains SAS 8 compression information that is not compatible in SPD Server 4.*x* and SPD Server 5.1, and SPDSCONV must update the data partition files. SPDSCONV updates the file by overwriting the compression block headers. SPDSCONV does not change the size of the data partition file, of any file components, or of any data that is contained in the files. The increase in metadata file size is modest and represents only a small percentage of storage space when compared to its corresponding data partition file component.

After SPDSCONV updates the data partition file, you cannot restore or re-create the original SPD Server 3.*x* data partition file. Ensure that you have complete backup images of the SPD Server 3.*x* data sets that you intend to convert before you run the conversion.

After the SAS job file re-creates the SPD Server 3.*x* indexes for use with SPD Server 4.*x* and SPD Server 5.1, all remnants of the SPD Server 3.*x* table are deleted. The SPDSCONV utility does not perform ACL checks during the conversion process. You cannot browse the contents of table rows from within the utility. During the metadata file conversion, no table rows are accessed, and there are no options to expose table row contents as part of logging or index job creation. The SPD Server 4.*x* and SPD Server 5.1 table retains the same SPD Server owner as the SPD Server 3.*x* table.

Using SPDSCONV

The SPDSCONV program is a command-line utility. You use a set of command-line options and parameters to specify the name and location of tables that you want to convert. Then you specify the options that you want for your conversion. If your SPD Server software is installed on a UNIX platform, see Chapter 3, “Installing SAS Scalable Performance Data (SPD) Server on UNIX,” on page 9 for information about setting up your environment before you run SPDSCONV.

The command-line syntax is as follows:

```
SPDSCONV <Options> [-a | table1 [table2]]
```

The order of options and table names on the command line does not matter. All of the currently available options are global options. Placing a global option before or after a table does not change the option setting for that table alone. Global options are always applied globally.

The SPDSCONV command has the following options:

-d pathname
: the directory path that corresponds to an existing SPD Server LIBNAME domain. This pathname should be the same as the PATHNAME= directory path in the libnames.parm file.

-l logpath
: the directory path where SAS job files that are created during the conversion process should be stored. The default logpath setting is the directory from which the SPDSCONV command is issued.

-a
: converts all SPD Server 3.*x* compatible tables in the **`-d pathname`** directory to tables that are compatible with SPD Server 4.*x* and SPD Server 5.1.

-j
: creates a SAS job in the log directory for each SPD Server 3.*x* table that contains an index. When you use the -j option, the SAS job re-creates the indexes on the SPD Server 5.1 table. You must run the SAS job after the SPDSCONV utility completes the conversion process. Because re-creating an index recreation can be computation-intensive, you should schedule SAS index re-creation jobs as a SAS batch job during off-peak hours. The utility generates the name of the SAS job file, which has the following format:

  ```
  TableName_v4ix.sas
  ```

  In the name, TableName is the name of the table that is compatible with SPD Server4.*x* and SPD Server 5.1. The SAS job file contains the SAS language statements that are required for re-creating the indexes that the SPD Server 3.*x* table used. You need to edit the job file before the job executes. Editing the file ensures that the correct SPD Server 4.*x* and SPD Server 5.1 LIBNAME is used with the PROC DATASETS statement.

-v
: creates verbose output for the conversion process.

SPDSCONV Utility Examples

Example LIBNAME Parameter File

These examples assume that you use the following LIBNAME parameter file for your SPD Server installation:

```
LIBNAME=usmkt pathname=/mdat1/usmkt
 roptions="datapath=('/dat11/usmkt'
                     '/dat12/usmkt'
                     '/dat13/usmkt'/
                     '/dat14/usmkt')
 indexpath=('/ix11/usmkt'
            '/ix12/usmkt')";

LIBNAME=sales pathname=/mdat1/sales
```

```
 roptions="datapath=('/dat21/sales'
                     '/dat22/sales'
                     '/dat23/sales'/
                     '/dat24/sales')

 indexpath=('/ix21/sales'
            '/ix22/sales')";
```

Converting a Simple Table

Suppose you have an SPD Server 3.*x* table named CT010299 that belongs to the **usmkt** domain, and you want to convert CT010299 to use with SPD Server 4.*x* and SPD Server 5.1. Table CT010299 does not have indexes and you want a verbose output of the table conversion.

Issue the following command:

```
SPDSCONV -v -d /mdat1/usmkt CT010299
```

Converting Tables and Re-creating Indexes

Suppose you want to convert all tables in the **sales** domain. You also want SPDSCONV to create SAS jobs that you can run to re-create the indexes after the table conversion is complete. You want the SAS jobs put into the directory **$HOME/salesv9**. You also want a verbose output of the conversion.

Issue the following command:

```
SPDSCONV -v -d /mdat1/sales -l $HOME/salesv9 -j -a
```

Converting SPD Server 4.*x* 32-Bit Windows Tables to SPD Server 5.1 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 5.1 tables in 64-bit mode. SPD Server 5.1 for 64-bit Windows does not recognize SPD Server 4.x components. If you submit a PROC CONTENTS command to SPD Server 5.1 on a Windows 64-bit server, SPD Server 5.1 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 5.1 tables for use in a 64-bit Windows environment:

Convert SPD Server 4.x Tables to SAS Tables, Then to SPD Server 5.1 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 5.1 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 a SAS 9.4 LIBNAME connection to your SPD Server 5.1 domain. Now you are able to use the SAS 9.4 PROC COPY command to restore the former SPD Server 4.*x* tables to your SPD Server 5.1 domain in SPD Server 5.1 format.

```
LIBNAME spd51 sasspds "<SPD-Server-5.1-domain-name>"
  host=localhost
  serv=<SPD-Server-5.1-server-port>
  user=<user-ID> ;

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

PROC COPY in=SAS94 out=spd51;
```

SPD Server 5.1 will be able to recognize and use the copied tables in the new SAS 9.4 and SPD Server 5.1 domain space.

Convert Directly from SPD Server 4.x Tables to SPD Server 5.1 Tables

You can convert directly from SPD Server 4.*x* tables to SPD Server 5.1 tables by using SAS 9.4 with an SPD Server 4.*x* and an SPD Server 5.1 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.*x* server, and then you make the second LIBNAME connection to your SPD Server 5.1 server. After you establish a LIBNAME connection between the SPD Server 4.*x* host and the SPD Server 5.1 host, you can use a PROC COPY statement to copy your SPD Server tables directly from your 4.*x* host to your SPD Server 5.1 host.

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

LIBNAME spd51 sasspds "<SPD-Server-5.1-domain-name>"
  host=localhost
  serv=<SPD-Server-5.1-server-port>
  user=<user-ID> ;

PROC COPY in=spd44 out=spd51;
```

Using PROC COPY with SPD Server Clusters

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 tables that use SPD Server clusters, 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 5.1, you must first undo the table cluster in the source (SPD Server 4.*x*) location before copying the component tables to the destination (SPD Server 5.1) location.

After the tables are unclustered, you can use PROC COPY to copy the files from the source domain to the destination domain. After you successfully copy your tables to the destination domain, use SPD Server 5.1 CLUSTER TABLE commands to reform the clustered table structures.

Part 4

Configuration

Chapter 6

Using the SAS Scalable Performance Data (SPD) Server Name Server to Manage Resources

Managing SPD Server Computing Resources with a Name Server

The SPD Server name server gives you the ability to manage computing resources without disturbing system users. The SPD Server name server maps logical names that are referenced in SAS programs to the physical location of SPD Server tables. A name server enables you to add, remove, or reallocate disk space and computing power without needing to change SAS source code. You also do not need to inform others about changes in resource allocation as long the name of the machine that hosts the name service remains the same.

The following example demonstrates how to configure SPD Server to spread the processing load across multiple machines in a network.

Configuring SPD Server on a Corporate Network

Example Scenario

The corporation in this example maintains a network of computers. The corporate network contains a mix of computers and processing power: machines with multiple processors and large amounts of available disk space, smaller machines that are used for servers, and desktop machines for client users. The dedicated servers have the following names:

worldcpu
: a data store for the company's worldwide operations

asiacpu
: a data store for the company's Asia department, which uses the data to generate reports, analysis, and so on

namecpu
: the machine that runs the name server

Because data for worldwide operations is stored in an SPD Server table on **worldcpu**, the Asia department must periodically access that server. The Asia users want to extract **worldcpu** data to create SPD Server tables that will reside on their own departmental server, **asiacpu**. The Asia users can then access tables that contain only Asia data, and transfer that information to their desktops for further analysis.

The SPD Server system administrator runs the name server on the **namecpu** machine. Consequently, **namecpu** must be accessible by every machine in the network that wants to locate an SPD Server table. The administrator must also run a data server on the **worldcpu** and **asiacpu** machines. The following section describes how to configure the servers in order to distribute the processing load.

Run the Name Server on the namecpu Machine

Invoke the name server by using the -listenport option. Specify a valid TCP/IP port number. Use the same port number when you invoke SPD Server on the **worldcpu** and **asiacpu** servers.

Configure SPD Server on the worldcpu Server

The libname.parm file that resides on the **worldcpu** server contains the following line:

```
LIBNAME=world pathname=/spds;
```

This code instructs SPD Server to register the combination **`(world, worldcpu, /spds)`** with the name server. Thereafter, when a SAS LIBNAME statement contains the domain name **`world`** in combination with the appropriate name server, it will locate SPD Server tables in the directory **`/spds`** on the **worldcpu** server. The following SAS LIBNAME statement invokes the SPD Server engine and makes this association:

```
LIBNAME worldlib sasspds 'world'
  server=namecpu.spdsname;
```

When your network uses an SPD Server name server, the users do not have to remember which machine houses a particular domain. Users need to remember only that the SAS domain named **`world`** contains the tables that they need. Even if the machine that stores the domain changes without the users' knowledge, the users' SAS programs continue to run as before.

Specify **`namecpu`** as the value for the -nameserver option and invoke SPD Server. The value for the name server port must match the port number that you used to start the name server on that machine.

Set Up asiacpu, the Asia Departmental Server

The libname.parm file that resides on **asiacpu** contains the following line of code:

```
LIBNAME=asia pathname=/spds;
```

This code instructs the SPD Server that is running on **asiacpu** to register the combination **`(asia, asiacpu, /spds)`** with the name server. When a SAS LIBNAME statement contains the domain name **`asia`** in combination with the

appropriate name server, it will locate SPD Server tables in the directory **/spds** on machine **asiacpu**. The following SAS LIBNAME statement invokes the SPD Server engine and makes this association:

```
LIBNAME asialib sasspds 'asia'
  server=namecpu.spdsname;
```

The value that follows the LIBNAME server specification is the same in all these LIBNAME statements because both SPD Servers use a common name service. Asia departmental users do not need to know the name of the machine that provides storage for their domain.

Which SAS Program Statement Runs Where?

Assume that a user in the Asia department needs to create an SPD Server table on the departmental server **asiacpu**. This task requires data to be extracted from an SPD Server table named **alldata**. The user knows that the **alldata** table resides in the domain **world**. The user submits the following SAS code on a desktop SPD Server client:

```
LIBNAME worldlib sasspds 'world'
 server=namecpu.spdsname;
LIBNAME asialib sasspds 'asia'
 server=namecpu.spdsname;

data asialib.mydata;
set worldlib.alldata;
where region='Asia';
if country='Japan' then
    subreg=1;
run;
```

This code extracts records from an SPD Server table named **alldata** that resides in the domain **world**. The **world** domain is stored on machine **worldcpu** in the directory **/spds**. Because the **alldata** table resides on **worldcpu**, and SPD Server processes certain SAS WHERE clauses, the search for the value **Asia** is performed on **worldcpu**.

The SAS program runs on the Asia user's desktop machine. The desktop machine scans each row in the **alldata** table, looking for the string **Japan**. If the string is found, the desktop client forwards the row to the machine on which the output table resides, which is **asiacpu** in this example.

Disk space for the output table **mydata** is allocated in the **/spds** directory on **asiacpu**. The processing work (transferring data received from the user's desktop machine to the SPD Server table) is also performed by **asiacpu**.

The processing that was required to create the output SPD Server table was distributed across three machines. However, the user's desktop machine requires no permanent disk space, because SAS WHERE clauses execute on the machine that stores the source table. Only the selected rows that match the submitted WHERE clause are sent over the network to the desktop client. This strategy significantly reduces both network traffic and the time that is needed to complete a SAS program.

Chapter 7

Administering and Configuring SAS Scalable Performance Data (SPD) Server Using the SAS Management Console

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:

- 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 or 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.

SAS Management Console for SPD Server 5.1 is installed with SAS 9.4 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 **SPDS 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 46.

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
SPD Name Server listen port

Domain
the SPD Server domain

User
user name that is designated as ACL special

Password
password associated with the user name

Group
optional group name

ACL Special
select this box to enable ACL special privileges

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.

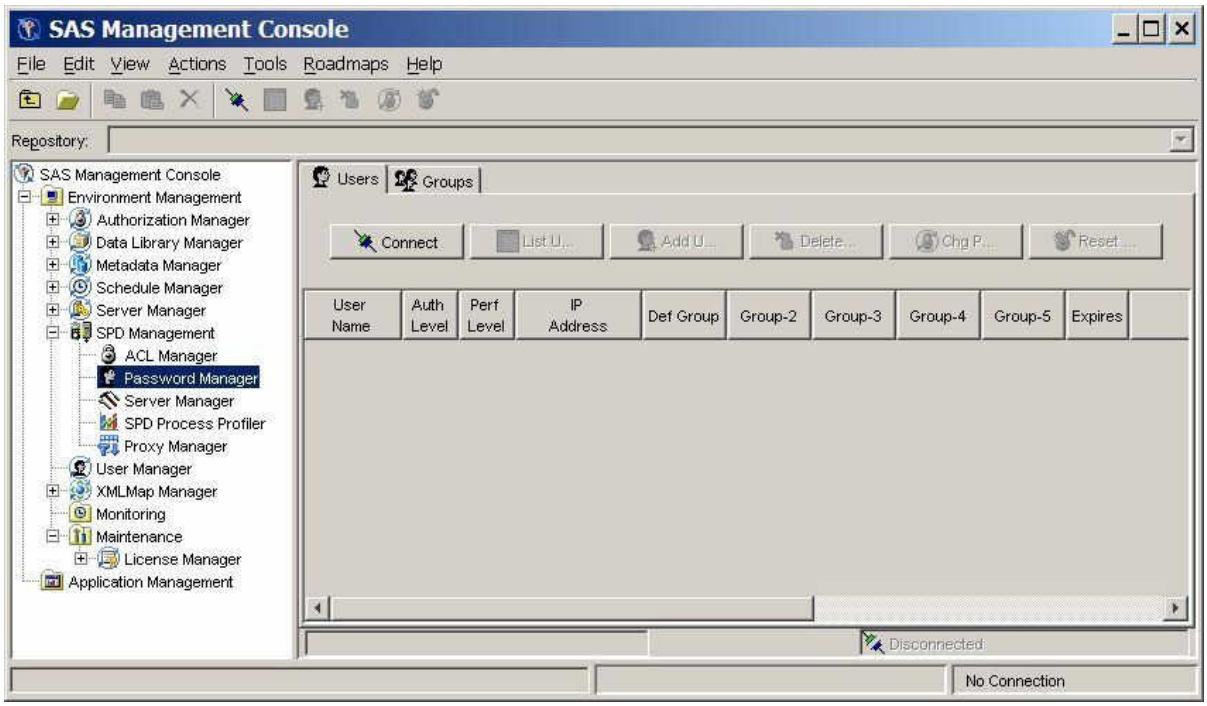
Note: Do not confuse the SAS Management Console status indicator in the lower right corner of the SAS Management Console window with the SPD Server connection status at the bottom of the active tab.

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:

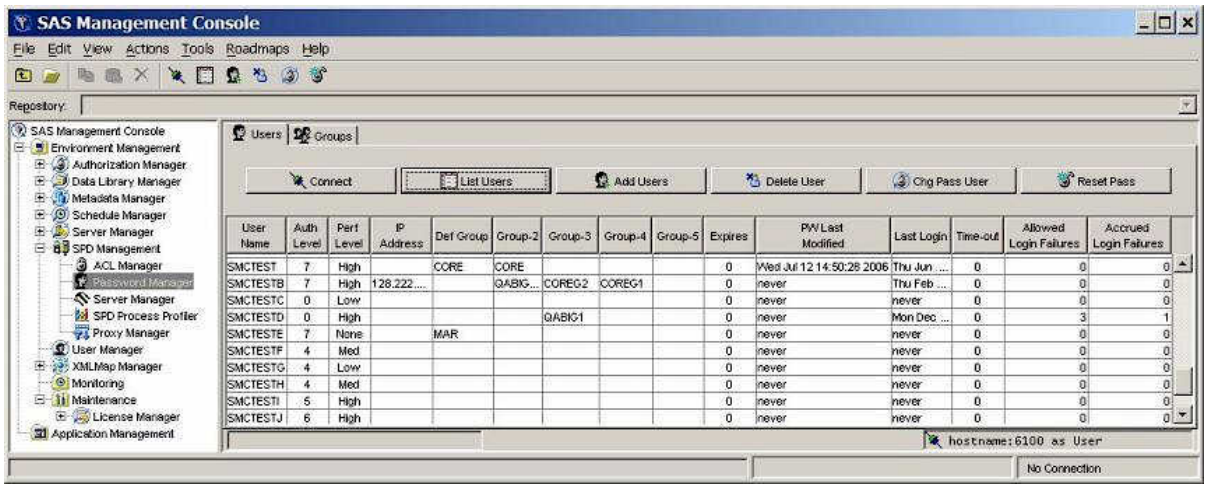


When no server connection exists, no data is displayed. To display data, you must connect to an SPD Server. For more information, see "Connect to an SPD Server" on page 46.

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.



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 50 and "Add a User" on page 50.

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 the individuals listed in the User Name column are using.

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.

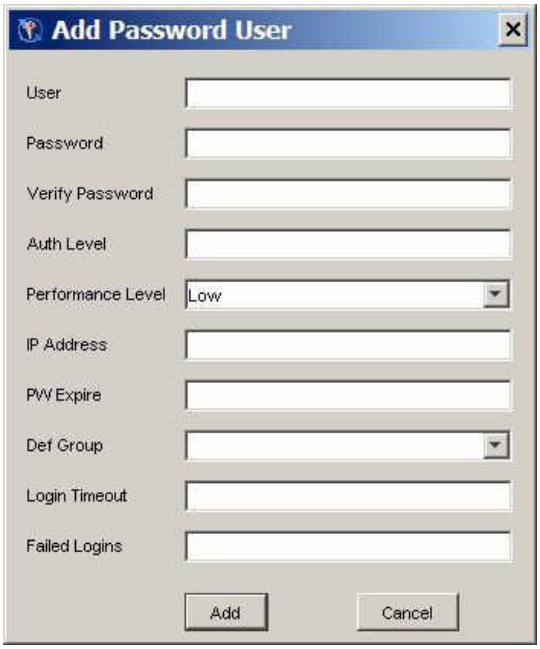
Time-out
number of idle days since the user's last login. When the number of days since a user's last login equals the **Time-out** value, the user's access is disabled. For example, if the **Time-out** value is 7, and if a given user does not log on at least every seven days, then the user's access 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 is a check that 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 82.



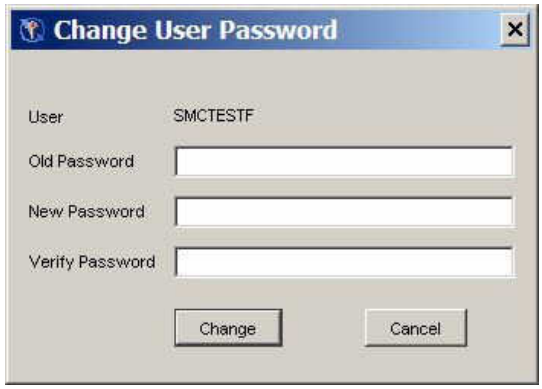
User, **Password**, **Auth Level**, **Def Group**, and **Failed Logins** are required values. The **IP Address**, **PW Expire**, and **Login Timeout** values are optional. If you omit the optional settings, they default to "no limits".

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

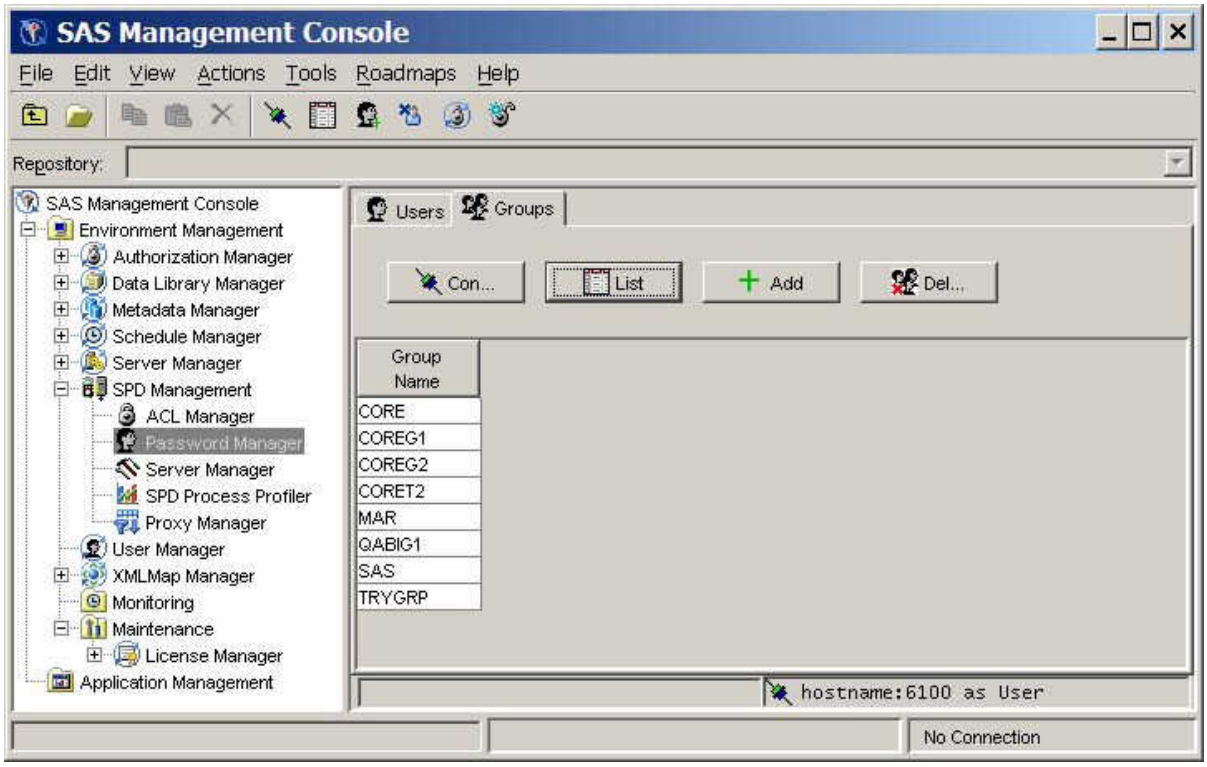
To reset the password for a selected user, click **Reset Pass**. Specify the user's new password and click **Change**. You do not need to know the user's current password to change it. The user must change the password before he can connect to the server.

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.



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.

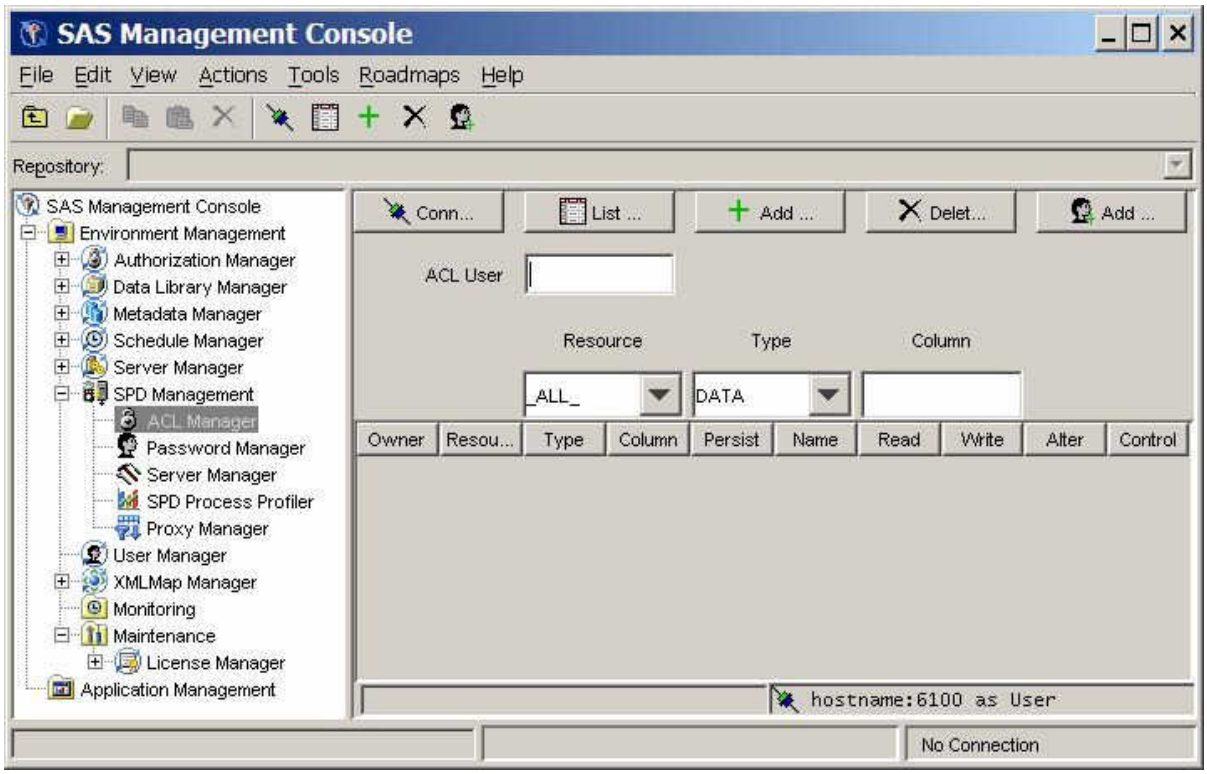
Delete a Group

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

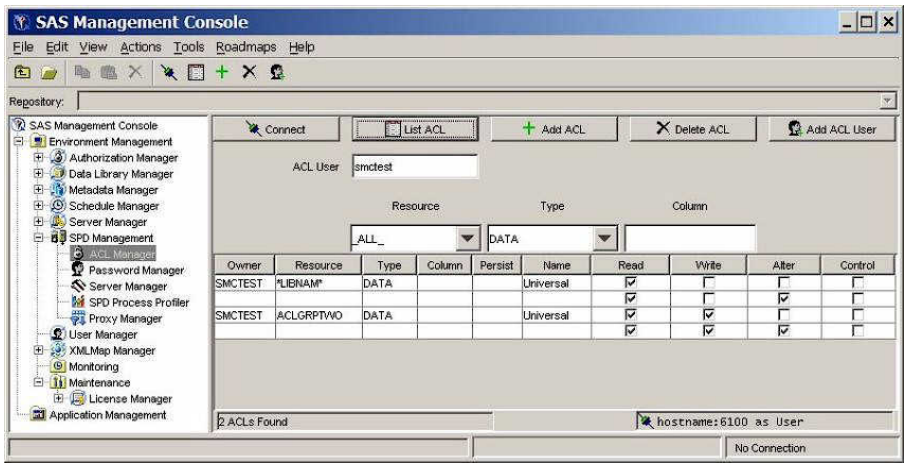
Click the ACL Manager folder in the SAS Management Console window to manage ACL resources. The ACL Manager panel resembles the following display.



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 46.

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.



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, VIEW, or MDDB). 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.

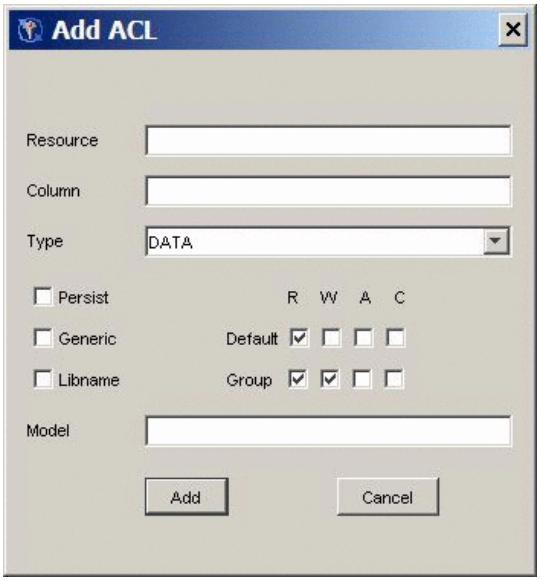
Write
if this check box is selected, the user or group has permission to write to this resource.

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.



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, VIEW, or MDDB).

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.

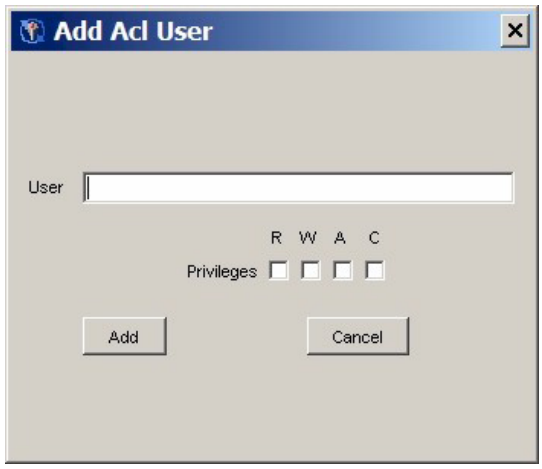
Model
specify the name of an existing ACL resource to use as a model for this ACL resource.

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

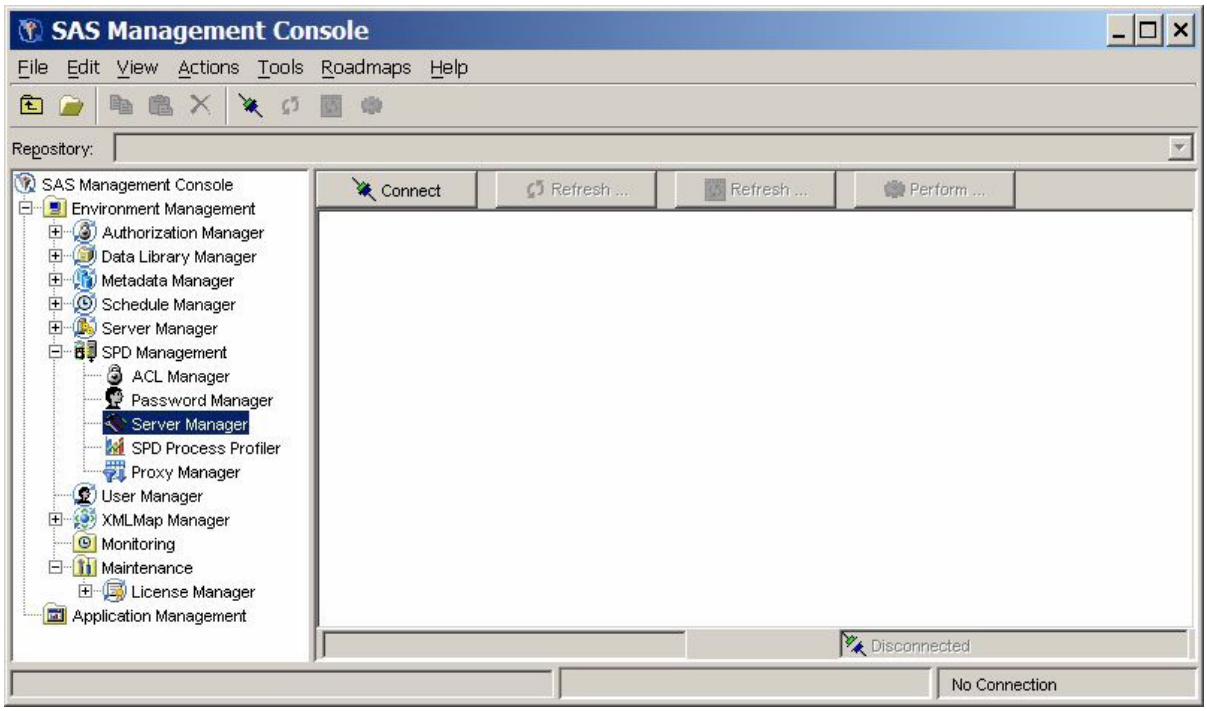
Each ACL resource has at least one set of permissions called universal permissions. Universal permissions are the default permissions for the ACL resource if no other permissions are applied. If any group names or user names have permissions for the ACL resource, they are displayed.

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

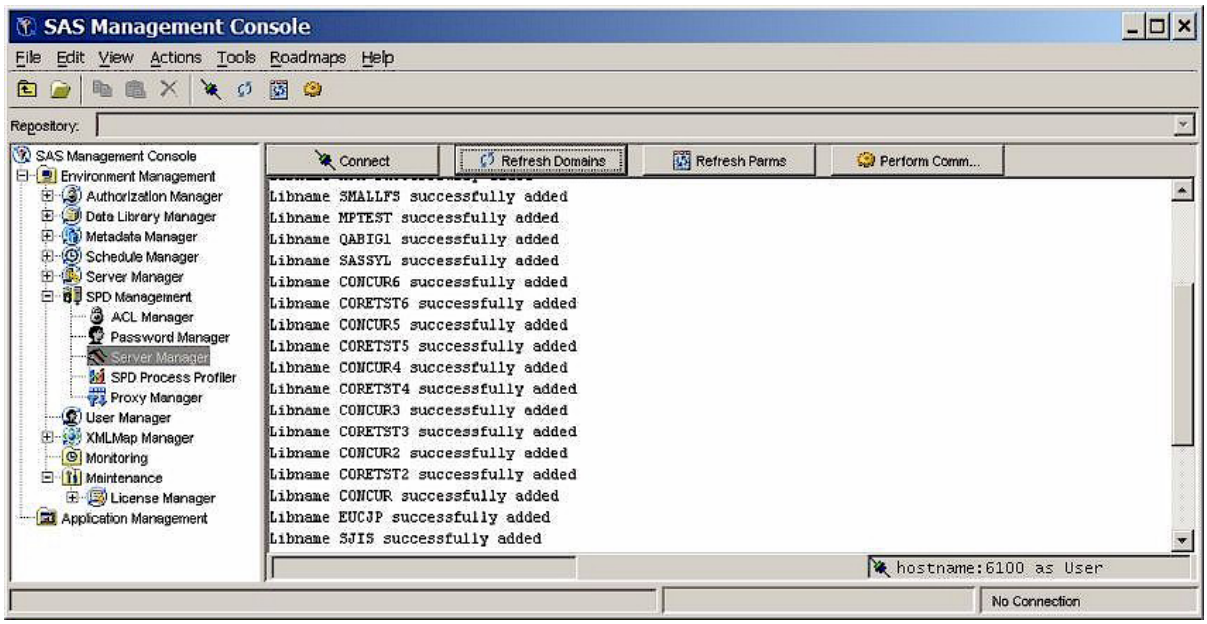
Click the **Server Manager** folder in the SAS Management Console window to refresh the SPD Server configuration and to run selected SPD Server utilities. The Server Manager panel resembles the following display when no server connection exists.



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 46.

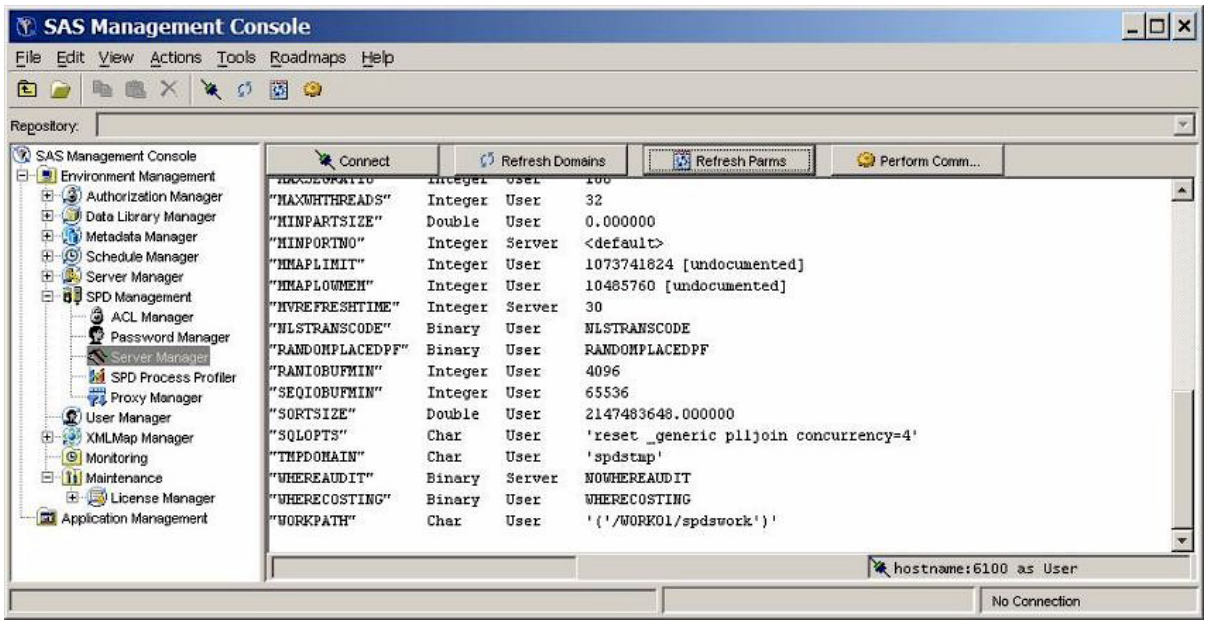
Refresh Domains

To reload the current libnames.parm file, click **Refresh Domains** in the Server Manage panel of the SAS Management Console window. The libnames.parm file describes the list of available domains.



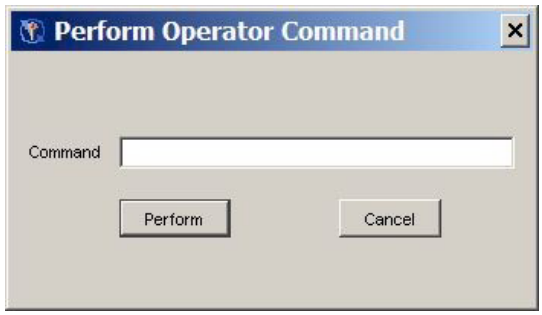
Refresh Configuration Information

To reload the current spdsserv.parm file, click **Refresh Parms** in the SPD Server Manager panel. The spdsserv.parm file controls the server configuration and options.

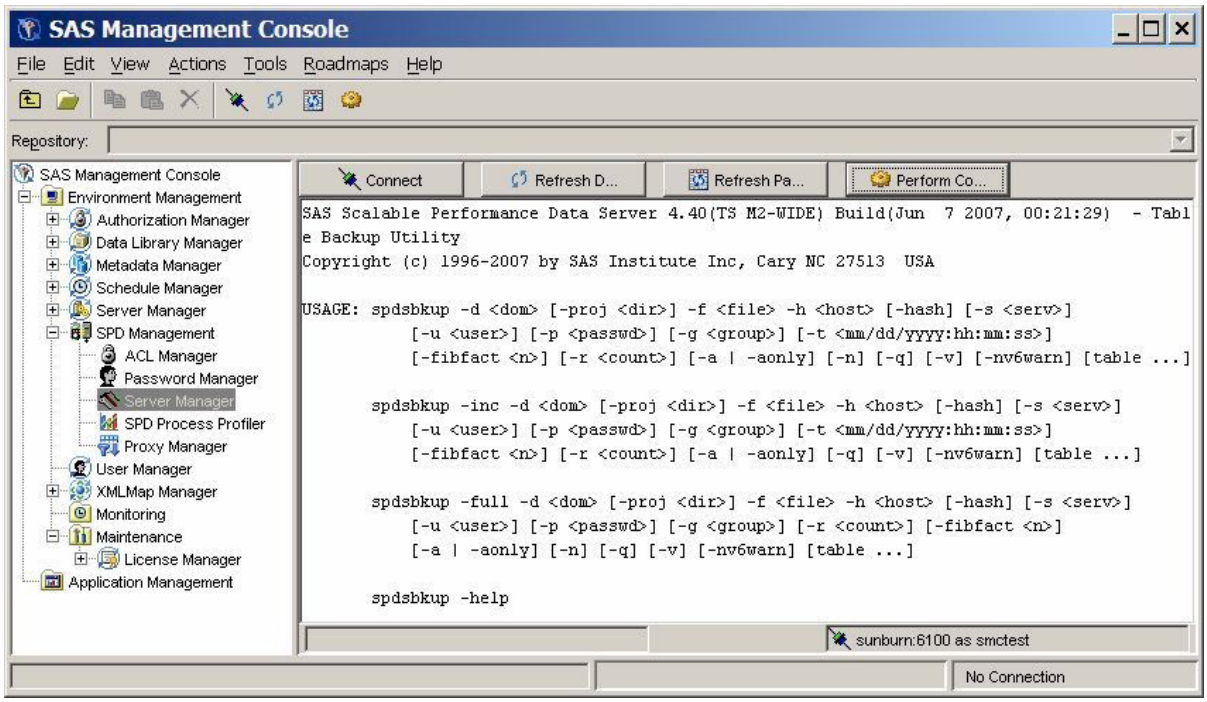


Run Commands

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**.



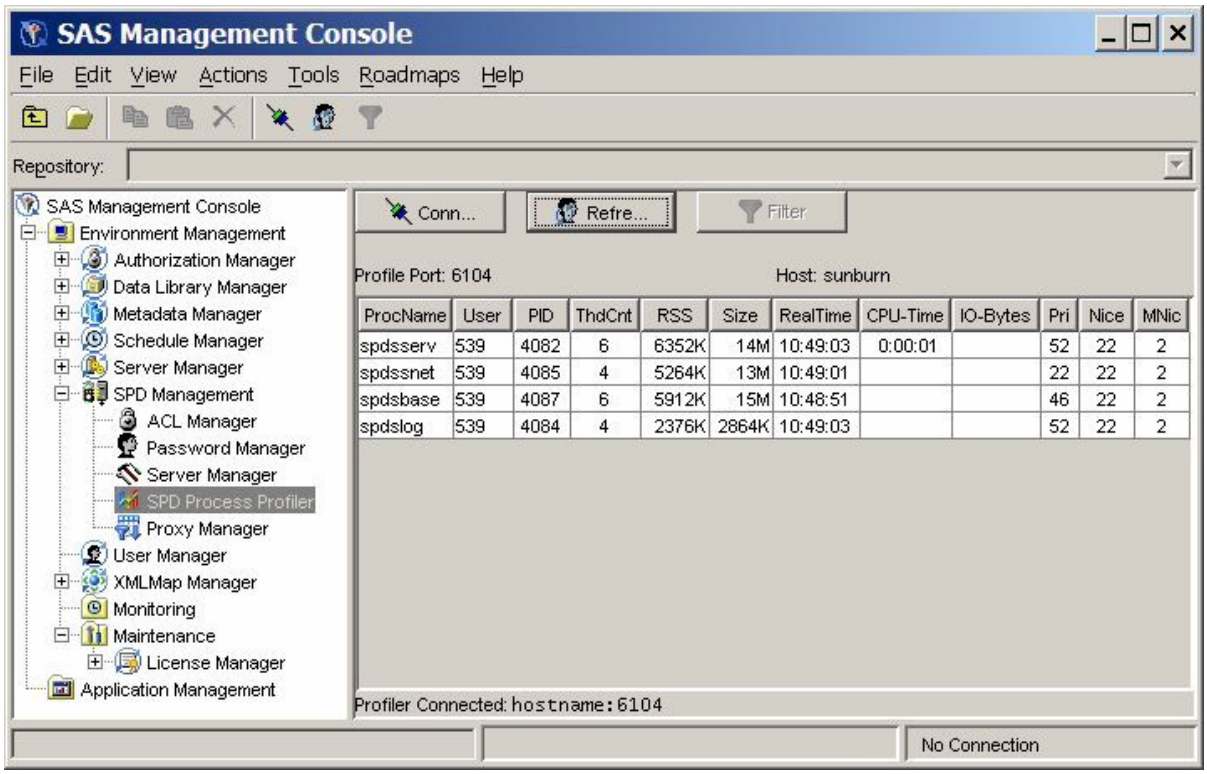
The following example shows the Server Manager panel after the **spdsbkup** command was issued:



Process Profiler

Click the **Process Profiler** folder in the SAS Management Console window to view server resources that are monitored by the Performance Server.

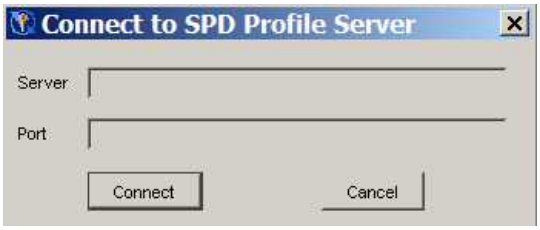
Note: The SPD Server Process Profile Manager is not supported on Windows or on Linux on X64.



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. More information about Performance Server start-up settings is available in Chapter 13, "Setting Up SAS Scalable Performance Data (SPD) Server Performance Server ," on page 113.

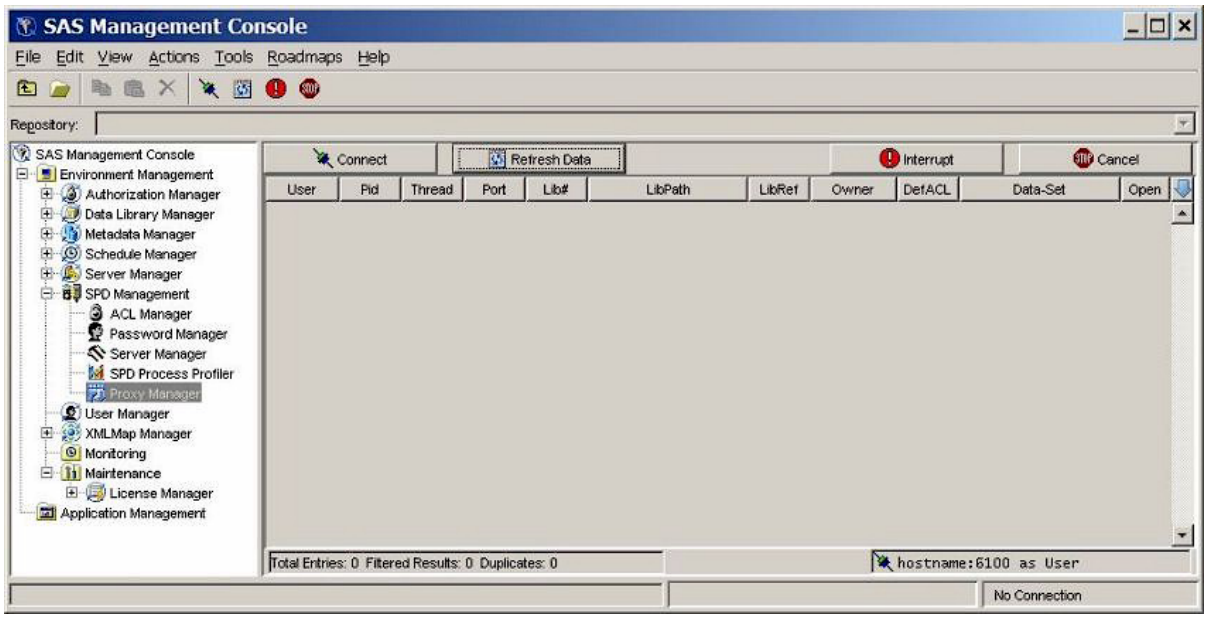
If the SAS Management Console user also connects to the same SPD Server name server through ACL Manager, Password Manager, or Server Manager, the user information that is displayed is the SPD Server user name that was used to connect with the LIBNAME statement. Otherwise, the user name of the user that 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 ACL special privileges, 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 46. After you connect to an SPD Server host, click **Refresh Data** in the Proxy Manager to update the table data.



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.

Chapter 8

SAS Scalable Performance Data (SPD) Server SQL Query Rewrite Facility

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 sub-queries 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 sub-queries 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. You must provide adequate space in order for the intermediate results tables to be generated and stored. The intermediate results tables are formatted as SPD Server tables. Optional indexes are permitted.

Intermediate results tables are stored in a common SPD Server LIBNAME domain that you specify. One dedicated SQL Rewrite Facility LIBNAME domain is sufficient to provide adequate intermediate results table storage for many concurrent SPD Server

users. Why is only one domain enough? The SQL Query Rewrite Facility uses the SPD Server TEMP=YES option setting when it accesses the LIBNAME 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 LIBNAME domains for SQL Query Rewrite Facility intermediate results storage. Administrators can use the TMPDOMAIN= parameter in the spdsserv.parm file to specify the SQL Query Rewrite Facility intermediate results storage domain:

```
TMPDOMAIN=<DomainName>;
```

The <DomainName> value is the name of a LIBNAME domain that is defined in the libnames.parm file that is associated with SPD Server.

SPD Server users can override the primary TMPDOMAIN= location by specifying their own LIBNAME domain for SQL Query Rewrite Facility intermediate results storage. Users specify their own LIBNAME domain by using the pass-through SQL RESET command with the TMPDOMAIN= option. For example, if an individual SPD Server user wants to use the EMATMP LIBNAME domain for SQL Rewrite Facility intermediate results, that user submits the following RESET command in his or her SQL job stream:

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

TMPDOMAIN=EMATMP causes the EMATMP domain to take precedence over the TMPDOMAIN= setting that was specified in the spdsserv.parm file. Any LIBNAME domain that an individual user submits as an SQL Query Rewrite storage location must be defined in the libnames.parm 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 TMP=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 sub-expressions that SPD Server can handle more efficiently by using the SQL Query Rewrite Facility, the software optimizes the SQL query. 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 9

Using SAS Scalable Performance Data (SPD) Server with Other Clients

Overview of Using SPD Server with Other Clients

SAS Scalable Performance Data (SPD) Server provides ODBC, JDBC,, and SQL C API access to SPD Server data stores from all supported platforms. SPD Server can read tables exported from Base SAS software using the COPY procedure. When the appropriate drivers are installed on the network, SPD Server allows queries on the tables from client machines that do not use SAS software. SPD Server provides the following four connectivity options.

- **ODBC** (Open Database Connectivity): ODBC is an interface standard that provides a common interface for accessing databases. Many software applications running in a Windows environment are compliant with ODBC and can access data created by other software. ODBC is a good choice if you have client machines running Windows applications, such as Microsoft Excel or Microsoft Access.
- **JDBC** (Java Database Connectivity): Users with browsers can log on to a web page and make a query. The results of the request are formatted and returned to a web page. After you install the JDBC driver on SPD Server, all you need is a web page with some Java code and a client machine with a browser enabled by Java. For this reason, information is available across a wide range of client platforms.
- **SQL C API**: SPD Server tables are accessible from SQL statements that are generated by C/C++ language applications. This access is provided in the form of a

C-language run-time access library. This library provides a set of functions that you can use to write custom applications to process SPD Server tables and to generate new ones. This library supports multi-threaded applications and is available on all supported SPD Server platforms.

Note: GUI interfaces might not display all return codes or error messages that the server generates.

Using Open Database Connectivity (ODBC) to Access SPD Server Tables

Requirements and Considerations for Using ODBC

To use ODBC, you must have SPD Server tables available on your network, SPD Servers and SPD SNET servers running, or client machines that are in a Windows environment. If you are working in such an environment, you might want to use ODBC if one or more of the following criterion are true:

- You do not have Base SAS software running on the Windows client, but you need to view or change SPD Server tables.
- You need to view or change the SPD Server tables using a Microsoft spreadsheet, database, or word processor.
- You need to view or change SPD Server tables in ways that cannot be predetermined or programmed into a web page.
- You need to view or change SPD Server tables using Windows tools that you are familiar with.

Install OBDC Drivers on the Client

Instructions for installing the SAS OBDC driver for SPD Server 5.1 are included in the download package that is available at **`http://support.sas.com`**.

Prepare Your Client Machine for ODBC Installation

Before you create an OBDC data sources driver, obtain the following information from your network administrator:

- a user name and password that is defined by an SPD Server administrator
- the primary LIBNAME domain of the SPD Server (also called the DBQ)
- the port number of the SPD name server (also called the SERV)
- the machine name or IP address of the SPD Server name server (also called the HOST)
- any secondary LIBNAME domains that you want to assign to the ODBC connection

Configure ODBC on the Client

There are two basic steps to configure ODBC on your client machine:

1. Configure an ODBC data source.
2. Make your query using a Windows program. For more information, see "Edit the Services File on Your Machine" on page 72.

You can connect directly to an SPD Server without going through the SPD SNET server. This configuration is the preferred one. However, you can configure connections through the SPD SNET Server. Both of these configurations are shown in the following figures.

Figure 9.1 *Configure ODBC to Connect SPD Server Client to SPD Server Host*

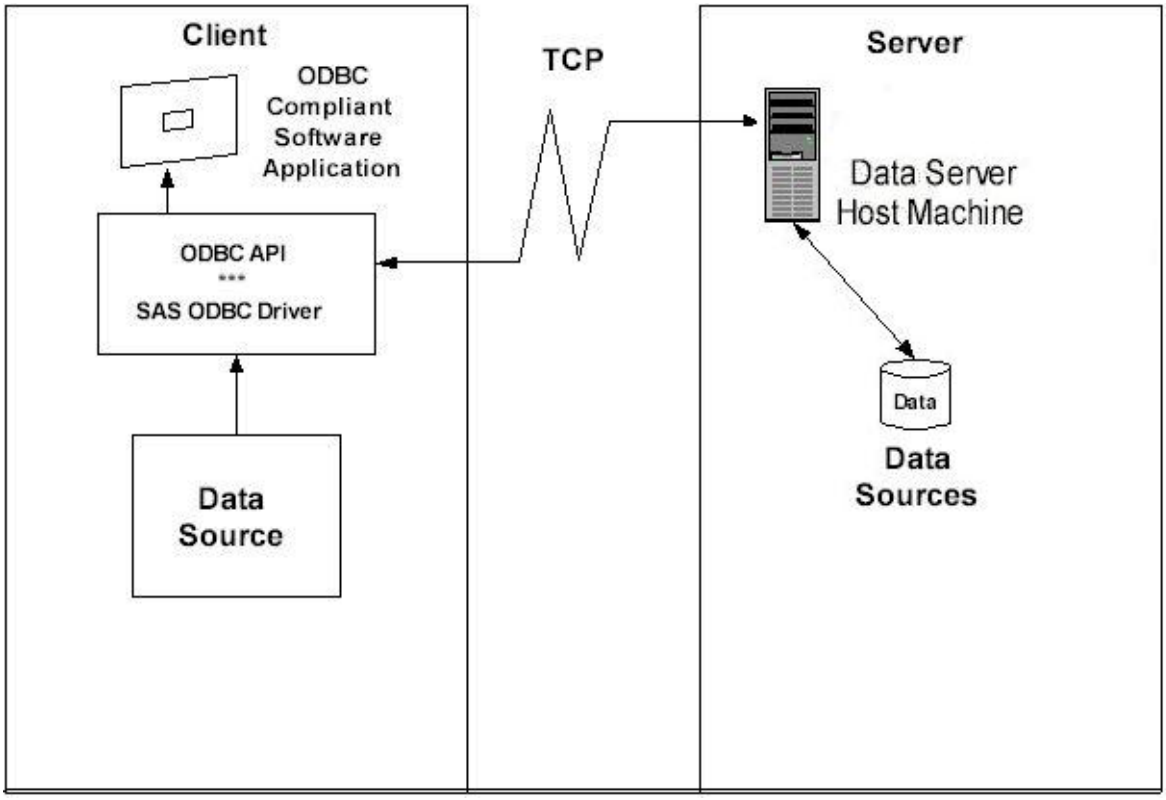
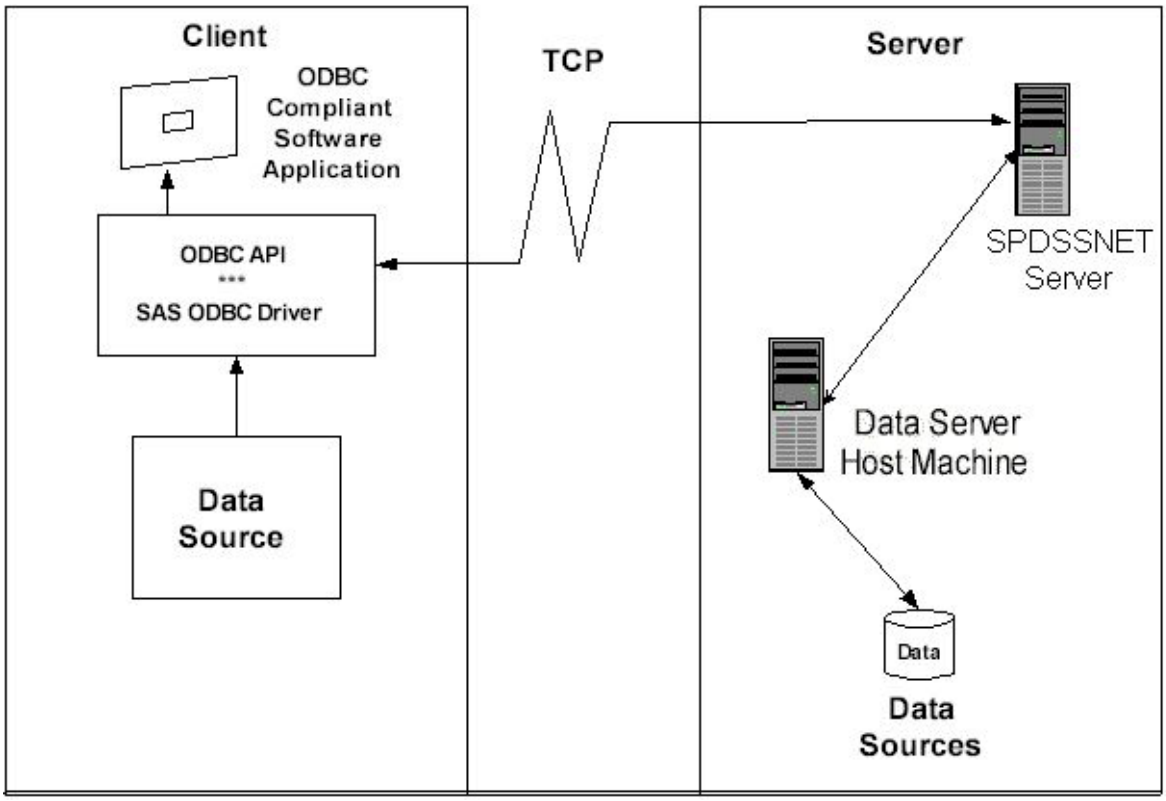


Figure 9.2 *Configure ODBC to Connect SPD Server Client to SPD SNET Server*



Connecting to SPD Server Using an ODBC Connection

Primary and Secondary LIBNAME Domains

When a connection to the SPD server is established, a primary LIBNAME domain is assigned. The primary LIBNAME domain is specified by the DBQ connection options parameter. Immediately after the connection is made, the SAS ODBC driver assigns the secondary LIBNAME domains. You configure these domains on the **Libraries** tab of the SAS ODBC Driver Configuration window.

To make an ODBC connection through the SPD SNET server, you must configure an odbc.parm file on the SPD SNET Server machine.

Configure an ODBC Data Source to Connect Directly to an SPD Server

After you install the SAS ODBC driver, configure your ODBC data source. When you open the ODBC manager, enter information that points the OBDC driver to the data on the SPD Server.

1. From the Windows **Start** button, select **Start** ⇨ **Settings** ⇨ **Control Panel**.
2. Locate the **ODBC Data Sources** icon and open the Microsoft ODBC Data Source Administrator. The exact location of this program depends on your version of Windows.
3. Select the **Add** button, and then select the SAS ODBC driver.
4. Enter a data source name. You can also enter a description if you want to.

5. Select the **Servers** panel and enter your two-part server name.
6. Click the **Configure** box. In the TCP Options window, enter the following information:

   Server Address
   Enter the network address of the machine on which the SPD Server is running.

   Server User Name
   Enter the user name as configured for a DBQ (an SPD Server primary LIBNAME domain) on the SPD Server to which you are connecting.

   Server User Password
   Enter the user password as configured for a DBQ (an SPD Server primary LIBNAME domain) on the SPD Server host to which you are connecting.

   Connection Options
   Enter values for the following options. Additional SPD Server LIBNAME options might be listed. For more information, see Chapter 10, "SAS Scalable Performance Data (SPD) Server LIBNAME Options ," in *SAS Scalable Performance Data Server: User's Guide*.

   DBQ='*SPD-Server-primary-LIBNAME-domain*'
   the SPD Server LIBNAME domain

   HOST='*name-server-node-name*'
   the location of the host computer

   SERV='*name-server-port-number*'
   the port number of the SPD Server name server that is running on the host
7. Click **OK**. Click **Add** and select the **Libraries** panel.
8. Enter the DBQ name of a secondary LIBNAME domain in both the **Name** and **Host File** text fields.
9. Enter **`spdseng`** in the **Engine** text field.
10. Enter appropriate information in the **Options** text field. In this field. use SQL pass-through syntax rules for libref statements.

Configure an ODBC Data Source for SPD SNET

After you install the SAS ODBC driver, configure your ODBC data source. When you open the ODBC manager, enter information that points the OBDC driver to the data on the SPD Server.

1. From the Windows **Start** button, select **Start** ⇨ **Settings** ⇨ **Control Panel**.
2. Click the **ODBC** icon and select the **Add** button.
3. Select the SAS ODBC driver.
4. Enter a data source name. You can also enter a description if you want to.
5. Select the **Servers** panel and enter your two-part server name. The second part of the server name should match the entry in the services file. For more information, see "Edit the Services File on Your Machine" on page 72. In that example, the server name is **spdssnet**.
6. Click the **Configure** box. In the TCP Options window, enter the following information:

   Server Address
   Enter the network address of the machine on which the SPD Server is running.

Server User Name
Enter the user name as configured for a DBQ (an SPD Server primary LIBNAME domain) on the SPD Server to which you are connecting.

Server User Password
Enter the user password as configured for a DBQ (an SPD Server primary LIBNAME domain) on the SPD Server host to which you are connecting.

Connection Options
Enter values for the following options. Additional SPD Server LIBNAME options might be listed. For more information, see Chapter 10, "SAS Scalable Performance Data (SPD) Server LIBNAME Options ," in *SAS Scalable Performance Data Server: User's Guide*.

DBQ='*SPD Server primary LIBNAME domain*'
the SPD Server LIBNAME domain

HOST='*name server node name*'
the location of the host computer

SERV='*name server port number*'
the port number of the SPD Server name server that is running on the host

7. Click **OK**, and then click **Add**.

Edit the Services File on Your Machine

Editing the Services file is required only for ODBC connections through the SPD SNET Server.

1. Find the Services file on your Windows machine. On Windows, the Services file is usually located in `c:\windows\services`.
2. Open the Services file using a text editor.
3. The Services file contains four columns. The rows of information can be sorted in port number order. Find the port number that is closest to the SPD Server port number (you obtained the SPD Server port number from the network administrator. See "Prepare Your Client Machine for ODBC Installation" on page 68.
4. Use the following syntax to add an entry to the Services file. Add the entry on its own line, in correct numeric order.

Table 9.1 *How to Add Service Name and Port Number to the Services File*

column1 <service name>	column2 <port number and protocol>	column3 <aliases>	column4 <comment>
spdssnet	nnnn / tcp	not required	not required
spdssnet=name	nnnn=port number		
assigned to server	protocol is always /tcp		

Remember: The service name, **spdssnet** must match the server name that you used in step 5 of the section "Configure an ODBC Data Source for SPD SNET" on page 71. The port number must match the port number on which the SPD SNET server is running.

Create a Query Using an ODBC-Compliant Program

The following instructions create a query using Microsoft Access.

1. Start the SPD SNET server.
2. Start Microsoft Access.
3. From the Microsoft Access main menu, select **File** ⇨ **Get External Table**.
4. Select **Link Table**.
5. Select **Files of Type**.
6. Select **ODBC Databases**.
7. Select the data source.

Using JDBC (Java) to Access SPD Server Tables

Requirements and Considerations for Using JDBC

To use JDBC to access SPD Server tables, you must have SPD Server tables on the network, and SPD Server and SPD SNET servers must be running on the same server as the web server. If you are working in such an environment, you might want to use JDBC if one or more of the following criterion are true:

- You do not have Base SAS software on the network client to process the data sets.
- You want to distribute the information across your corporate intranet through a web page.
- The clients on your network are varied: UNIX machines, Windows PCs, and workstations.
- The audience for the information wants point-and-click access to it.
- You want to distribute the information over the Internet.
- Your planned application requires the power of the Java programming language.

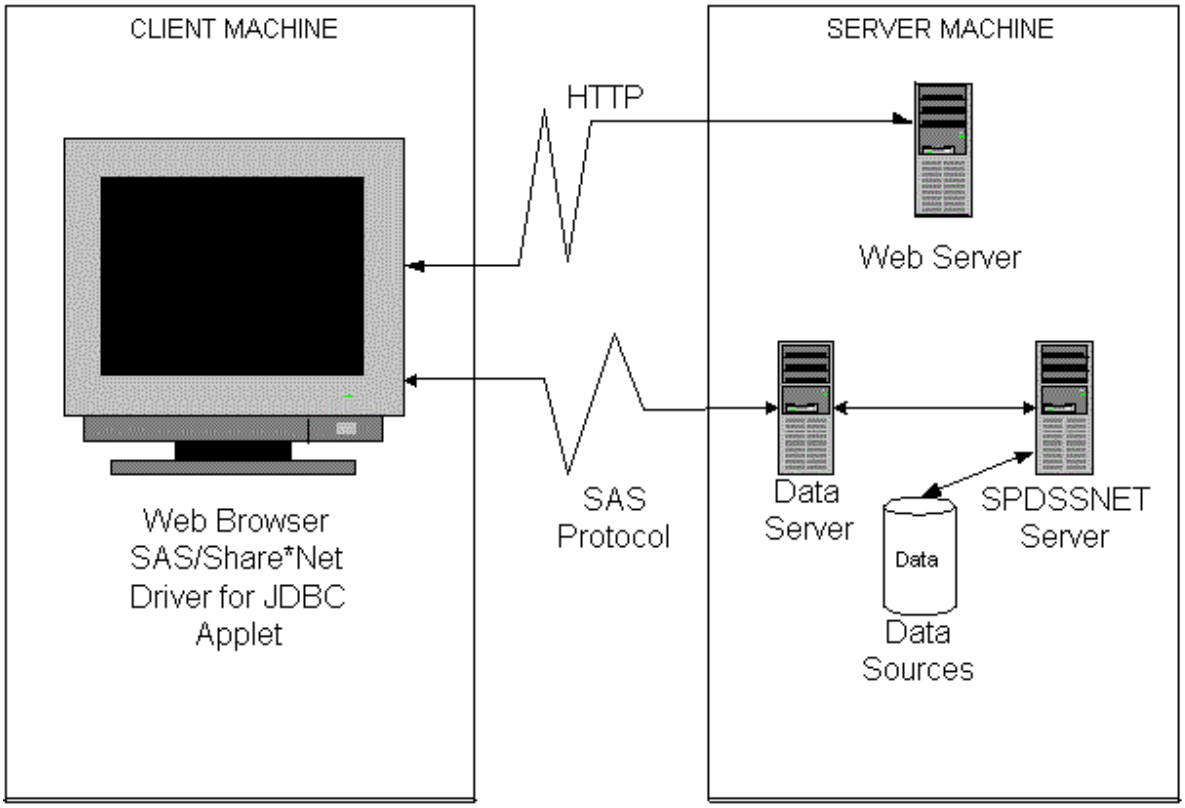
Set Up JDBC Access to the Server

JDBC access to the server is performed through the SPD Server SNET process. Review your server start-up logs to verify that the spdssnet process is called and initialized.

Set Up JDBC on the Client

To set up JDBC on the client, the client needs a browser that can accept Java applets.

Figure 9.3 *JDBC Set Up on an SPD Server Client*



Make a Query with JDBC

To use JDBC to make a query:

1. Open a browser and enter the URL for the web page that contains the JDBC code.
2. Click on the information that you are interested in. JDBC handles the request, formats the information, and returns the result to the web page.

HTML File Requirement for JDBC

The following lines must be part of the HTML file for JDBC:

```
<applet code="CLASSPATH.*.class codebase="../ width=600 height=425>
<param name=url value="jdbc:sharenet://spdssnet_node:PORT">
<param name="dbms_options" value=DBQ='LIBNAME' HOST='host_node' SERV='NNNN'>
<param name="shareUser" value="userid">
<param name="sharePassword" value="thepassword">
<param name="shareRelease" value="V9">
<param name="dbms" value="spds">
</applet>
```

Line 1

CLASSPATH points to the class path where the JDBC driver is installed.

***.class** is the name of the Java class that consumes all of the **<param name...>** lines.

Line 2

spdssnet_node is the node name of the machine on which the SPD SNET server is running.

PORT is the port number of the machine on which the SPD SNET server is running.

Line 3

value=DBQ='LIBNAME' is the LIBNAME domain of the SPD Server.

HOST='host_node' is the location of the SPD SNET server.

SERV='NNNN' is the port number of the name server.

Line 4

value="userid" is the user ID that queries the SPD Server table.

Line 5

value="thepassword" is the password of the user ID that will make the query.

Line 6

value="V9" is the version of the driver that you are using. Do not modify this value.

Line 7

Sets the foreign database property on the JDBC driver. If you specify a foreign database in line 7, the foreign database server does not need JDBC to create a SAS DataBaseMetaData object.

Limitations of Using JDBC with SPD Server

Using JDBC with SAS versus Using JDBC with SPD Server

SPD Server is treated as a foreign database. SPD Server clients cannot query the JDBC metadata class for available tables and other metadata. Users must write their own queries for these actions.

Example JDBC Query for Getting a List of Tables

The following example shows JDBC used with SPD Server:

```
SELECT '' AS qual,
LIBNAME AS owner,
MEMNAME AS name,
MEMTYPE AS type,
MEMNAME AS remarks FROM dictionary.tables AS tbl
WHERE ( memtype = 'DATA' OR memtype = 'VIEW' OR memtype = 'SYSTEM TABLE' OR
        memtype = 'ALIAS' OR memtype = 'SYNONYM')
AND (tbl.LIBNAME NE 'MAPS' AND tbl.LIBNAME NE 'SASUSER' AND tbl.LIBNAME NE 'SASHELP')
ORDER BY type, qual, owner, name
```

Example JDBC Query for Getting Metadata about a Specific Table

The following example shows using JDBC to query your data file for metadata:

```
SELECT '' AS qual,
LIBNAME AS owner,
MEMNAME AS tname, name,
length AS datatype,
```

```
type || ' ',
length AS prec,length,
length AS scale, length AS radix, length AS nullable,label,
FORMAT FROM dictionary.columns AS tbl
WHERE memname = 'your data file'
AND (tbl.LIBNAME NE 'MAPS'
     AND tbl.LIBNAME NE 'SASUSER'
     AND tbl.LIBNAME NE 'SASHELP')
```

Chapter 10

Configuring Disk Storage for SAS Scalable Performance Data (SPD) Server

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
Table data	.dpf	Large	1-to-many

File Type	File Extension	Relative Size	Number of Component Files
Index	.idx	Medium-to-large	1 or more per index
Index	.hbx	Medium-to-large	1 or more per index

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-gigabyte limit on file size that some UNIX systems impose).

The 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 LIBNAME Domain Disk Space

You define the primary file system for each LIBNAME domain for the SPD Server user base. 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

The primary file system is the base directory that you assign to the LIBNAME domain by issuing a PATHNAME= statement in the SPD Server LIBNAME parameter file, libnames.parm. Here is an example of a libnames.parm parameter file entry:

```
spdsserv -acl
   -acldir InstallDir/site
   -nameserver samson
   -libnamefile libnames.parm
```

Sample libnames.parm file entry for a UNIX system:

```
LIBNAME=all_users
  pathname=/disk1/peruser_tables;
```

Sample libnames.parm file entry for Windows:

```
LIBNAME=all_users
  pathname=d:\peruser_tables;
```

When SPD Server users create new tables in a LIBNAME domain, you must keep in mind that thee 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 should store the data and index components separately from the primary file system. Example 2 shows how to do this using ROPTIONS= with your LIBNAME statement in your libnames.parm file.

Example 2: Using ROPTIONS= to Store SPD Server Table Data and Index Component Files in Other File Systems

The following SPD Server code invokes the libnames.parm file:

```
spdsserv -acl
   -acldir InstallDir/site
   -nameserver samson
   -libnamefile libnames.parm
```

Sample libnames.parm for a UNIX system:

```
LIBNAME=all_users pathname=/disk1/peruser_tables
   roptions="datapath=('/disk2/userdata' '/disk3/userdata'
                       '/disk12/userdata' '/disk13/userdata')
   indexpath=('/disk4/userindexes' '/disk5/userindexes'
              '/disk14/userindexes' '/disk15/userindexes')";
```

Sample libnames.parm for a Windows system:

```
LIBNAME=all_users pathname=d:\peruser_tables
   roptions="datapath=('e:\userdata' 'f:\userdata')
      indexpath=('g:\userindexes' 'h:\userindexes')";
```

In Example 2, the PATHNAME= directory stores metadata files for SPD Server tables in the **`all_users`** LIBNAME 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; the primary file system is reserved for only small metadata files. The larger data and index files are stored in the other file systems specified with the DATAPATH= and INDEXPATH= options in the LIBNAME parameter file.

Example 3: Adding More File Systems to a Path Option When Its File System Is Full

If you need to add file systems to your path because an existing file system is running out of space, see the following:

```
spdsserv -acl
   -acldir InstallDir/site
   -nameserver samson
   -libnamefile libnames.parm
```

Sample libnames.parm for a UNIX system:

```
LIBNAME=all_users pathname=/disk1/peruser_tables
   roptions="datapath=('/disk2/userdata' '/disk3/userdata'
                       '/disk12/userdata' '/disk13/userdata')
   indexpath=('/disk4/userindexes' '/disk5/userindexes'
              '/disk14/userindexes' '/disk15/userindexes')";
```

Sample libnames.parm for a Windows system:

```
LIBNAME=all_users
pathname=d:\peruser_tables
   roptions="datapath=('e:\userdata'
                       'f:\userdata'
                       'i:\userdata')
   indexpath=('g:\userindexes'
              'h:\userindexes'
              'j:\userindexes')";
```

In Example 3, SAS users can continue to create more SPD Server tables, as long as space is available for the metadata files in the primary file system. When the primary file system is exhausted, you must create a new LIBNAME domain. You cannot expand storage for the .mdf components by adding the METAPATH= specification to your ROPTIONS= value in your LIBNAME parameter file. Remember the SPD Server restriction : the initial partition file for all .mdf components must be created in the primary file system (the directory that was first specified by the PATHNAME= option for the LIBNAME domain).

Chapter 11

Setting Up SAS Scalable Performance Data (SPD) Server Parameter Files

Overview of the spdsserv.parm File

The **spdssrv** process requires a parameter file at start-up. You specify the name of the parameter file using the -parmfile command-line option.

Syntax for the -parmfile Option

The syntax for the spdsserv -parmfile option is:

```
-parmfile file-spec
```

file-spec is an explicit file path for the SPD Server parameter file. The parameter file is required because it maintains options that control the SPD Server processing behavior and use of server resources. If you do not specify your SPD Server parameter file with the -parmfile option, SPD Server assumes that a file with the default name of spdsserv.parm is located in the current working directory.

Syntax for the spdsserv.parm Options

The syntax for spdsserv.parm file options is:

```
Option[ = Value] ;
```

The value is option dependent. Option keywords are not case sensitive. Comments are allowed in the SPD Server parameter file. Any value that is a memory size is stated in bytes. These values can support an "m" or "M" suffix to specify megabytes, or a "g" or "G" suffix to specify gigabytes.

Examples of SPD Server parameter files are in **`InstallDir/samples/spsdsserv.parm`**. **`InstallDir`** is a placeholder for the path to the root directory of your SPD Server installation.

Server Performance Levels

You can specify different SPD Server performance-level parameters based on the performance class of the user, or based on whether the user is assigned the SPD Server locking proxy specified by the LOCKING=YES LIBNAME option. Every SPD Server user in the SPD Server Password Manager database is assigned a performance class attribute. For more information about user performance classes, see "The Password Manager Utility psmgr" on page 158. Each performance class can be associated with

server parameters that use the performance class to control how user resources are allocated.

You enter users into the Password Manager database as either low, medium, or high-performance users. Low users get the LOW (or default) values that are specified in the spdsserv.parm file. Medium users inherit the LOW values plus any values redefined in the MED section of the spdsserv.parm file. High users inherit the LOW values plus any values redefined in the HIGH section of the spdsserv.parm file. LOCKING=YES is a special user class that is activated by a LOCKING=YES LIBNAME assignment for the user. LOCKING=YES users inherit the LOW values plus any values redefined in the LOCKING section of the spdsserv.parm file.

When you use the following keywords in the spdsserv.parm file, they associate a user with the performance-level parameter settings:

MED
: Parameters defined in this section are applied to medium performance class users.

HIGH
: Parameters defined in this section are applied to high-performance class users.

LOCKING
: Parameters defined in this section are applied to LOCKING=YES users.

Consider the following example:

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 file:

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

In the spdsserv.parm file, all low users receive the initial LOW (default) values for SORTSIZE=, BINBUFSIZE=, MAXWHTHREADS=, and WORKPATH=. Medium users inherit the LOW values, with the exceptions of the SORTSIZE= setting, which has a value of 256 MB, and the MAXWHTHREADS= setting, which has a value of 6. High users inherit the LOW values, with the following exceptions: the SORTSIZE= setting, which has a value of 1 GB, the MAXWHTHREADS= setting, which has a value of 6, and the WORKPATH= setting, which has a value of /var/hightmp. All initial values are inherited from the LOW setting. Users who are set to HIGH do not inherit any parameter settings from MEDIUM users. Therefore, you must specify MAXWHTHREADS=6 for users who are set to HIGH or MED.

The LOCKING=YES value is a special user case. All users share the same SPD Server locking proxy. For LOCKING=YES, all users inherit the LOW values in the spdsserv.parm file, regardless of 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 initial default parameter values, with the exception of the initial MAXWHTHREADS=4 setting, which changes to MAXWHTHREADS=2.

SPD Server Parameter File Options

BINBUFSIZE=

BINBUFSIZE= specifies the amount of memory to allocate for each bin buffer during a sort operation. 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 comprises 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.

Usage:

```
BINBUFSIZE= <bin-buffer-size> ;
```

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.

FEDSQLSIZE=

specifies the number of bytes in virtual address space to allocate to progressive threaded kernel memory calls in SPD Server. The default amount is 4 GB.

Usage:

```
FEDSQLSIZE=4GB ;
```

FMTNAMENODE=

specifies the server on which the user-defined formats are stored. Use FMTNAMENODE= with FMTDOMAIN= and FMTNAMEPORT=.

Usage:

```
FMTNAMENODE=d8488 ;
```

FMTNAMEPORT=

specifies the port number of the server on which the user-defined formats are stored. Use FMTNAMEPORT= with FMTDOMAIN= and FMTNAMENODE=.

Usage:

```
FMTNAMEPORT=5400 ;
```

GRPBYROWCACHE=

specifies the maximum number of memory threads that are used during parallel group aggregations. 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-megabyte 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.

Usage:

```
GRPBYROWCACHE= <memory-cache-size> ;
```

IDLE_TIMEOUT=

specifies the interval of idle time that lapses before the SPD Server client process automatically terminates the client connection. When IDLE_TIMEOUT= is greater than 0, the option is enabled. If the value is less than or equal to 0, SPD Server does not enable idle timeouts. The default value is 0.

Usage:

```
IDLE_TIMEOUT= <timeout_seconds> ;
```

INDEX_MAXMEMORY=

restricts the amount of memory that is allocated for each open index. This option affects Read operations on SPD Server tables.

Usage:

```
INDEX_MAXMEMORY= <maximum-allocated-index-memory> ;
```

INDEX_SORTSIZE=

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

Usage:

```
INDEX_SORTSIZE= <allocated-async-sort-index-memory> ;
```

LDAPSERVER=

specifies the network IP address or the host machine for the LDAP server. This value is usually the same as the IP address of the SPD Server host, which is the default value.

Usage:

```
LDAPSERVER=<ldap_server_host_ip> <LDAP-Server-IP-address-or-LDAP-Server-name>;
```

LDAPPORT=

specifies the TCP/IP port that is used to communicate with the LDAP server. The default value is LOCAL_HOST or port 389.

Usage:

```
LDAPPORT=<ldap_server_tcpip_port_number> <port-number-or-port-name> ;
```

LDAPBINDMETH=

indicates the LDAP authentication security level. The default value is LDAP_AUTH_SASL. The Simple Authentication and Security Layer (SASL) performs LDAP_AUTH_SASL and SPD Server user authentication using Digest-MD5.

Usage:

```
LDAPBINDMETH=<LDAP_SERVER_BINDMETH_STRING> <LDAP-bind-method-string> ;
```

LDAPBINDDN=

specifies the relative distinguished name (RDN) or the location in the LDAP server database where the information for the connecting client is stored. You can obtain RDN strings from the LDAP server administrator when you are configuring the SPD Server to use LDAP authentication.

Usage:

```
LDAPBINDDN=<ldap_server_binddn_string> <RDN-string> ;
```

MAXGENNUM=

specifies the maximum number of member tables that can be created in an SPD Server cluster table.

Usage:

```
MAXGENNUM= <maximum-number-of-member-tables> ;
```

MAXSEGRATIO=

controls segment candidate pre-evaluation for WHERE clause predicates with a hybrid index. 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. If you omit this value, the default value is 75%.

Usage:

```
MAXSEGRATIO= <maximum-ratio-of-segment-candidates-to-segments> ;
```

MAXSORTTHREADS=

specifies the number of parallel threads to create for a parallel sort operation. 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. If you omit this value, SPD Server assigns the value of MAXWHTHREADS= to MAXSORTTHREADS=.

Usage:

```
MAXSORTTHREADS= <maximum-number-of-parallel-sort-threads> ;
```

MAXWHTHREADS=

specifies the number of parallel threads to launch for a WHERE clause evaluation.

Usage:

```
MAXWHRTHREADS= <maximum-number-of-parallel-threads> ;
```

MINPARTSIZE=

ensures that large SPD Server tables cannot be created with an arbitrarily small partition size. Large SPD Server tables with small partition sizes create an excessive number of physical files, which increases clutter and degrades I/O performance. The default value is 16 MB. The most common values for the MINPARTSIZE parameter are in the range 128 MB–256 MB.

Usage:

```
MINPARTSIZE= <minimum-partition-size> ;
```

MINPORTNO= and MAXPORTNO=

specifies a range of port numbers that can the SPD Server user proxy processes can use to communicate with the client. You must set both the MINPORTNO= and the MAXPORTNO= option. 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.

Usage:

```
MINPORTNO=<lower-port-range-number> ;
MAXPORTNO= <upper-port-range-number> ;
```

[NO]BYINDEX

controls whether to use an index for a BY sort. The default value is NOBYINDEX, which indicates that an index is not used. The [NO]BYINDEX server parameter is used only when the SPDSNBIX= macro is set to NO (the default value).

Usage:

```
BYINDEX ; NOBYINDEX ;
```

[NO]COREFILE

controls whether the LIBNAME proxy creates a core file when an unexpected process trap occurs. The default value is NOCOREFILE.

Usage:

```
COREFILE ; NOCOREFILE ;
```

[NO]LDAP

turns SPD Server LDAP user authentication on or off. If the LDAP option is found or set during SPD Server start-up, then the SPD Server host creates a context for LDAP user authentication.

Usage:

```
LDAP ; NOLDAP;
```

[NO]NLSTRANSCODE

enables or suppresses the server-side SPD Server NLS processing. The default value for NLSTRANSCODE is NONLSTRANSCODE if the option is not found in the spdsserv.parm file. The default spdsserv.parm file for SPD Server does not contain the NLSTRANSCODE option. Users must explicitly activate server-side transcoding in SPD Server 5.1.

When you specify NONLSTRANSCODE, SPD Server treats all character column data as 8-bit raw bytes internally, regardless of the table's specified character set encoding (CEI). SPD Server 5.1 (with SAS 9.4) performs normal server-side processing of tables and ignores the CEI of the table. SAS 9.2, however, reads the CEI value of the table and performs transcoding for any pertinent character data in the rows that are returned from SPD Server.

When you specify NLSTRANSCODE, SPD Server reads the table's CEI value and the CEI value of the associated SAS 9.2 session. SPD Server does not perform transcoding if these values are the same. If the CEI values are different, SPD Server restricts the types of WHERE clause predicates that are permitted in indexed lookups. SPD Server also ensures that data is returned to SAS 9.2 using the same encoding that the SAS 9.2 session uses.

Usage:

```
NLSTRANSCODE ;

NONLSTRANSCODE ;
```

[NO]WHERECOSTING

controls whether to use dynamic WHERE costing. The default value is NOWHERECOSTING. When dynamic WHERE costing is not enabled, SPD Server uses the rules-based heuristic WHINIT.

Usage:

```
WHERECOSTING ;
NOWHERECOSTING ;
```

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. RANDOMPLACEDPF is enabled by default. To disable RANDOMPLACEDPF in SPD Server 5.1, include a NORANDOMPLACEDPF statement in your spdsserv.parm file.

Usage:

```
RANDOMPLACEDPF;
```

RANIOBUFMIN=

specifies the minimum random I/O buffer size. This value becomes the minimum I/O buffer size that is used by the proxy when it performs random I/O and table requests.

Usage:

```
RANIOBUFMIN= <minimum-random-i/o-buffer-size> ;
```

SEQIOBUFMIN=

specifies the minimum sequential I/O buffer size. This value becomes the minimum I/O buffer size that is used by the proxy when it performs sequential I/O and table requests.

Usage:

```
SEQIOBUFMIN= <minimum-sequential-i/o-buffer-size> ;
```

SORTSIZE=

controls the amount of memory to allocate for sort operations. During parallel sort operations, the memory that SORTSIZE= aallocates 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=.

Usage:

```
SORTSIZE= <memory-allocated-for-sort-operations> ;
```

STARSIZE=

controls the amount of memory to allocate for STARJOIN operations. 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.

Usage:

```
STARSIZE= <memory-allocated-for-STARJOIN-operations> ;
```

SQLOPTS=

overrides SQL default options for each SQL connect when you use the SQLOPTS= statement with an SQL RESET command. If you omit SQLOPTS=, SQL default options apply. See the For more information about SPD Server SQL RESET options, see “Specifying SPD Server SQL Planner Options” in Chapter 7 of *SAS Scalable Performance Data Server: User's Guide*.

Usage:

```
SQLOPTS= "RESET <SQL-option> [ <SQL-option>]" ;
```

TMPDOMAIN=

specifies an SPD Server domain that is defined in the libnames.parm file. The SQL query rewrite facility uses this domain to store intermediate tables.

Usage:

```
...
LIBNAME=qrw pathname=/IDX1/spdsmgr/spds45_sasdqh/qrw ;
...
TMPDOMAIN=qrw ;
```

WORKPATH=

specifies the LIBNAME proxy path for work files. If you think that the work files might overflow a single file system, you can specify multiple paths. When you specify multiple paths, enclose the complete path statement in double quotation marks.

Usage:

```
WORKPATH= "('DirPath1' 'DirPath2' ...)";
```

SPD Server Parameter File Configurations for LDAP

There are five possible SPD Server parameter file configurations for LDAP:

- **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. To run an LDAP server on the SPD Server host, add the LDAP option to your SPD Server 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. Also assume that you want to perform LDAP user authentication on the LDAP server. To run an LDAP server on the SPD Server host by using a port assignment other than LOCAL_HOST, add the LDAP option and the LDAPPORT= port specification to your SPD Server parameter file.

- **Configuration 3:** LDAP Server and SPD Server host that are running on different machines

  For this configuration, assume that you want to perform LDAP user authentication, but the LDAP server and the SPD Server hosts are on different machines. To run an LDAP server and the SPD Server hosts on different machines, add the LDAP option and the LDAPSERVER= specification (such as <*host.domain.company.com*>) to your SPD Server parameter file. An LDAP user is authenticated when the LDAP server is running at port LOCAL_HOST on *host.domain.company.com*.

- **Configuration 4:** SPD Server user IDs and passwords that are not in their default location in the LDAP database

  For this configuration, assume that you want to perform LDAP user authentication, but the SPD Server user IDs and passwords are not in their default locations in the LDAP database. Assume that all other LDAP settings use the default configuration. Add the LDAP option and the LDAPBINDDN= specification. The LDAPBINDDN= property setting is ou=people, dc=domain, dc=company, dc=com. An LDAP user is authenticated when the LDAP server is running at port LOCAL_HOST on the SPD

Server host machine. The LDAP server looks for SPD Server users at the location that corresponds to ou=people, dc=domain, dc=company, dc=com in its database.

- **Configuration 5:** SPD Server user IDs and passwords that are not in their default location in the LDAP database and in the LDAP Server that is using TCPIP_PORT

  For this configuration, assume the following: You want to perform LDAP user authentication. The SPD Server user IDs and passwords are located at ou=people, dc=domain, dc=company, dc=com in the LDAP database, and the LDAP server is using the port TCPIP_PORT. Add the LDAP option and set the LDAPPORT= port specification to TCPIP_PORT in your SPD Server parameter file. Next, add the LDAPBINDDN= specification. The LDAPBINDDN= property setting is ou=people, dc=domain, dc=company, dc=com. A user is authenticated when the LDAP server is running at port TCPIP_PORT on the SPD Server host machine. The LDAP server looks for SPD Server users at the location that corresponds to ou=people, dc=domain, dc=company, dc=com in its database.

SPD Server Parameter File Validation

The SPD Server parameter file is validated during a system start-up, and when refreshed by the PROC SPDO REFRESH PARMS command. For more information about the REFRESH PARMS command, see REFRESH PARMS"Overview of the REFRESH Command" on page 195. Any errors in the server parameter file at start-up will cause the SPD Server program start to fail. Errors are reported in the SPD Server log file.

Errors that occur during the server parameter file refresh will cause the refresh to fail, and the server will 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.

Pathname Substitution

You can use the keyword @HOSTNAME@ in your SPD Server parameter file WORKPATH= option to substitute the short host name in the workpath.

Usage:

```
workpath= /work/@HOSTNAME@ ;
```

Pathname substitution enables you to use the same parameter file on different hosts, when each host requires its own workspace area.

SPD Server Parameter File Configurations for Auditing

[NO]WHEREAUDIT

Enables audit logs for WHERE clauses that are submitted to SPD Server. Specify the WHEREAUDIT option in the spdsserv.parm file to enable the audit logs. The spdslog message logger logs messages, and the spdsaud audit logger logs audits. If you use the

WHEREAUDIT option, both the spdslog log file and the spdsaud log file contain WHERE statement information.

Usage:

```
WHEREAUDIT;

NOWHEREAUDIT;
```

The WHEREAUDIT option enables audit logging for the server. It also enables automatic audit log file creation by spdsaud. The file specification is a partial pathname or filename that is used to generate the complete audit filename. For example, if your file specification is **`/audit/spds`**, the generated filename is **`/audit/spds_mmddyyyy.spdsaudit`**. In the output, **`mmddyyyy`** is the system date when the audit log file was created.

SQLAUDLEN=

Specifies the maximum size of the SQL statement in the audit log when the following conditions are both true: proxy auditing is enabled in SPD Server, and the WHEREAUDIT option is specified.

Usage:

```
SQLAUDLEN=<maximum-number-of-characters-in-SQL-statement>
```

The default value for SQLAUDLEN is 1,024 characters. The maximum value is 4,096 characters.

WHAUDLEN=

Specifies the maximum size of the WHERE clause in the audit log when the following conditions are both true: proxy auditing is enabled in SPD Server and the WHEREAUDIT option is specified.

Usage:

```
WHAUDLEN=<maximum-number-of-characters-in-WHERE-clause>
```

The default value for WHAUDLEN is 512 characters. The maximum value is 4,096 characters.

Chapter 12

Setting Up SAS Scalable Performance Data (SPD) Server Libname Parameter Files

Overview of the libnames.parm File

When SPD Server starts, it reads the information stored in the libnames.parm file. The libnames.parm file establishes the names and file storage locations of SPD Server domains during the server session. SPD Server administrators can use the libnames.parm

file as a central tool to control SPD Server domain resources and user access to SPD Server domains.

Domain Naming Syntax for libnames.parm

To define an SPD Server domain in the libnames.parm file, you must define the domain as a LIBNAME and define the path that points to the directory in which data files for the domain are stored.

```
LIBNAME=domain-name PATHNAME=primary-metadata-path
<optional specifications>
      OWNER=owner-id
   LIBACLINHERIT=<YES/NO>
   DYNLOCK=<YES/NO>
   BACKUP=<YES/NO> ;
```

The domain name that is associated with the LIBNAME must follow standard SAS LIBNAME nomenclature rules. The PATHNAME= specification defines the computing path that contains the metadata tables that are associated with the domain. By default, the PATHNAME= specification also contains the data tables, index tables, and intermediate tables that the domain creates. SPD Server administrators and users can use the options described in "LIBNAME Domain Path Options" on page 96 to enhance computational performance by specifying separate paths for domain data, index, and work tables. All SPD Server domain names must be unique. Different SPD Server domains should never share the same domain path.

Here are some examples of simple libnames.parm file domain declarations:

```
LIBNAME=spds123 PATHNAME=c:\data\spds123;

LIBNAME=123spds PATHNAME=c:\data\123spds;

LIBNAME=_under PATHNAME=c:\data\_under;

LIBNAME=under_ PATHNAME=c:\data\under_;
```

The libnames.parm file is the preferred method to declare domains for use in SPD Server. Users can connect to domains by submitting SAS code to SPD Server after a session has started. The following example SAS code connects to the first domain in the previous example:

```
LIBNAME spds123 sasspds 'spds123'
  server=d8488.5200
  user='anonymous';
```

LIBNAME Domain Path Options

You can specify optional path parameters for a domain in libnames.parm libref statements. You specify these optional path parameters using either standard option statements or using reserved option statements. The difference between non-reserved and reserved option statements is that non-reserved option statements can be altered by subsequent libref statements that are submitted to SPD Server with SAS code.

Use the following syntax to specify optional path parameters in the libnames.parm file for ROPTIONS:

```
LIBNAME=domain-name
  PATHNAME=primary-metadata-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')";
```

DATAPATH=
specifies a list of paths that contain SPD Server data tables that are associated with the declared domain. You use ROPTIONS to specify DATAPATH=.

Usage:

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

INDEXPATH=
specifies a list of paths that contain SPD Server index tables that are associated with the declared domain. You use ROPTIONS to specify INDEXPATH=.

Usage:

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

WORKPATH=
specifies a list of paths that contain temporary SPD Server work tables and temporary SPD Server intermediate files that are associated with the declared domain. You use ROPTIONS to specify WORKPATH=.

Usage:

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

METAPATH=
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 statement becomes full. The additional metapaths provide a buffer space that can be used for Update and Append operations to existing SPD Server tables.

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 gigabytes of metadata for every terabyte of compressed physical data.

You can use OPTIONS or ROPTIONS to specify METAPATH=.

Usage:

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

Path Substitution

You can use the keyword @HOSTNAME@ in your libnames.parm primary path or ROPTIONS= path to substitute the short host name in the path. Path substitution is useful when SPD Server is installed on a shared file system, 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 file, but substitutes the unique pathnames that each host requires.

```
LIBNAME host_specific= /data/@HOSTNAME@;
```

Verification of libnames.parm File

The libnames.parm file is verified on system start-up and when the system is refreshed. The validation includes syntax checking and path validation. Libnames.parm 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 file during SPD Server start-up, the session is not launched. If errors are detected in the libnames.parm file during a refresh, the refresh is cancelled and the previous libnames.parm definitions are used.

Domain Access Options

A domain is an SPD Server resource that enables you to access and manipulate SPD Server tables, views, and so on. When you issue a libref statement to create a domain for SPD Server, you can use the following optional specifications to control the accessibility of resources among other SPD Server users.

LIBNAME access to the OWNER= domain controls Read or Write access to the domain. LIBNAME access to the OWNER= domain does *not* grant access to the objects in the domain. Once the LIBNAME is established, normal SPD Server ACL access rules are used to grant or deny access to the objects in the domain.

OWNER=

specifies the owner of a domain. The SPD Server owner controls LIBNAME access to the domain. The domain owner can create a LIBNAME ACL on the domain to grant or deny LIBNAME access to other SPD Server users. When the domain is specified with an owner, only the owner can use the TEMP=YES LIBNAME option with the domain.

Note: The owner can use LIBNAME ACL to grant the following access levels:

- Read access to allow a user or group to get a LIBNAME to the domain to read SPD Server tables that the user or group has access to
- Write access to allow a user or group to create new objects in the domain

- Control access to allow a user or group to modify the owner's LIBNAME ACL

Usage:

```
OWNER=owner-id
```

LIBACLINHERIT=
controls the ACL precedence of permission checks. For more information about the LIBACLINHERIT= domain access option and its use, see "Controlling the Precedence of Permission Checks with the LIBACLINHERIT= Option and the OWNER= Option" on page 99.

DYNLOCK=
specifies whether dynamic locking is enabled. If you omit the DYNLOCK= option, the default SPD Server setting is NO. For more information about dynamic locking, see "Dynamic Locking" on page 102.

Usage:

```
DYNLOCK=<YES/NO>
```

BACKUP=
controls whether the objects in the domain can be backed up or restored using the SPD Server Backup and Restore utilities. For more information about back up and restore utilities, see "SAS Scalable Performance Data (SPD) Server Backup and Restore Utilities " on page 212.

Usage:

```
BACKUP=<YES/NO>
```

Controlling the Precedence of Permission Checks with the LIBACLINHERIT= Option and the OWNER= Option

Precedence of permission checks for a LIBACLINHERIT=YES OWNER= domain can include inheriting the permissions of the LIBNAME ACL of the user for resources that are owned by the domain owner. This allows the domain owner to create domain resources that other users can access via LIBNAME ACLs, without having to create user or group ACLs for those resources.

When a suer attempts to access resources in a LIBACLINHERIT=YES OWNER=<*domain-name*>, the following ACL precedence of permission checks are made on the resource:

1. If user-specific ACLs are defined on the object for the user, the user gets these permissions.
2. If group-specific ACLs are defined on the object for the user's group, the user gets these permissions.
3. If LIBNAME ACL permissions are defined for the user and the resource belongs to the OWNER= of the domain, then the user gets the domain LIBNAME ACL permissions on the object.
4. If LIBNAME ACL permissions are defined for the user's group and the user is a member of the OWNER= group of the domain, then the user gets the LIBNAME ACL group permissions on the object.

5. If universal ACLs are defined on the object for the user, the user gets the universal ACL permissions.

An OWNER=<*owner-name*>LIBACLINHERIT=YES domain statement uses a slightly different methodology. When the owner specifies the OWNER= parameter with LIBACLINHERIT=YES, the owner can grant the following access levels:

- Read access to allow a user or group to get a LIBNAME to the domain
- Alter access to allow a user or group to create new objects in the domain
- Control access to allow a user or group to modify the owner's LIBNAME ACL

The owner can use Alter access with OWNER= and LIBACLINHERIT=YES to allow a user or group to create a new resource in the domain. Alter access is sometimes preferable to Write access for an OWNER= domain. Alter access prevents users or groups that inherit Write access from writing to, updating, or deleting resources that were created by the domain owner. When the owner uses LIBNAME Alter access with OWNER= and LIBACLINHERIT=YES, the owner can grant privileges to users to create objects in the domain. The owner can use Write access to inherit Write access to the owner's resources.

The following example shows SAS code submitted to SPD Server using LIBACLINHERIT. The example begins with information in the libnames.parm file where domain names and paths are declared.

Contents of the libnames.parm file:

```
LIBNAME=libinher
PATHNAME=/IDX1/spdsmgr/spds45test/libinher
LIBACLINHERIT=YES
OWNER=admin ;

LIBNAME=noinher
PATHNAME=/IDX1/spdsmgr/spds45test/noinher
OWNER=admin ;
```

SAS code submitted to SPD Server by the user:

```
LIBNAME libinher sasspds 'libinher'
  server=gomez.5129
  user='admin'
  password='spds123' ;

LIBNAME noinher sasspds 'noinher'
  server=gomez.5129
  user='admin'
  password='spds123' ;

data libinher.admins_table
  noinher.admins_table ;

  do i = 1 to 10 ;
    output ;
end ;
run ;


/* LIBNAME access for user anonymous */
  PROC SPDO library=libinher ;
```

```
/* Admin owns these ACLs */
  set acluser admin ;


/* Add a LIBNAME ACL to d1 */
  add acl / LIBNAME ;


/* Modify LIBNAME ACL Domain d1  */
/* Allow users in Group 1        */
/* read-only access to domain    */

modify acl / LIBNAME read ;

list acl _all_ ;
quit ;


/* Set up LIBNAME access for  */
/* user anonymous             */
PROC SPDO library=noinher ;


/* Specify who owns these ACLs */
  set acluser admin ;


/* add a LIBNAME ACL to d1 */
  add acl / LIBNAME ;


/* Modify LIBNAME ACL Domain d1 */
/* Allow users in Group 1 read- */
/* only access to the domain    */

modify acl / LIBNAME read ;

list acl _all_ ;
quit ;

LIBNAME a_inher sasspds 'libinher'
  server=gomez.5129
  user='anonymous' ;

LIBNAME a_noher sasspds 'noinher'
  server=gomez.5129
  user='anonymous' ;

PROC PRINT data=a_inher.admins_table ;
  title 'with libaclinher' ;
run ;

PROC PRINT data=a_noher.admins_table ;
  title 'without libaclinher'
run ;
```

Dynamic Locking

Overview of Dynamic Locking

Dynamic locking is an SPD Server feature that gives multiple users concurrent access to SPD Server tables. Multiple users can perform read and write functions (insert, append, delete, and update), and the integrity of the table contents is preserved.

Dynamic locking is enabled or disabled at the domain level. All tables that are stored within the domain are subject to the setting of the dynamic locking feature. The DYNLOCK= statement should be used in libnames.parm file domain declarations.

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

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, which 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.

Dynamic locking can also provide better performance than record-level locking. Dynamic locking has performance advantages over record-level locking when concurrent READ and WRITE access to a table is required. This advantage is due to the more distributed processing and parallelism of that occurs when multiple SPD Server user proxies are used. The performance benefit depends on the opportunities for parallelism, and you should quantify the benefit on a case-by-case basis.

How Dynamic Locking Works

To use dynamic locking, SPD Server tables must be part of a named SPD Server domain. When dynamic locking is enabled for a domain, all of the SPD Server users that access tables in that domain automatically use dynamic locking. The SPD Server clients do not need to set any additional parameters to take advantage of the benefits of dynamic locking.

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.

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 file to optimize how SPD Server stores files. "Domain Access Options" on page 98 describes how to specify named paths for the three data components of SPD Server tables (observation data tables, index data tables, and metadata tables), and how to specify paths for temporary intermediate calculation tables. LIBNAME domain declaration statements can specify the system paths that are associated with each table space 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 table spaces that are defined by the DATAPATH=, INDEXPATH=, WORKPATH=, and METAPATH= options of the LIBNAME domain declaration statements. Use this information to determine the best sizing, I/O, and redundancy requirements to optimize performance and scalability for named SPD Server domain paths.

Data Table Space

When you declare a domain in a LIBNAME statement, 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 table space 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 table space using batch processes during evenings or off-peak hours. You can restrict access to data table space to Read-Only access for all users except administrators who perform the load and refresh processes.

To ensure reliability, organize data table space 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 Table Space

When you declare a domain in a LIBNAME statement, 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 table space, 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 Table Space

When you declare a domain in a LIBNAME statement, metadata tables are stored in the space that is defined in the PATHNAME= specification. If the space configured in PATHNAME= is full, SPD Server stores overflow metadata for existing tables in the space that is defined in the METAPATH= specification, if it is declared. 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 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 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 WORKPATH=.

A workspace is an area on disk that SPD Server software 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.

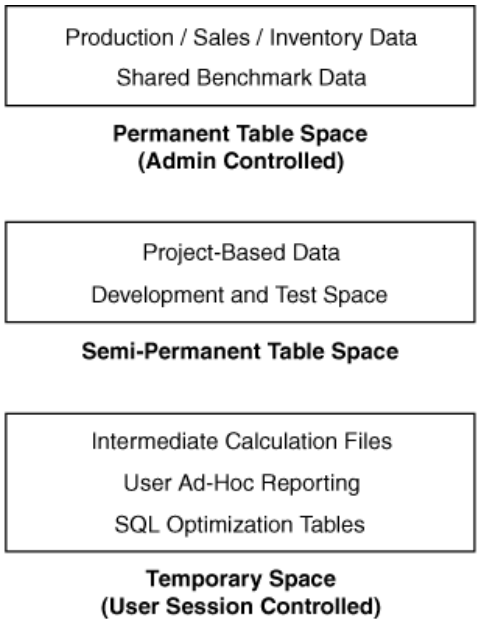
Workspace on disk is typically a RAID 0 configuration or a hardware-redundant RAID design. RAID 0 configurations are risky because if the RAID 0 disk goes down, the

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 SPD Server domain space, you can use domain declarations in the libnames.parm 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 table spaces. 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.



Permanent Table Space

In SPD Server, large production, inventory, and sales data storage areas work best using permanent table space. 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 table spaces devoted to production reporting. In this situation, permanent table space should be accessible to a specific group (such as analysts) of regular SPD Server users.

You can use the libnames.parm 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 table space, use the optional DATAPATH=, INDEXPATH=, and OWNER= statements on the LIBNAME domain statement in the libnames.parm file to specify unique, appropriately sized disk areas for data tables and index tables. The OWNER= statement configures ownership and access. You must ensure that the paths named in domain declarations have access to sufficient disk space.

You can grant user access to permanent table spaces by using individual user account access privileges, or by establishing an ACL group of approved users through the owner of the domain. LIBNAME domain statements create permanent table space by default.

Semi-Permanent Table Space

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, but 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 SPD Server users can configure semi-permanent table space. However, administrators should allocate semi-permanent spaces using the libnames.parm file.

Temporary Table Space

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 table space. The life span of temporary table spaces begin and end with the user's SPD Server sessions.

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

Any report might require temporary table space for intermediate calculation tables. SPD Server users can configure temporary table space with the LIBNAME domain statements

that they submit during an SPD Server session. To create temporary table space, users specify the optional TEMP=YES option when they issue the LIBNAME domain statement in the SPD Server job code. All tables in temporary table space are lost at the end of the SPD Server user session, when temporary table space is automatically deleted.

Example libname.parm File Configurations

The following SPD Server code examples show the range of LIBNAME domains that you can create using the libnames.parm file. The code examples begin with the simplest form of LIBNAME domain declaration and increase in complexity.

Example 1: Minimum Configuration for Domain Declaration

This example contains the syntax that is required for the simplest form of LIBNAME domain configuration:

```
LIBNAME=SKULIST PATHNAME=c:\data\skulist;
```

This statement creates the SPD Server LIBNAME domain SKULIST. All SPD Server 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`**.

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

This example contains the syntax that is required to declare a LIBNAME domain with separate paths allocated for the domain 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 domain path 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. In SPD Server 5.1, index components can take advantage of the SPD Server RANDOMPLACEDPF feature. You can configure smaller disk partitions for index space with the RANDOMPLACEDPF feature, which benefits SPD Server backup and recovery operations.

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

Example 3: Query-Rewrite Domain Configuration

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

The SPD Server 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 table space, even if they are not submitting code for an SPD Server SQL query rewrite job.

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

- Create a specific domain for the query rewrite operations in the libnames.parm file. This example names the query rewrite domain **QRW**.
- In the spdsserv.parm file, include a TMPDOMAIN=<QRW-*domain-name*> statement that references the QRW domain that you created in the libnames.parm file.

The libnames.parm file code (the LIBNAME=QRW statement creates a specific 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')";
```

The spdsserv.parm file code (the TMPDOMAIN=QRW statement references the domain created for query rewrite tables):

```
TMPDOMAIN=QRW;
```

Example 4: Multiple Domain Types and Paths Configuration

This example uses a combination of libnames.parm, spdsserv.parm, and user-issued SAS code that is submitted to SPD Server to create 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.

The libnames.parm file code defines the domain named 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 file code defines the domain named 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 file code defines the domain named SPDTEMP, which contains temporary table space that is automatically deleted at the end of the SPD Server 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 file code uses the TMPDOMAIN=SPDTEMP statement to reference the domain that was created for temporary tables. This code also uses the WORKPATH= statement to identify an array of RAID-enabled disk paths for temporary SPD Server work tables and temporary SPD Server 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 SPD Server 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;
```

Chapter 13

Setting Up SAS Scalable Performance Data (SPD) Server Performance Server

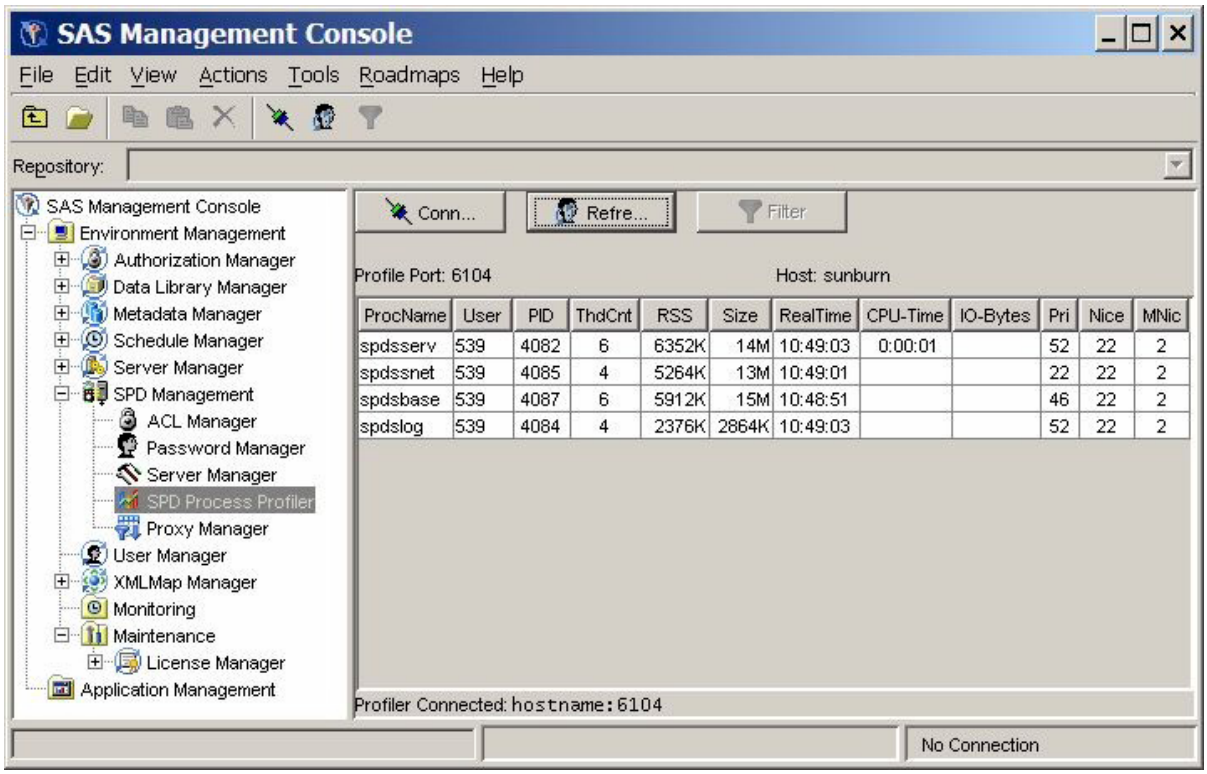
Overview of SPD Server Performance Server

SAS SPD Server 5.1 provides a performance monitoring server called spdsperf. SPD Server Performance Server is an optional component and is not required for the normal operation of SPD Server.

Note: SPD Server Performance Server is currently not available for the Windows or Linux X64 platforms.

SPD Server Performance Server gathers SPD Server process performance information and posts it to the SPD **Server Management** section of the SAS Management Console application. The information consists of memory and resource allocations by users, and SPD Server processes that were spawned by an SPD Server name server. All SPD Server users must connect to an SPD Server name server before their SPD Server session is spawned. Each SPD Server name server owns a dynamic family of subordinate SPD Server processes that SPD Server users and jobs create and terminate.

The information that is gathered by SPD Server Performance Server is stored in the SAS Management Console. The SAS Management Console has a folder that is reserved for SPD Server management. The **SPD Management** folder is a child of the **Environmental Management** folder in the SAS Management Console. 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 single SPD Server process that was spawned on the SPD Server name server that resides on the specified port ID (PID).



The SPD Process Profiler displays information about memory and resource allocations. For this reason, you can use the SAS Management Console to review which SPD 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.

Not only can you display performance summary statistics in the SAS Management Console application. You can also configure the SPD Server 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 SPD Server Performance Server created.

Starting SPD Server Performance Server

Overview of Starting the Performance Server

You can start SPD Server Performance Server in two ways. You can invoke the Performance Server on the command line, or you can launch it by calling the rc.perf script that is configured for your location's SPD Server installation.

By default, the SPD Server Performance Server process displays the captured performance data on the user's screen. To disable the user screen display, redirect stdout and stderr to **`/dev/null`**. If you redirect the screen output, it is easier to run spdsperf in the background, or as an orphan.

Start Performance Server from the Command Line

You can start the SPD Server Performance Server from a UNIX command line. SPD Server and the SAS Management Console applications must be running before you start the Performance Server. If you need to restart SPD Server, you must also shut down and restart the SPD Server Performance Server after SPD Server is restarted. The Performance Server utility is not compatible with SPD Server releases earlier than SPD Server 4.4.

Use the following command to start Performance Server:

```
spdsperf -g SMID -n NSP -s SNP -p PLP -l LOG [-i SEC] [-c CNT]
```

SMID
SMID is the shared memory ID that is passed to the spdsbase user proxy when it is started by SPD Server. SMID contains information that spdsperf requires. You can find the shared memory ID by issuing a UNIX process status report command for all processes and then grep for 'spdsbase', using a form similar to **`ps –eo args | grep spdsbase`**. The shared memory ID should be the first parameter (after the process name) that is passed to the process.

NSP
the process-ID number of the SPD Server name server whose family of spawned processes you want to monitor.

SNP
the process-ID number of the SPD Server SNET Server.

PLP
the listening port number that the SAS Management Console uses to contact the Performance Server.

LOG
the path to which you want to write the profile log.

SEC
an optional property that specifies the number of seconds that transpire between instances of performance monitoring data captures. Valid values are integers that are greater than or equal to 1 and less than or equal to the SPD Server MAXINT value.

CNT
an optional property that specifies the number of performance monitoring data captures that you want the Performance Server to take. Valid values are integers that are greater than or equal to 0 and less than or equal to the SPD Server MAXINT value. The value 0 specifies an infinite number of data captures.

Start Performance Server by Using the rc.perf Script

You can also start SPD Server Performance Server by calling the rc.perf script during start-up. See "Sample rc.perf Script" on page 116 for a sample rc.perf script that you can cut and paste into an editor of your choice, and customize for your use. SPD Server 5.1 is also shipped with a sample rc.perf script that you can modify. The sample rc.perf script is in your SPD Server installation folders at **`.../samples/perfmon/rc.perf`**. You can use either example file to create a custom rc.perf script.

Make the following changes when you customize your version of the rc.perf script:

1. If your SPD Server installation uses a custom SPD Server installation path, modify the INSTDIR path setting to specify your installation path.

2. Modify the UNIX environment setting for DISPLAY. This environmental variable tells the X server where to display the window for the Performance Server program.
3. If your SPD Server installation uses custom NSPORT and SNPORT assignments, modify the NSPORT and SNPORT settings in the sample script to specify the port addresses that your SPD Server installation uses.
4. The script uses the -PARGS setting to specify how many times the Performance Server should capture performance information snapshots before shutting down. The sample 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.

Sample rc.perf Script

The following sample code is a typical rc.perf script that you can modify for use at your own site. Follow the instructions in "Start Performance Server by Using the rc.perf Script" on page 115 to customize the script for your SPD Server installation. Copy and paste the example code into a text editor. Make your changes, and then save the file to your SPD Server installation in a location where the script can be called for execution.

```
#!/bin/ksh
#-------------------------------------------------------------------------
#
# PURPOSE:    Start the SPD Performance Profiler for the specified servers.
#
# PARAMETERS: version - Version of SPDS to build and run (e.g., dev, 403).
#
# NOTES:      Common optional parameters:
#               -nsport    overrides NSPORT for server.
#               -snport    overrides SNPORT for server.
#               -debug     use alternate port numbers for development.
#
#             The default repetition count for spdsperf is 3. This script
#             over-rides the default to run indefinitely. Supplying a -c
#             option to this script will over-ride this new default.
#
# HISTORY:    12Sep06 mjm Optimized for customer use.
#             02Aug06 mjm Created.
#
#-------------------------------------------------------------------------


#
# enable XPG4 versions of ps command on some platforms
#
export UNIX95=1


#
# initialize variables
#
NSPORT=6100
SNPORT=6101
```

```
DEBUG=
PARGS="-c 0"

#
# parse parameters
#
while [ $1 ]; do
    #echo "Parsing Option $1 of length ${#1}"
    case "$1" in
        -nsport) if [ $# -lt 2 ]; then
                    echo "$1 parameter value not specified"
                    exit 1
                    fi
                NSPORT=$2
                shift;;
        -snport) if [ $# -lt 2 ]; then
                    echo "$1 parameter value not specified"
                    exit 2
                    fi
                SNPORT=$2
                shift;;
        -debug)  DEBUG="YES";;
        -trace)  echo "********\n* Script: $0\n* Args: $*\n********"
                set -x
                trace="-trace"
                echo "Script tracing turned on";;
        *)       echo "Found unknown arg, passing on to profiler."
                                PARGS="$PARGS $1";;
        esac
    shift
    done

echo "NSPORT=$NSPORT"
echo "SNPORT=$SNPORT"
echo "DEBUG=$DEBUG"
echo "PARGS=$PARGS"


#
# Check for debug option
#
if [ -n "$DEBUG" ]; then
        NSPORT=9876
        SNPORT=9877
        echo "Using Debug Ports: NS=$NSPORT   SN=$SNPORT"
    fi


SSRVPID=$(ps -eo pid,ppid,args | grep spdsserv | grep 6100
  | tr -s "      " " " | sed -e "s/^ *//" | cut -d " " -f1)

SNETPID=$(ps -eo pid,ppid,args | grep spdssnet | grep 6101
  | tr -s "      " " " | sed -e "s/^ *//" | cut -d " " -f1)

SHMATID=$(ps -eo pid,ppid,args | grep spdsbase | grep $SSRVPID
  | tr -s "\t" " " | sed -ne "1s/^ *//p" | cut -d " " -f4)
```

```
echo "SPDSNSRV Pid: $SSRVPID"
echo "SPDSSNET Pid: $SNETPID"
echo "SHMATID:      $SHMATID"

INSTDIR=/usr/local/spds
PATH=$INSTDIR/bin
export PATH
LD_LIBRARY_PATH=$INSTDIR/bin
export LD_LIBRARY_PATH
LIBPATH=$INSTDIR/bin
export LIBPATH

# substitute user's display machine name below.
export DISPLAY=machine:0.0

#sleep 4
spdsperf -g $SHMATID -n $SSRVPID -s $SNETPID $PARGS
```

Performance Server Log File

You can configure SPD Server Performance Server to save the process performance information to a text log file. Your SPD Server installation includes a PERL utility called process_perf_log that is in the **.../samples/perfmon** directory of your SPD Server installation. When you use the process_perf_log PERL script with your SPD Server Name Server log files, the files are parsed and formatted for SAS processing.

A sample SAS script for importing the parsed log file data is in the **.../samples/perfmon/PerfDataSample.sas** directory in your SPD Server installation.

Part 5

SAS Scalable Performance Data (SPD) Server Security

Chapter 14

ACL Security Overview

SPD Server ACL Security Overview

SPD Server uses access control lists (ACLs) and SPD Server user IDs to secure domain resources. Users obtain their user ID and password from the SPD Server administrator.

SPD Server also supports ACL groups, which are similar to UNIX groups. You can associate an SPD Server user as many as five ACL groups.

ACL permissions affect all SPD Server resources, including domains, tables, table columns, catalogs, catalog entries, and utility files. SPD Server grants access rights only to the owner (creator) of an SPD Server resource. Resource owners can use PROC SPDO to grant Read, Write, and Alter ACL permissions to a specific group (called an ACL group) or to all SPD Server users.

The resource owner can use the following properties to grant ACL permissions to all SPD Server users:

READ
grants universal Read access to the resource (read or query)

WRITE
grants universal Write access to the resource (append to or update)

ALTER
grants universal Alter access to the resource (add, rename, delete, or replace a resource, and add or delete indexes associated with a table)

The resource owner can use the following properties to grant ACL permissions to a named ACL 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)

SPD Server ACL Security Model

Overview of the ACL Security Model

SPD Server provides a security system that is based on Access Control Lists, or ACLs.

SPD Server comes bundled with SAS Management Console. SAS Management Console is a GUI utility that you can use to manage passwords and ACLs. SAS Management Console manages passwords using the same capabilities that the **psmgr** utility provides, and it also manages ACLs using the same capabilities provided by PROC SPDO.

ACL Security

UNIX File-Level Protection with ACL Security

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 or write only files that have the appropriate file and directory access permissions to the SPD Server's user's ID. Use the UNIX **umask** command to restrict the permissions for creating files.

Validating User IDs and Passwords

SAS users must issue a user ID and password with the LIBNAME statement to connect to SPD Server. The user ID and password are verified against an SPD Server user ID table that is set up by the system administrator. You can enforce the expiration of passwords by using the psmgr administration tool for the user ID table or by using SAS Management Console, if it is installed and configured for SPD Server. Whether you are in a Windows or UNIX environment, you can prevent logins under the anonymous user ID by placing user **anonymous** in the user ID table with a password that is unknown to the SAS users.

Controlling LIBNAME Domains with ACL Security

You define the valid LIBNAME domains by adding entries in the LIBNAME parameter file for each SPD Server. The PATHNAME= specification defines the file system for the LIBNAME. LIBNAME= specifications provide the access route to the file system. Restricting knowledge of the LIBNAME= specification information restricts access to the corresponding file systems.

User Ownership of LIBNAME Domains and Tables

In the LIBNAME parameter file, you can attach the OWNER= specification to any defined LIBNAME domain. Only the system user whose user ID matches the OWNER= specification can create tables in this domain. (However, that user can grant other users Read or Write access rights through ACLs that were issued from the SAS LIBNAME statement.)

Each table created is tagged with the SPD user ID (referred to as the owner) of the user who created it. Only the owner or ACLSPECIAL users can access a table. (However, the owner can grant access to other users through ACLs by adding a LIBNAME ACL with PROC SPDO.)

Granting Permissions with ACLs

An SPD Server ACL permits three distinct levels of permission on a resource. First, you can grant Universal permissions to SPD Server users who are not in the same ACL group as the resource owner. Second, you can grant Group permissions to SPD Server users who are in the same ACL group as the resource owner. Third, you can grant User permissions to a specific SPD Server user ID. The precedence of permission checks is as follows:

1. Check user-specific permissions first. If permissions are defined, the accessor is granted these permissions.
2. If a resource is owned by the same ACL group as the accessor, the accessor is granted the resource's Group permissions.
3. If the resource is owned by a different ACL group than the accessor, the accessor is granted the resource's Universal permissions.

ACL Concepts

ACL Groups

ACL groups are similar to UNIX groups. Each SPD Server user ID can belong to one or more ACL groups.

You can associate a given SPD Server user ID with up to five ACL groups. When a user connects to an SPD Server using a LIBNAME assignment, the user specifies a specific

ACL group by using the ACLGRP= option. The ACLGRP= value in the user's LIBNAME assignment must match one of the five groups that you defined for that user. If the user does not specify ACLGRP= in the LIBNAME assignment, the SPD Server affiliates the user with his or her default ACL group (which is the first group in the list).

When you define user-specific ACL permissions, you can use an ACL group wherever you can use an explicit SPD Server user ACL. Using an ACL group grants permissions to the ACL group instead of to only a specific SPD Server user.

Column Security

You can control access to table contents at the column level by using ACLs. You can apply column-level security ACLs to individual users at the user level, or to collections of users at the group level. SPD Server enforces precedence for user and group ACL permissions: first, user ACL permissions are applied, then group ACL restrictions are applied. SPD Server user permissions override SPD Server 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* individual 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 security clearance level will not be certified for several weeks. All members of the department use an SPD Server 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 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. You 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. Instead of issuing user-level column ACL permissions to the rest of the testing group individually, you issue a group-level ACL column permission to the user group Testgroup. The permission must explicitly grant access to the protected column.

   After a column is protected with ACL security, you must grant explicit permissions in order for any user (or groups 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 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.

   Joe is a member of Testgroup, but the user-level ACL permissions maintain precedence over group-level ACL permissions. All the members of Testgroup have full table access, except Joe. Joe's user-level ACL column security restriction prevents him from accessing the classified column.

Now consider another scenario, in which John manages a group Devgroup whose members record their billable project hours and codes in an SPD Server 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 ACLs override any group-level ACLs for the Devgroup table.For example code that implements column-level security, see "Column-Level Security Example" on page 151.

Generic ACL

You can use generic ACL names for a class of resources that have a common prefix. You can use an asterisk (*) as a wildcard character to make a single ACL entry instead of making explicit entries for each resource. For example, if you have tables named Salesne, Salesse, Salesmw, Salessw, Salespw, and Salesnw, you could use the wildcard character to create the generic ACL name, Sales*, to include all of them. You then would define your ACL permissions on the Sales* generic ACL.

When you are using the SPDO procedure, use the /GENERIC command option to identify a generic ACL. (See "Controlling SPD Server Resources with PROC SPDO" on page 126.)

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.

LIBNAME ACL

You can control access permissions to an entire LIBNAME domain by using the SPD Server ACL facility. When you are using PROC SPDO, use the /LIBNAME option to identify the LIBNAME domain ACL.

Persistent ACL

A persistent ACL entry is an ACL that is not removed from the ACL tables when the resource itself is deleted. When you are using PROC SPDO, use the /PERSIST command option to identify a persistent ACL.

Resource

A PROC SPDO resource can be one of the following items:

- a table (data set)
- a table column (data set variable)
- a catalog
- a catalog entry
- a utility file (for example, a VIEW, an MDDB, and so on)
- a LIBNAME domain

Two-Part Resource Name

Two-part names identify a column entry in a table. Use the SAS convention *table.column* when you specify the table and column that you want to secure.

When you issue SPDO commands, you can use two-part names in any context that defines, modifies, lists, or deletes table-related ACLs. You can also specify the reserved word _ALL_ as the column name when you issue SPDO commands that support the _ALL_ resource name.

Giving Control to Others

You can permit other SPD Server users to alter your own ACL entry by granting a specific user or group access to that ACL entry. See "MODIFY ACL and MODIFY ACL _ALL_" on page 128 for more information about user-specific ACL entries.

Controlling SPD Server Resources with PROC SPDO

Overview of PROC SPDO

PROC SPDO is the SAS procedure for the SPD Server operator interface. This procedure runs only on systems where the SAS is installed.

PROC SPDO Command Set

To invoke PROC SPDO, submit the following command:

```
PROC SPDO LIB=libref ;
```

In this command, *libref* is a LIBNAME that was previously allocated to the sasspds engine.

PROC SPDO commands are divided into two classes:

- ACL commands
- LIBNAME proxy commands

The ACL commands are described in "Using the ACL Command Set" on page 126with some simple examples that demonstrate their syntax and usage. For more information about LIBNAME proxy commands, see Chapter 21, "SAS Scalable Performance Data (SPD) Server Operator Interface Procedure (PROC SPDO) ," on page 189.

Using the ACL Command Set

Overview of the ACL Command Set

This section describes PROC SPDO commands that you use to create and maintain ACLs on SPD Server resources.

To issue an ACL-related command, you must first specify an ACL user ID to define the scope of your access. You might also want to set up a scoping member type to access ACLs for resource types other than DATA. Then you can add, modify, list, or delete ACLs within the scope that you set up. You can switch the scope of a user type, a member type, or both at any point in a command sequence. Then you can continue with additional ACL commands in the new scope.

SET ACLTYPE memtype;

sets the member type for subsequent ACL operations. Valid values are DATA, CATALOG, VIEW, and MDDB. The default is DATA.

SET ACLUSER [name];

sets the SPD Server user scope for subsequent ACL operations. The user scope restricts the user's view to only those ACL records that have the specified user name as the owner of the ACL entry. If you omit the *name* value, the default name is the user who assigns the libref.

To perform an ACL operation on a resource entry, one of the following statements must be true:

- the user is the owner of the ACL entry
- the user has Control access to the ACL entry
- ACLSPECIAL=YES is enabled on the user's PROC SPDO LIBNAME connection

Note: You must first issue a SET ACLUSER command before you issue any ADD ACL commands that are described in the following section.

ADD ACL Command

ADD ACL acl1 acl2... [C=cat T=type] [/options]

creates new ACL entries acl1 acl2, and so on. All the ACL entries that you create can be one-part resource names or two-part (table.column) names.

Specify one or more of the following ADD ACL options:

READ
grants universal Read access to the resource.

WRITE
grants universal Write access to the resource.

ALTER
grants universal Alter access to the resource.

GROUPREAD
grants group Read access to the resource.

GROUPWRITE
grants group Write access to the resource.

GROUPALTER
grants group Alter access to the resource.

GENERIC
indicates that the specified ACLs are generic ACLs.

PERSIST
indicates that the specified ACLs are persistent ACLs.

LIBNAME
identifies the special LIBNAME domain resource.

MODEL=*acl-name*
specifies the name of another ACL. This option requests SPD Server to copy all the access permissions and access list entries from this ACL.

C=cat
identifies the specified ACL names as the names of catalog entries in the catalog **cat**. Pair this value with the T= option.

T=*type*
identifies the catalog entry type that is used to qualify the specified ACLs when the C=cat option is specified.

ADD ACL Command Examples

Add a LIBNAME Domain ACL
This ACL grants universal Read and group Write access.

```
add acl/LIBNAME
   read
   groupwrite;
```

Add a Resource ACL
This ACL for the resource MINE_JAN1999 grants universal Read and Write access.

```
add acl mine_jan1999/read write;
```

Add a Generic ACL
This generic ACL for MINE* grants universal Read access.

```
add acl mine/generic read;
```

Add a Column ACL
This ACL for the column MINE_JAN2006.SALARY grants group Read access and denies access to all others.

```
add acl mine_jan2006.salary/groupread;
```

Add a 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;
```

Add a 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;
```

Add a 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;
```

MODIFY ACL and MODIFY ACL _ALL_

MODIFY ACL acl1 acl2... [C=cat T=type] /options User List;

MODIFY ACL _ALL_ /options User List;

modifies existing ACLs for the specified resources (acl1, acl2, and so on). The ACL entries can be one-part resource names or two-part (table.column) names. Specify _ALL_ to modify all existing ACLs for which you have Control access. 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. Specify the characteristics that you want to modify by specifying one or more of the following options, or by specifying *user list*.

Specify one or more of the MODIFY ACL options:

READ
: grants universal Read access.

NOREAD
: removes universal Read access.

WRITE
: grants universal Write access.

NOWRITE
: removes universal Write access.

ALTER
: grants universal Alter access.

NOALTER
: removes universal Alter access.

GROUPREAD
: grants group Read access.

NOGROUPREAD
: removes group Read access.

GROUPWRITE
: grants group Write access.

NOGROUPWRITE
: removes group Write access.

GROUPALTER
: grants group Alter access.

NOGROUPALTER
: removes group Alter access.

GENERIC
: indicates that the specified ACLs are generic ACLs.

LIBNAME
: identifies the special LIBNAME domain ACL.

C=cat
: identifies the specified ACLs as names of catalog entries from the catalog **cat**. Pair this value with the T= option.

T=*type*
: identifies the catalog entry type that is used to qualify the specified ACLs when the C=cat option is specified.

User List
a list of the following form:*user name* = (Y/N,Y/N,Y/N,Y/N) where each comma-delimited Y or N represents, in order, user settings for Read, Write, Alter, and Control privileges.

MODIFY ACL Command Examples

Modify a LIBNAME Domain ACL

This command modifies a LIBNAME domain to set Read and Write access for a given user.

```
modify acl/LIBNAME
  ralph=(y,y,n,n);
```

Modify the ACL MINE

This command modifies the ACL MINE_JAN2003 to deny universal Write access and to add user-specific permissions.

```
modify acl mine_jan2003/nowrite
  bolick=(y,n,n,n)
  johndoe=(n,n,n,n);
```

Modify a Generic ACL

This command modifies the generic ACL MINE* to add user-specific permissions.

```
modify acl mine/generic
  tom=(y,y,y,n);
```

Modify All ACLs

This command modifies all ACLs to grant Read access to a given user.

```
modify acl _all_/gene=(y,,,);
```

Modify a Column ACL

This command modifies the column ACL MINE_JAN2006.SALARY to add explicit Read and Write access for a given user.

```
modify acl mine_jan2006.salary/ralph=(y,y,n,n);
```

Modify a Generic Column ACL

This command modifies the 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);
```

Modify an ACL for a Catalog

This command modifies the catalog MYCAT to remove universal Read and group Write access.

```
set acltype catalog;
  modify acl mycat/noread nogroupwrite;
```

Modify a Generic ACL for Catalog Entries

This command modifies the 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;
```

*LIST ACL and LIST ACL _ALL_*

LIST ACL acl1 acl2... [/options];

LIST ACL _ALL_ [/options];

lists information about the specified ACLs (acl1, acl2, and so on). The ACL entries can be one-part resource names or two-part (*table.column*) names. Specify _ALL_ to list all existing resource ACLs for which you have Control access. 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.

Specify one or more of the LIST ACL options:

GENERIC
: indicates that the specified ACLs are generic ACLs.

LIBNAME
: identifies the special LIBNAME domain ACL.

C=cat
: identifies the specified ACLs as names of catalog entries from the catalog **cat**. Pair this value with the T= option.

T=*type*
: identifies the catalog entry type that is used to qualify the selected ACLs when the C=cat option is specified.

VERBOSE
: performs the requested table ACL listing, followed by the column ACLs for one or more specified tables. This option is equivalent to specifying a LIST ACL table command followed by a LIST ACL table._ALL_ command.

LIST ACL Command Examples

List All ACL Entries

This command lists all ACL entries for the current ACL type setting.

```
list acl _all_;
```

List a Generic ACL

This command lists a generic ACL entry for MINE*.

```
list acl mine/generic;
```

List All Column ACLS for a Table

This command lists all column ACLs for table MINE_JAN2003.

```
list acl mine_jan2003._all_;
```

List All Column ACLs for All Tables

This command lists all column ACLs for all tables.

```
list acl _all._all_;
```

List a Specific Column

This command lists the column ACL for MINE_JAN2006.SALARY.

```
list acl
 mine_jan2006.salary;
```

List All ACL Data for a Table
This command provides all ACL information for table MINE_JAN2006.

```
list acl
 mine_jan2006/verbose;
```

List All ACLs for Catalogs
This command lists all ACLs for the ACL type **catalog**.

```
set acltype catalog;
 list acl _all_;
```

List All ACLs for a Catalog
This command lists all ACLs for catalog MYCAT.?.CATAMS.

```
set acltype catalog;

  list acl _all_ c=mycat t=catams;
```

DELETE ACL and DELETE ACL _ALL_

DELETE ACL acl1 acl2... [C=cat T=type] /options

DELETE _ALL_[C=cat T=type] /options;

deletes existing ACLs for the specified resources (acl1 acl2, and so on). The ACL entries can be one-part resource names or two-part *table.column* names. Specify _ALL_ to delete all existing resource ACLs for which you have Control access. 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.

Specify one or more of the DELETE ACL options:

GENERIC
indicates that the specified ACLs are generic ACLs.

LIBNAME
identifies the special LIBNAME domain ACL.

C=cat
identifies the specified ACLs as names of catalog entries from the catalog **cat**. Pair this value with the T= option.

T=*type*
identifies the catalog entry type that is used to qualify the specified ACLs when the C=cat option is specified.

DELETE ACL Examples

Delete a LIBNAME ACL
This command deletes a LIBNAME ACL.

```
delete acl/LIBNAME;
```

Delete All ACLs for the Current ACL Type
This command deletes all the ACLs for the current ACL type.

```
delete acl _all_;
```

Delete a Resource ACL
This command deletes the ACL MINE_JAN2003.

```
delete acl mine_jan2003;
```

Delete a Generic ACL
This command deletes the generic ACL MINE*.

```
delete acl mine/generic;
```

Delete a Column ACL
This command deletes a column ACL on MINE_JAN2003.SALARY.

```
delete acl mine_jan2003.salary;
```

Delete All Column ACLs on a Table
This command deletes all column ACLs on the table KBIKE.

```
delete acl kbike._all_;
```

Delete All Column ACLs on All Tables
This command deletes all column ACLs on all tables.

```
delete acl _all_._all_;
```

Delete a Catalog ACL
This command deletes an ACL on the catalog RBIKE.

```
set acltype catalog;
 delete acl rbike;
```

Delete a Generic ACL on Catalog Entries
This command deletes a generic ACL on the catalog entries MYCAT.MY*.CATAMS.

```
set acltype catalog;
delete acl my
 c=mycat
 t=catams/generic;
```

ACL Security Examples

Overview of Security Examples

Precedence of permission checks includes inheriting the permissions of the LIBNAME ACL for resources owned by the domain owner. The LIBNAME ACL is used first to grant LIBNAME access to the domain, and then to inherit ACLs to resources that belong *to the domain owner*. When a user attempts to access resources in a domain in which the domain owner specifies LIBACLINHERIT=YES, the following ACL precedence of permissions checks are made on the resource:

1. If user-specific ACLs are defined on the object for the user, the user gets these permissions.
2. If group-specific ACLs are defined on the object for the user's group, the user gets these permissions.
3. If LIBNAME ACL permissions are defined for the user and the resource belongs to the OWNER= of the domain, then the user gets the domain LIBNAME ACL permissions on the object.

4. If LIBNAME ACL permissions are defined for the user's group and the user is a member of the OWNER= group of the domain, then the user gets the LIBNAME ACL group permissions on the object.
5. Otherwise, the user gets UNIVERSAL ACLs on the resource.

The following listing contains the libnames.parm files that are used in the code examples, and a list of users and groups in the password database.

```
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 ;

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

When you specify the libname.parm option OWNER=, no other user can access the domain unless the user is given permissions by the domain owner. Permissions to access a domain are given using a LIBNAME ACL statement.

The following code example uses a LIBNAME ACL statement to give access permissions to different groups.

```
LIBNAME d2 sasspds 'd2'
  server=zztop.5162
  user='prod1'
  password='spds123'
  IP=YES ;

/* Give permissions to LIBNAME */

PROC SPDO library=d2 ;

/* assign who owns the ACLs */

  set acluser prod1 ;

/* Give specific groups access */
/* to the domain. */

  add ACL / LIBNAME ;
  modify ACL /
  LIBNAME prodgrp=(y,y,y,y)
    group1=(y,y,n,n)
    group2=(y,n,n,n)
    group3=(y,n,n,n) ;

/* Give spedific users access to */
/* the domain */

  modify ACL /
  LIBNAME user7=(y,n,n,n)
    admin1=(y,n,n,n) ;
  list ACL _all_ ;
quit ;
```

The ID **prod2** is in the group that has permissions to control the LIBNAME ACL. Any ID in that group can modify the LIBNAME ACL.

Because the ACL was created by user **prod1**, the user **prod2** must use the user ID **prod1** to modify the LIBNAME ACL. This is allowed because the group was given control. User **prod1** still remains the owner of the LIBNAME ACL.

```
LIBNAME prod2d2 sasspds 'd2'
  server=zztop.5162
  user='prod1'
  password='spds123'
  IP=YES ;

PROC SPDO library=prod2d2 ;

/* Set user ID as 'user1', who owns */
/* the ACL to be modified */

  set acluser prod1 ;
  modify ACL /
  LIBNAME group1=(n,n,n,n)
    group4=(y,n,n,n) ;
  list ACL _all_ ;
```

```
quit ;
```

A user who has ACL special privileges can also change the LIBNAME ACL s. In the following example , the user **admin1** uses the ACLSPECIAL= statement to modify the LIBNAME ACL. As in the previous example, the user **admin1** must use the user ID **prod1**.

```
LIBNAME admin1d2 sasspds 'd2'
  server=zztop.5162
  user='admin1'
  password='spds123'
  ACLSPECIAL=YES
  IP=YES ;

PROC SPDO library=admin1d2 ;

/* The ACLSPECIAL= statement allows */
/* the user 'admin1' to operate under */
/* the user ID 'prod1', allowing the */
/* ACLs to be modified. */

  set acluser prod1 ;
  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 file, the ACL precedence of permission checks changes for resources that belong to the domain owner. Turning on LIBACLINHERIT creates a LIBNAME ACL on the specified LIBNAME domain. For information about LIBACLINHERIT permissions and how they affect ACL precedence of permission checks, see “Controlling the Precedence of Permission Checks with the LIBACLINHERIT= Option and the OWNER= Option” on page 99.

The following example uses LIBACLINHERIT:

```
/* information from libnames.parm                 */
/*                                                */
/* LIBNAME=LIBINHER                               */
/*    pathname=/IDX1/spdsmgr/spds41test/libinher  */
/*    LIBACLINHERIT=YES                           */
/*    owner=admin;                                */
/* LIBNAME=noinher                                */
/*    pathname=/IDX1/spdsmgr/spds41test/noinher   */
/*    owner=admin;                                */


LIBNAME libinher sasspds 'libinher'
   server=zztop.5129
   user='admin'
   password='spds123';

LIBNAME noinher sasspds 'noinher'
   server=zztop.5129
```

```
   user='admin'
   password='spds123';

data libinher.admins_table
     noinher.admins_table ;

   do i = 1 to 10;
     output;
   end;
run;


/* Set up LIBNAME access for user anonymous */

PROC SPDO library=libinher;


/* set who will own these ACLs */

set acluser admin;


/* Add a LIBNAME ACL to d1 */

add acl / LIBNAME;


/* Modify LIBNAME ACL Domain d1     */
/* Allow users in Group 1 read-only */
/* access to the domain             */
modify acl / LIBNAME read;

  list acl _all_;
quit;


/* Set up LIBNAME access for user anonymous */

PROC SPDO library=noinher;


/* Specify who owns these ACLs */

set acluser admin ;


/* add a LIBNAME ACL to d1 */

add acl / LIBNAME ;


/* Modify LIBNAME ACL Domain d1      */
/* Allow users in Group 1 read-only  */
/* access to the domain              */

modify acl / LIBNAME read ;
```

```
  list acl _all_;
quit;

LIBNAME a_inher sasspds 'libinher'
   server=zztop.5129
   user='anonymous';
LIBNAME a_noher sasspds 'noinher'
   server=zztop.5129
   user='anonymous';

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

The 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. The anonymous ID cannot be removed from the password database using the psmgr utility and the **delete** command. If you want to prevent anonymous user ID access, use the psmgr utility to add a user called **anonymou**s to the password database and keep the password secret.

Any table that is created by the anonymous user ID can be viewed by all users who have access to that table's domain. The anonymous ID can place ACLs on the 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 user ID anonymous */
/* can have ACLs                       */

    PROC SPDO library=john ;

/* assign who owns the ACL */

    set acluser anonymous ;

/* The MODIFY statement sets an ACl so */
/* only user ID '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 ID '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 ;

    LIBNAME 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 ;
```

Read-Only Tables Examples

A common security measure in SPD Server assigns an SPD Server ID to act as the owner of a domain and to provide control over it.

Typically, one or two user IDs 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 ID for data administration spreads responsibility and still 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 LIBNAME 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 LIBNAME 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 owna the ACLs */

      set acluser admin1 ;
```

The two ACL commands in the following code give Read privileges to members of the ACL group ADMIN1 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 SPD 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 access to different groups of users access to the domain owned by the user ID 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 ID in admingrp is grantedRead/Write/Alter access to the domain.
- Any user ID in group1 or group2 is granted Read access to the domain.
- Any user ID in group3 or group4 is granted Read / 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 LIBNAME 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 LIBNAME for groupread     */
/* and groupwrite.  The ACL MUST    */
/* inlcude groupread if other       */
/* users  in the same group as      */
/* admin2 need to be able to read   */
/* tables that were created by      */
/* admin2.                          */

      add ACL admin1_table1 /
        generic
```

```
        read
        groupread
        groupalter ;

      list ACL _all_;
      run;

/* 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 LIBNAME 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 acdmin2.  The */
/* groupread and groupalter allow     */
/* access by users within admingrp.   */
```

```
     PROC SPDO library=admin2d1 ;

/* Assign who owns the ACLs */

     set acluser admin2 ;

/* Modify LIBNAME 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 LIBNAME 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 user IDs, except the ID 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 will 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 special 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 the user ID prod1 to perform table refreshes. Because user ID prod1 is part of the group prodgrp, that ID loses access to the table when the permissions are changed. Prod1, the domain and table owner, can still modify ACLs to gain access.

   ```
   modify ACL prod1_table /
   prod1=(y,y,y,y) ;
   list ACL _all_;
   quit;
   ```

   Now user ID 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.

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 ;
```

Alternatively, you can revoke Read access at the LIBNAME or 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 special operator commands to identify the user and disconnect the process so that the refresh can occur.

```
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 user IDs 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 ID 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 ;
```

ACL Special Users Example

SPD Server user IDs are divided into two levels, 0 through 3 and 4 through 7. Level 4 through level 7 user IDs 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.

Before you give a user SPD Server super user status, consider the following questions:

- Do you trust this user? SPD Server super users can access any data in any domain.
- How many SPD Server super users do you want? Limit the number in order to maintain Control access.
- Is this user knowledgeable about the data and the database users' needs?

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 ID '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 / group2=(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

Symbolic Substitution

SAS Scalable Performance Data (SPD) Server SQL 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 ACL Special privileges, 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 SPD Server user and SPD Server 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 16

Managing SAS Scalable Performance Data (SPD) Server Passwords and Users

Introduction

SPD Server user data is maintained in an internal SPD Server password manager database . Each record in the password manager database describes the specific attributes and capabilities that are associated with an individual user. SPD Server uses two types of user authentication: LDAP (Lightweight Directory Access Protocol) and traditional authentication (which SPD Server performs internally). LDAP authentication is performed by an LDAP server that runs on the SPD Server host machine. For more information about using LDAP with SPD Server, see “LDAP Authentication” on page 167.

The Password Manager Utility psmgr

Overview of the psmgr Utility

The **psmgr** utility manages the password manager database that enables access to the SPD Server host. When you start SPD Server, the command line option -ACLDIR specifies the location (directory path) of the password manager database. The owner of the password manager database, typically the SPD Server administrator, can update the database.

The password manager 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 8 characters and does not need to correspond to any system user ID. A password is also restricted to 8 characters. All alphanumeric characters and the underscore symbol are acceptable for use in user IDs and passwords.

A password for the **psmgr** table must contain a minimum of 6 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 the SPD Server host, that user ID is no longer enabled. That user cannot connect to the SPD Server host until an administrator resets the user ID.

If you are upgrading to SPD Server 5.1 from SPD Server 3.*x*, you must repopulate the SPD Server 5.1 **psmgr** utility from the SPD Server 3.*x* password table.

Invoking the psmgr Utility

You invoke the **psmgr** utility by entering the **psmgr** command and specifying the directory path where the password manager database is located. (Or you can specify a path for a password table that has not yet been created..)

Use the following command:

```
psmgr <full-path-specification-to-password-table>
```

This command invokes the **psmgr** utility and specifies the directory path for the password manager database.

Migrating a psmgr Database from a Previous SPD Server Installation

You can use the psmgr EXPORT and IMPORT commands to migrate psmgr data from a previous installation.

To convert your SPD Server password file to SPD Server 4.*x* or SPD Server 5.1 format from SPD Server 3.*x* format, do the following:

1. Start the SPD Server **psmgr** utility using your previous SPD Server installation.
2. Export your SPD Server password table to a file.
3. Start your new installation of the SPD Server **psmgr** utility with a new password table.
4. In your new installation of SPD Server, import the file from the previous installation into the new password table.

Example:

The following example creates a **psmgr** table from an old format **psmgr** table that exists at /installdir3_0/site.

```
/Installdir3_0/bin/psmgr /Installdir3_0/site

Enter Command > export /Installdir3_0/site/oldtable
Enter Command > quit

/Installdir4_0/bin/psmgr /Installdir4_0/site

Enter Command > import /Installdir3_0/site/oldtable
```

psmgr Commands

The **psmgr** utility is an interactive program. It reads commands and operands from your computer, and prompts you for input when necessary. You can also send a file of commands to the utility and structure each command so that no input is required.

The commands and operands are positional, and they must be separated by blank spaces. If you specify an insufficient number of operands, the utility prompts you for the remaining operands. Password operands, which are obtained with a prompt, are not echoed back to the computer.

You can enter default values for a command in two ways. If you are entering a single operand, a carriage return in place of the operand value causes SPD Server to populate the operand with its default value. If you are entering multiple operands, entering a hyphen or minus "–" symbol for each operand value causes SPD Server to populate each operand with its default value.

psmgr Commands

ADD

adds a new user to the password manager database.

Syntax

```
add username passwd passwd privilege
    [ip_addr|-] [expiretime|-] [group|-]
    [timeout|-] [failures|-] [class|-]
```

Arguments

user name
: the user ID of an SPD Server user. The user ID is restricted to 8 characters. All characters must be alphanumeric or underscores. The SPD Server user ID does not have to correspond to any system user ID.

passwd
: the user's password, which is restricted to 8 characters. The **psmgr** table requires a password with a minimum of 6 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 normal, non-privileged user.

  The numbers 4–7 are equivalent. Use the numbers 4–7 to specify a special user. Special users can update the password manager database and override any ACL restrictions on SPD Server tables. You should grant special privileges to the SPD Server user ID and password for yourself only.

ip_addr
: a numerical IP address. A hyphen (-) indicates that no IP address is specified. This argument restricts the user's access to SPD Server to the specified IP address.

  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 specified, the group definition must have been created by a previous GROUPDEF command. You can change group affiliation by using a GROUPMEM command.

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 can be re-enabled by the **psmgr** administrator using the reset command.

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. The SPD Server server can be configured to provide different server parameters, based on the user's performance class setting.

AUTHORIZE

authorizes a user to modify the password manager database.

Syntax

```
authorize username userspasswd
```

Arguments

username
the user ID of an SPD Server user.

userspasswd
a valid user's password.

Description

Only a special user can update the password manager database. You must be a special user or the owner of a password manager database to use modification commands such as ADD and DELETE. If you are not the owner of the password manager database, you can use the AUTHORIZE command to authorize yourself to update the password manager database. Enter your user ID and password in the password manager database, and then mark the user ID as special (by specifying the authorization level as 4 or higher).

For example, assume that you used the **psmgr** LIST command to obtain the following output:

```
  USER   AUTHORIZATION  IP ADDRESS
-------- ------------- ------------
bar            7
foo            1        192.149.173.5
```

You can grant yourself privileges by issuing the AUTHORIZE command and specifying **bar** as the user name and barpwd1 as the bar password.

Example

```
authorize bar barpwd1
```

CHGAUTH

changes the authorization level for a user.

Syntax

```
chgauth username authlevel
```

Arguments

username
the user ID of an SPD Server user.

authlevel
an authorization level for the user, in the range 1–7. The authorization level number assigns access privileges to the user.

The numbers 0–3 are equivalent. Use the numbers 0–3 to specify a normal, non-privileged user.

The numbers 4–7 are equivalent. Use the numbers 4–7 to specify a special user. Special users can update the password manager database and override any ACL restrictions on SPD Server tables. You should grant special privileges to the SPD Server user ID and password for yourself only.

CHGEXPIRE

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
: the user ID of an SPD Server user. The user ID is restricted to 8 characters. All characters must be alphanumeric or underscores. The SPD Server user ID does not have to correspond to any system user ID.

exptime
: 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.

CHGIP

changes the IP address from which the user must connect to the 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 manager database.

Syntax

```
chgip username "New IP Address"
```

Arguments

username
: the name (user ID) of an SPD Server user. This name must already exist in the password manager database.

ip_addr
: 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

changes the logon time-out date for a user's password.

Syntax

```
chgtimeout username timeoutperiod
```

Arguments

username
: the user ID of an SPD Server user.

timeoutperiod
: a password logon time-out period, specified in days. The time-out period requires the user to successfully log on before the specified number of days has expired. The value is the number of days from the last successful logon that the password is valid.

CHGPASS

changes the password for a user to a permanent password.

Syntax

```
chgpass username oldpwd newpwd
```

Arguments

username
: the user ID of an SPD Server user.

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 6 characters, with at least one numeric character and with at least one alphabetic character. The password cannot contain the user ID.

CHGPERFCLASS

changes the performance class of a user.

Syntax

```
chgperfclass class
```

Arguments

username
: the user ID of an SPD Server user.

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

deletes a user ID.

Syntax

```
delete username !
```

Arguments

username
: the user ID of an SPD Server user.

!
: verifies that you intend to delete the user ID from the password manager database. If you do not specify !, you are prompted to verify the deletion.

EXPORT

exports the current password manager 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 manager database.

Description

The EXPORT command generates a single line in the flat file for each record in the password manager database. User passwords are encrypted in the table.

The contents of the flat file is a representation of what is stored in the password manager 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 manager database.

GROUPDEF

defines a new ACL group entry.

Syntax

```
groupdef groupname
```

Arguments

groupname
: the name of a group. The name must be unique and is restricted to 8 characters. All characters must be alphanumeric or underscores. This argument verifies that the groups that are specified on the GROUPMEM command are valid.

GROUPDEL

deletes an ACL group entry.

Syntax

```
groupdel groupname !
```

Arguments

groupname
: the name of a group.

!
: verifies that you intend to delete the group from the password manager database. If you do not specify **!**, you are prompted to verify the deletion.

GROUPMEM

updates the ACL group list for a user ID. You can specify up to 32 groups for a user.

Syntax

```
groupmem username groupname [groupname|""] [groupname|""] !
```

Arguments

username
the user ID of an SPD Server user.

groupname
the name of an ACL group. The name must be unique and is restricted to 8 characters. Separate each ACL group name with a space. The first ACL group name that you specify becomes the default ACL group for the user. You can specify up to 32 groups.

Note: If you specify fewer than 32 ACL groups, the utility prompts you for additional ACL groups (up to 32). Enter an exclamation point (!) to indicate to SPD Server that there are no more groups in your groupname declaration.

Note: If you use the **groupmem** command in batch mode, the syntax enables you to submit 32 groupname arguments. If you want to update the user ID with less than 32 ACL group members, use an exclamation point (!) to indicate to SPD Server that there are no more groups in your groupname declaration.

GROUPS

lists the all the ACL groups that are in the password manager database.

Syntax

```
groups
```

HELP

displays general or command-specific help for the **psmgr** utility.

Syntax

```
help [command]
```

Arguments

command
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

imports user information from a flat file to the password manager database. The flat file was 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 manager database.

Description

The IMPORT command reads the flat file, interpreting each single line as a record in the password manager database. Typically, the flat file is output from a submitted EXPORT command that was issued on the same password manager database or another password manager database.

If the **psmgr** utility encounters an identical user name (SPD Server user ID) in the password manager database during the import process, it skips the line. The **psmgr** utility displays a message that states that the line was skipped.

LIST

lists the contents of the password manager database or a specific user.

Syntax

```
list [username]
```

Arguments

username
: the user ID of an SPD Server user. If you do not specify a user ID, the entire password manager database is listed.

Example

```
list bar
```

This example might produce the following listing:

```
USER AUTHORIZATION IP ADDRESS
---- ------------- -----------
 bar       7
```

RESET

resets a password for a user to a new temporary, one-time password. 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, using either the NEWPASSWD= or the CHANGEPASSWD= LIBNAME option.

Syntax

```
reset username newpwd newpwd
```

Arguments

username
: the name (user ID) of an SPD Server user.

newpwd
: a new password for the user. The new password can be up to 8 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.

Example

```
reset tom abc123 abc123
```

This example resets the password for **tom .**

QUIT

ends the session and exits from **psmgr**.

Syntax

```
quit
```

Using a File as Input to psmgr

You can create and then send a file of commands to the **psmgr** utility.

Here is a command file named pscmds:

```
groupdef group1
add newuser newpwd1 newpwd1 0 - - group1 - - -
list
quit
```

The command file contains the group group1 and puts newuser in group1.

To run the **psmgr** utility using the command file named pscmds as input, use the appropriate syntax.

For UNIX:

```
psmgr /usr/local/SPDS/site < pscmds
```

For Windows:

```
psmgr d:\spds\site < pscmds
```

SAS Management Console

Overview of SAS Management Console

SPD Server supports SAS Management Console. SAS Management Console is an application that a privileged user can use to manage passwords and ACLs.

For more information about using SPD Server with SAS Management Console, see Chapter 7, “Administering and Configuring SAS Scalable Performance Data (SPD) Server Using the SAS Management Console ,” on page 45.

LDAP Authentication

Overview of LDAP Authentication

SPD Server user passwords can be authenticated with LDAP or the **psmgr** utility. An LDAP server that runs on the SPD Server machine performs LDAP authentication. When you use LDAP authentication, the operating system handles password maintenance. LDAP authentication has the added benefit of operating-system-level security and convenience.

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

- SPD Server users can be authenticated by an LDAP server or by the **psmgr** utility, but not by both. The type of authentication to be performed is specified in the server.parm file, which is read when SPD Server is invoked.

- If you are changing from using the LDAP server to using the **psmgr** utility for authentication, you must remove all LDAP parameters from the SPD Server server.parm file. In order for the changes to the server.parm file to be read, you must restart SPD Server.
- When you configure SPD Server to perform user authentication using the LDAP server, you still need the **psmgr** utility. When you use the LDAP server, a password database record is required for each SPD Server user. SPD Server uses the **psmgr** utility's password database to perform user access control tasks and other tasks that are not related to user password authentication.
- Users that connect to an SPD Server must have corresponding logon information about 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. A user ID must be at least 8 characters in length.
- You must enter the initial password in the psmgr table when you are adding a new user. This password is never used, and simply enables you to add the new user. 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 you create a user connection that uses the NEWPASSWORD= LIBNAME option, the user password is not changed. If you want to change a user password, follow the operating system procedures to change a user password, and check with your LDAP server administrator to ensure that the LDAP database records the password changes.

Configuring LDAP Authentication

To set up LDAP authentication, add the following parameters to the SPD Server's spdsserv.parm file:

(NO)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
: specifies a valid IP address or 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
: specifies 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
: the distinguished name (DN), or the location in the LDAP Server database where the client information is stored. LDAPBINDDN is a Lightweight Directory Access Protocol (LDAP) term. LDAPBINDDN is the combination of the user name 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 spdsserv.parm 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 spdsserv.parm file. To use the default value for any LDAP parameter, omit the parameter specification from the spdsserv.parm file. Undeclared parameters automatically assume default values.

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

Enabling 32 Groups for Selected 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 supports SPD Server user membership in up to 32 SPD Server 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 5.1 password data sets.

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

Default SPD Server user account configurations for SPD Server 5.1 have a default limitation of five SPD Server group memberships per user. This ceiling does not affect most SPD Server users. When power SPD Server users to belong to more than five groups, you can raise the ceiling raised to a maximum of 32 memberships to SPD Server user groups.

To raise the maximum to 32 groups per user, you must change the user's password table data sets. Users that need the higher group membership must use the SPD Server 5.*x* formatted password table data set.

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

Note: SPD Server 5.*x* password table data sets are not backward compatible with earlier versions of SPD Server.

To convert an SPD Server 4.*x* password Table to an SPD Server 5.*x* password table:

1. Export existing password table data sets to EXPORT-FILE.
2. Back up old password table data sets.
3. Create a new target directory for new password table data sets.
4. Run the **psmgr** utility and make the new table an SPD Server Version 5 table by including the **`-ver 5`** switch in the command:

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

5. Import the old passwords from the export file that you created in Step 1.
6. Quit the **psmgr** utility.
7. Verify that the SPD Server start-up scripts point to the newly created password table data sets.

SPD Server 5.1 psmgr GROUPMEM Updates

SPD Server 5.1 enhancements include changes to the **psmgr** utility that accommodate greater flexibility in group membership and password management. These updates required the following minor changes in the GROUPMEM command.

- In SPD Server 4.*x* and earlier, if the user specifies fewer than five groups by using the GROUPMEM command, SPD Server prompts the user to supply additional group names until five group names are provided. In SPD Server 5.*x*, you can 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.

SPD Server 5.1 GROUPMEM Example

The following example uses the SPD Server 5.1 GROUPMEM command to change the groups for user **`dswider`** such 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 !
```

Using SPD Server SAS Management Console Plug-in with SPD Server 5.1 Groups

The SPD Server plug-in for SAS Management Console enables viewing and editing of up to five SPD Server groups, the utility opens an informational window that states this particular user is a member of more than five groups, and that the SPD Server **psmgr** utility should be used to view and edit SPD Server groups for that user.

Chapter 17

DICTIONARY.PWDB and DICTIONARY.ACLS

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

Using SAS Scalable Performance Data (SPD) Server with an Internet Firewall

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.

SPD Server clients communicate with the SPD Server name server via the SPD Server 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 SPD Server name server.

SPD Server clients communicate with the SPD Server host whenever a client needs to complete a LIBNAME connection, or whenever a client needs to issue SPD 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 an SPD Server 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 SPD Server 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 SPD name server listen port in the rc.spds start-up script. The NSPORT parameter in the rc.spds start-up script defines the SPD Server name server listen port. If NSPORT is not defined in the rc.spds start-up script, the SPD 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 19

SAS Scalable Performance Data (SPD) Server Auditing

SAS Scalable Performance Data (SPD) Server Auditing

Overview of SPD Server Auditing

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 an SPD Server resource by an SPD user
- access granted to an SPD Server resource for the SPD 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, see "SPD Server Parameter File Configurations for Auditing" on page 92.

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 SPD Server user that submitted the query
- the group ID of the SPD Server 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 "SPD Server Parameter File Configurations for Auditing" on page 92.

Chapter 20

SAS Scalable Performance Data (SPD) Server Table WHERE Constraints

SAS Scalable Performance Data (SPD) Server Table WHERE Constraints

Overview of Table WHERE Constraints

SPD Server table WHERE constraints enable SPD Server 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 any time the table is accessed by a DATA step or by an SQL query. You can use SPD Server table WHERE constraints with SPD Server symbolic substitutions to create row-level security that filters table rows based on the values for User ID, Group ID, or ACLSPECIAL privileges. SPD Server table WHERE constraints do not effect normal SPD table ACL settings, such as Read, Write, and Column access.

Table constraints are not applied to a table that is opened for Update 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 Constraint

To create a table WHERE constraint, you must be the table owner. Use PROC SPDO commands to define, delete, and describe table WHERE constraints.

To add a table WHERE constraint:

```
constraint add <table-name>;
where <where-clause-expression>;
```

To describe a table WHERE constraint:

```
constraint describe <table-name>;
```

To remove a table WHERE constraint:

```
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 SPD Server symbolic substitution to create row-level security that is based on the values for User ID, GROUP, or ACLSPECIAL privileges. For more information, see "SAS Scalable Performance Data (SPD) Server SQL Symbolic Substitution" on page 153.

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 access 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 SPD Server users exist:

```
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;
```

Part 6

SAS Scalable Performance Data (SPD) Server System Management

Chapter 21

SAS Scalable Performance Data (SPD) Server Operator Interface Procedure (PROC SPDO)

Special SPDO Commands

Overview of SPDO Commands

The SPDO commands described in this section require that ACLSPECIAL= is enabled for your SPDO LIBNAME connection.

To enable ACLSPECIAL=, you must first grant the SPD Server user ID ACLSPECIAL= access rights. Next, the user must request access for a specific connection. To request access, the user adds the ACLSPECIAL=YES option to the LIBNAME statement. The user can now submit SPDO commands:

```
SPDSCMD 'command'
```

This example runs the specified command in the context of the user ID for a spdsserv process. This spdsserv process is associated with the LIBNAME connection that the user used to invoke PROC SPDO. No restrictions are placed on commands that are executed in this manner. Therefore, you must carefully consider which SPD Server users need ACLSPECIAL access rights.

Note: For additional information about using PROC SPDO, see "Controlling SPD Server Resources with PROC SPDO" on page 126.

SPDO Command Examples

List the WORKPATH directory:

```
spdscmd 'ls /spdswork/*.spds';
spdscmd 'dir d:\spdswork\*.spds';
```

Clean up WORKPATH files:

```
spdscmd 'rm /spdswork/*.spds';
spdscmd 'del d:\spdswork\*.spds';
```

LIBNAME Proxy Commands

Overview of Proxy Commands

To issue proxy commands, you must first select the SPD Server user proxy.

LIST USERS;
lists the proxy processes that are accessible to the PROC SPDO lib=<LIBNAME> statement that was dispatched from the SPD Server host. Accessible proxies are anonymous proxies and proxies that are owned by the LIBNAME owner. If the LIBNAME owner has ACLSPECIAL privileges, then all user proxies are listed.

SET USER user-ID [port-number];
enables you to use the port number to distinguish between two proxies that share the same user ID.

LIST USERS/LOCKING;
lists the user-locking proxy threads that are accessible by the PROC SPDO lib=<LIBNAME> statement that was dispatched from the SPD Server host and that were created with the LOCKING=YES LIBNAME option. Accessible proxies are anonymous proxies and proxies that are owned by the LIBNAME owner. If the LIBNAME owner has ACLSPECIAL privileges, then all user-locking proxies are listed. For each user-locking proxy thread, SPD Server returns the SPD Server user ID, the client login, and the thread ID. You can select a user-locking proxy thread from the LIST USERS list by submitting a command in the following form:

SET USER/LOCKING [user-ID threadID=#];
After a user-locking proxy is selected, you can get LIBNAME information by submitting the following commands:

SHOWLIBNAME *libref* | _ALL_;
SHOWLIBNAME *libref* / DATA=[_ALL_| *dsname*];
SHOWLIBNAME *libref* / DUMP=[_ALL_| *dsname*];
The value for *libref* is an explicit SPD Server LIBNAME name. Specify _ALL_ to see every currently assigned LIBNAME for the proxy.

If the /DATA= option is used with _ALL_, information about all of the open tables in the proxy for the given LIBNAME is displayed. If the /DATA= option is used with a data set name *dsname*, detailed information about the specified data set table is displayed.

If the /DUMP= option is used with _ALL_, information about all of the accessible tables in the proxy for the given LIBNAME is displayed. If the /

DATA= option is used with a data set name *dsname*, detailed information about the specified data set table is displayed.

LIBNAME Proxy Command Examples

The following are examples of LIBNAME proxy commands that an administrator might use. They are presented in an order that would be typical of an administrator who is gathering user proxy information before issuing proxy commands on a user's behalf.

1. List all of the users for the server **`sunburn.6100`**:

```
LIBNAME example sasspds
  host='sunburn'
  serv='6100'
  user='sassyl'
  passwd='abc123'
  aclspecial=YES;

PROC SPDO lib=example;
list users;

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 to ANONYMOU and specify process ID (PID) 17610:

```
set user anonymou 17610;

NOTE: User ANONYMOU connected to proxy operator port with pid=17610.
```

3. Show every LIBNAME for user ANONYMOU for this proxy:

```
showlibname _all_;
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. Show all of the open tables in LIBNAME FOO:

```
showlibname FOO/data=_all_;

NOTE: No data sets currently opened for LIBREF FOO.
```

5. Show all of the accessible tables in LIBNAME FOO:

```
showlibname FOO/dump=_all_;

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. The user ANONYMOU issues a WHERE clause on the table BIGX. Show all of the open tables in LIBNAME FOO:

```
showlibname FOO/data=_all_;

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. User ANONYMOU performs a WHERE clause on the table BIGX and displays detailed information about the table BIGX:

```
showlibname FOO/data=bigx;

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
```

8. List all locking users for the server *sunburn.6100*:

```
list users/locking;

Users Currently Connected to the Record Level Proxy
SPDUserName  Client Login   Thread Id
----------------------------------------
ANONYMOU     SASTEST        7
TEST         SASTEST        8
```

9. Set the user to ANONYMOU and specify thread ID 7:

```
set user/locking anonymou threadid 7;
```

```
NOTE: User ANONYMOU connected to record
level proxy operator port with thread=7.
```

10. Show every LIBNAME for locking user ANONYMOU:

```
showlibname _all_;

LIBREF(LOCKING):Pathname assigned=/bigdisk/test/qabig1/
LIBREF(LOCKING):ACL Owner=
LIBREF(LOCKING):ACL Defaults(R,W,A,C)=(Y,Y,Y,Y)
```

11. Show all of the open tables in LIBNAME LOCKING:

```
showlibname LOCKING/data=_all_;

NOTE: No data sets currently opened for LIBREF LOCKING.
```

Privileged OPER Commands

You must have ACLSPECIAL access rights (LIBNAME option ACLSPECIAL=YES) to run privileged OPER commands. To use privileged OPER commands, submit the following command to set yourself as the proxy operator:

```
SET MODE OPER;
```

The SET MODE OPER command sets you as the operator of the user proxy that you are currently set to. A user proxy can have only one operator at any time. If you submit the SET MODE OPER command 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 following commands:

OPER CANCEL [/DUMP];
: cancels and exits the user proxy. If you specify the /DUMP option for a nonlocking user proxy, the proxy exits with an abort() call, which produces a core file. If you are the operator of a locking user proxy, the /DUMP option is ignored. The OPER CANCEL command 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 command to restore the tables to a usable state.

OPER 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 command to restore the tables to a usable state.

  The OPER DISCONNECT command differs from the OPER CANCEL command. When the OPER DISCONNECT command is submitted, the user proxy continues until it detects that the control socket connection has been dropped. As a result, the OPER DISCONNECT command has the potential to complete. However, the point

at which the user proxy detects that the control socket has been disconnected varies, which produces different results.

OPER 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 the OPER CANCEL command or the OPER DISCONNECT command, the OPER INTERRUPT command 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, then the results might be incomplete. However, the user LIBNAME is intact, and the user proxy is still viable. You cannot determine whether a job will be interrupted; it depends on the job that is currently running. To determine whether a job can be interrupted, submit a SHOWLIBNAME libref / DATA=_ALL_ command before you submit the OPER INTERRUPT command to see all of the open tables. You can also submit the SHOWLIBNAME libref/ DATA=_ALL_ command after you submit the OPER INTERRUPT command to see whether all of the open tables were closed. If the tables are still open after you submitted the OPER INTERRUPT command, you can wait and check again later. If the tables need to be closed immediately, you can issue OPER CANCEL to cancel the user proxy.

TRUNCATE Command and Example

The TRUNCATE command is a PROC SPDO command that enables you to delete all of the rows in a table without deleting the table structure or metadata.

```
%let host=kaboom;
%let port=5191;
%let domain=path2;

LIBNAME &domain sasspds "&domain"
 server=&host&port
 user='anonymous'
 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 command 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;
```

The REFRESH Command

Overview of the REFRESH Command

You can use PROC SPDO to dynamically refresh SPD Server parameter and LIBNAME files. If you make changes to your spdsserv.parm file or to your libnames.parm environment file for SPD Server, you can use the REFRESH command to avoid restarting the server. Submitting the REFRESH command refreshes the specified SPD Server file without restarting the server.

When you submit the REFRESH command, SPD Server refreshes the operating parameter file.

Use the following syntax for the REFRESH command:

- To refresh the libnames.parm file:

  ```
  REFRESH DOMAINS
  ```

- To refresh the spdsserv.parm file:

  ```
  REFRESH PARMS
  ```

Each REFRESH operation completely refreshes and replaces the contents of the previous libnames.parm file or spdsserv.parm file in the SPD Server environment.

REFRESH Command Examples

Add a new LIBNAME domain to your current libnames.parm file and use it without restarting the server:

```
LIBNAME spds45 sasspds 'spds45'
        server=estore.5180
        user='admin'
        password='spds123'
        aclspecial=YES
        prompt=YES;

PROC SPDO library=spds44;
```

```
  SET acluser admin;
  REFRESH PARMS;
  REFRESH DOMAINS;
quit;
```

Here is a more detailed version of the example above:

```
/* Domain reftest is a pre-existing domain.  */
/* Add domain reftest2 to libnames.parm and  */
/* specify owner=admin                       */

   LIBNAME=tmp pathname=c:\temp;
   LIBNAME=formats pathname=c:\data\formats;
   LIBNAME=reftest pathname=c:\data\reftest
     owner=admin;
   LIBNAME=reftest2 pathname=c:\data\reftest2
     owner=admin;

/* Run refresh job using admin with ACLSPECIAL */
/* The SPD Server user must have ACLSPECIAL    */
/* privileges to refresh domains.              */

   LIBNAME reftest sasspds 'reftest'
     server=d8488.5180
     user='admin'
     password='spds123'
     aclspecial=YES;

   PROC SPDO library=reftest;
     SET acluser admin;
     REFRESH DOMAINS;
   quit;

/* Domains that have an owner= option such as  */
/* reftest2 (owner=admin) must be reconnected  */
/* to the domain again.                        */

   LIBNAME reftest2 sasspds 'reftest2'
     server=d8488.5180
     user='admin'
     password='spds123';
```

Commands to Nonexistent Users

In SPD Server, you can submit operator commands to a user after you select the user with the SET USER or SET USER/RECORD command. However, user sessions are finite. The user that you select with SET USER or SET USER/RECORD might be unavailable when the user ends the SAS session or disconnects from a LIBNAME and the user proxy. If you submit an OPER command to a user that is no longer in session, or to a user that has ended a locking user proxy, you get the following message:

```
ERROR: Specified locking user no longer exists.
```

If the disconnected user used a nonlocking user proxy, and you submit an OPER command, you get the following message:

```
ERROR: Specified user <userID> with pid <Process-ID> no longer exists.
```

Both of these messages indicate that the user that was selected is no longer valid. In this case, you must use SET USER or SET USER/RECORD to select a different user.

Chapter 22

SAS Scalable Performance Data (SPD) Server Index Utility Ixutil

The Index Utility ixutil

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.

ixutil Usage

Note: Index names are case sensitive. You must specify them exactly as they are listed in the PROC CONTENTS output.

ixutil -crejidx <*data set1,column1*> <*data set2,column2*> ... <*data set_n,column_n*> -libpath <*physical path*> -joinparts <*number of parallel join work units*> ;
Create a join index for a pair of data sets that are in the same domain. The join index can be used by the SPD Server to optimize parallel range joins. The columns **must** already be indexed. The recommended number of parallel join work units is two times the number of processors.

ixutil -deljidx <*data set1,column1*> <*data set2,column2*> ... <*data set_n,column_n*> -libpath <*physical path*> ;
Delete a join index.

ixutil -lstjidx -libpath <*physical path*> [-verbose] ;
List the join indexes that are in a domain.

ixutil -statjidx <*data set1,column1*> <*data set2,column2*> ... <*data set_n,column_n*> -libpath <*physical path*> ;
Gather 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 performance will be. The percentage increases as the distribution of the data for the join column becomes more sorted.

ixutil -stats <*indx1,indx2,...*> -dsn<*data set*> -libpath <*physical path*> [-dist] ;
Print 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. To improve performance and reclaim the wasted disk space, the index should be reorganized using the **ixutil** option **-reorg**.

ixutil -runstats <*indx1,indx2,...*> -dsn <*data set name*> -libpath <*physical path*> [-maxruns <*number*>] ;
Print 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, **ixutil** run statistics display the ten longest runs in the data set. A run is defined as the number of successive observations that contain the same index value. Use the optional [**-maxruns**<number>] argument to change the default setting of 10 runs to any integer in the range 1–100. You can use **ixutil** run statistics to construct more efficient BY and WHERE clause constructs for the data set.

ixutil -reorg <*index1,index2,...*> -dsn <*data set name*> -libpath <*physical path*> ;
Reorganize 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.

ixutil -help
Print the Help menu.

Ixutil Options

-crejidx <*data set1,column1*> < *data set2,column2*> ... <*data set_n,column_n*>
Create a join index for the SPD Server parallel join utility to use.

-deljidx <*data set1,column1*> <*data set2,column2*> ... <*data set_n,column_n*>
Delete a join index.

-dist
Include the distribution statistics with the index statistics.

-lstjidx -libpath <*library path*>
List the join indexes for a domain.

-statjidx <*data set1,column1*> <*data set2,column2*> ... <*data set_n,column_n*>
Get statistics about a join index.

-stats <*index,index...*>
For each specified index, print the index disk usage statistics and value segment list statistics.

-runstats <*index,index...*>
For each specified index, print the run statistics. Runs are defined as successive observations in a table that contain the same index value.

-reorg <*index,index...*>
For each index, reorganize the index to reclaim any unused disk space and coalesce any fragmented value segment lists.

-joinparts <*number of parallel join work units*>
The number of parallel join work units for a join index. Parallel join threads join the work units concurrently and then merge their partial results into the final result.

-dsn <*data set name*>
The SPD Server table that contains the index.

-libpath <*physical path*>
The physical path of the domain that contains the table.

-help
Print the **ixutil** help menu.

Ixutil Examples

Create an Index

Assume that an SPD Server domain named **my domain** is assigned to the directory **/spds** on a machine named **spot**. A user has created a table with the following SAS program:

```
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;
```

This SAS program creates an index for column X of the table named **test**, on the machine named **spot**, in the directory named **/spds**.

Retrieve Disk Usage and Fragmentation Statistics

Use the -stats option of **ixutil** 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 5.1(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

Use the -dist option of **ixutil -stats** to get the index distribution for the index:

```
> ixutil -stats x -dsn test -libpath /spds -dist
SAS Scalable Performance Data Server
   5.1 (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 option 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
   5.1 (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 process results in the best query performance for the index.

Review Disk Usage Statistics

Use the -stats option 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
   5.1 (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 SPD 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 SPD Server 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 option of the SPD Server **ixutil** command to create the join index.

```
> ixutil -crejidx Table1,ID Table2,ID
  -libpath /tmp
  -joinparts 4

SAS Scalable Performance Data Server
   5.1 (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.
```

Generate Join Index Statistics

Obtain statistics on the join index that you created by using the -statjidx option of the **ixutil** command. 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
   5.1 (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 option of the **ixutil** command to delete the join index.

```
> ixutil -deljidx Table1,ID Table2,ID
-libpath /tmp

SAS Scalable Performance Data Server
   5.1 (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 23

SAS Scalable Performance Data (SPD) Server Table List Utility Spdsls

SAS Scalable Performance Data (SPD) Server Table List Utility spdsls

Description

The **spdsls** utility 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 SPD 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 SPD Server tables

Usage

```
spdsls -l [-i] [-o] [-a] [-s] [-v] [-v8] [-v6] [-aonly] <libpath> [Table...]
spdsls -c [-i] [-o] [-a] [-v] <ComponentPath>
spdsls -info [-o] [-v] [-verbose] [-n] [-s] [-v8] [-v6] <libpath> [Table...]
```

spdsls -l
: lists all component files for a specified SPD Server table in the LIBNAME domain. If no table is specified, all tables in the LIBNAME domain are listed. You can use the output list with any system full backup utility.

spdsls -c
: For a specified component file (which is identified by a complete path), lists all other component files for the table that contains the specified file. If you have an SPD

Server table file that is corrupted or that has been deleted, use this option to find all related component files that might be affected.

spdsls -info
lists information about a specified SPD Server table in the LIBNAME domain. This option provides one line of information about the table as a whole rather than a listing for each component of the table.

Options

-a
includes the domain ACL files in the listing. The files contain the access control lists (ACLs) for any SPD Server table in the domain.

-aonly
includes only the domain ACL files in the listing.

-c
For a specified component file (identified with a complete path), lists all other component files for the table. If you have an SPD Server table file that is corrupted or that has been deleted, use this option to find all related component files that might be affected.

-help
prints a list containing the command-line usage and option switches for the **spdsls** utility.

-i
lists the index files.

-l
For a specified SPD Server table in the LIBNAME domain, lists all component files for the table. If there is no table specified, lists all tables in the LIBNAME domain. The output list can be used with any system full backup utility.

-n
lists the number of component files.

-o
lists 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 an SPD 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.

<LibPath>
complete path of an SPD Server LIBNAME directory.

<ComponentPath>
complete path of a specified table component file.

[Table...]
specifies one or more tables to list. (If no table is specified, all tables in the LIBNAME domain are listed.)

Return Values

When **spdsls** exits, it generates a return value. If the return value is 0, the utility was successful. If the return value is 1, the utility completed normally.

spdsls Examples

Use spdsls to List All Components in a Domain

You can use the **-l** option of the **spdsls** command to display index, 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 4.5(TS M0) Build(Feb 26 2009, 11:51:36)
- Domain List Utility
Copyright (c) 1996-2008 by SAS Institute Inc, Cary NC 27513  USA

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

You can use the **-info** option of the **spdsls** command 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 4.50(TS M0)
Build(Feb 26 2009, 11:51:36) - Domain List Utility
Copyright (c) 1996-2008 by SAS Institute Inc, Cary NC 27513  USA

 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
```

Chapter 24

SAS Scalable Performance Data (SPD) Server Backup and Restore Utilities

Overview of the SPD Server Backup and Restore Utilities

The SPD Server backup and restore utilities provide you with the following advantages:

- You can back up an SPD Server table that remains open for query access.
- Each utility has a detailed Help menu.
- An enhanced user interface and user messages are available.

The standard file system backup and restore facilities that native operating systems provide are generally inadequate for backing up and restoring SPD Server tables. SPD Server tables can be enormous in size, surpassing the file size limits maintained by some operating environments. SPD Server is also dependent on the operation 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: are incremental backups of large tables worth the resources that they consume?

The SPD Server backup and restore utilities alleviate these problems because they perform their functions incrementally. Instead of backing up an entire table, the backup utility backs up only the records 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 only the changed records, SPD Server conserves valuable system resources. This, in turn, encourages more frequent backups. By performing backups more frequently, you minimize the possible loss or corruption of an SPD Server table. The software also gives you the option to perform periodic full backups. You can use the SPD Server full backup and restore capabilities or you can use your system's full backup and restore facilities.

The SPD Server provides the following backup and restore utilities:

- **spdsbkup**

  performs incremental or full backups of SPD Server tables, storing the information in an SPD Server backup data file

- **spdsrstr**

  performs incremental or full restores of SPD Server tables using the SPD Server backup data file that is created by the 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 5.1 backup and restore utilities are not compatible with SPD Server 3.*x* and earlier releases. You cannot restore SPD Server 3.*x* backup files using SPD Server 5.1 utilities. 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 5.1 utilities.

Privileged Access Protection

Running the SPD Server backup and restore utilities is a privileged operation. For a user to have access to 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 performs the SPD Server 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.

To run the backup and restore utilities remotely, use SAS PROC SPDO **spdscmd**. The PROC SPDO **spdscmd** command requires the same access rights as backup and restore: either identical user IDs or ACLSPECIAL=YES user privileges.

Access to the backup and restore utilities is granted to the special user SPDSBKUP. You can use the optional user and password options for the utilities to give a specific privileged user access to the utilities.

The SPD Server Table Backup Utility spdsbkup

Description

The backup utility **spdsbkup** performs a full or incremental backup of an SPD Server table or LIBNAME domain. It also creates a backup file that contains full backups of newly created SPD 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 SPD 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

Performing a Full Backup

When you do a full backup of an SPD Server table, all of the table rows and attributes (indexes, partition size, compression, sorted) are backed up. When you restore a full backup, the table is created with those attributes, and then all the rows are added. Any changes that were made to the table attributes since the last full backup was performed are not restored.

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.

Performing an Incremental Backup

When you perform an incremental backup of an SPD 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 the table at the time of the last full backup (indexes, partition size, compression, sorted) are applied to the restored rows.

Return Values

When **spdsbkup** exits, it generates a return value. If the return value is 0, the utility was successful. If the return value is 1, one or more data sets were not backed up. In that case, examine your SAS log for warning messages. If the return value is 2, a critical error caused the process to terminate early. Examine your SAS log for warning and error messages.

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 SPD Server domain that is being backed up. The client can access the physical path of the domain directly or through a network connection.

Creating the LIBNAME Domain

Before a table is eligible for backup, you must create the SPD Server LIBNAME domain by using the BACKUP=YES option in the parameter file. The following two example LIBNAME entries from a data server's LIBNAME parameter file, libnames.parm, show how domains are processed with and without the BACKUP= option:

Consider the following two entries:

```
LIBNAME=nobackup pathname=/usr/foo/test;

LIBNAME=canbackup pathname=/usr/foo/test BACKUP=YES;
```

The entry for the LIBNAME domain called **`nobackup`** creates tables in the directory **`/usr/foo/test`**, but no BACKUP= option is specified. For this reason, tables that are created through this domain definition are ineligible for backup. In contrast, the entry for the LIBNAME domain **`canbackup`**, which also creates tables in the directory **`/usr/foo/test`**, 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 an SPD 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 an SPD 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.

spdsbkup Usage

```
spdsbkup -d <dom> -f <file> -h <host> [-hash] [-s <serv>] [-u <user>]
          [-fibfact <n>] [-p <passwd>][-t <mm/dd/yy:hh:mm:ss>] [-r <count>]
          [-a | -aonly] [-n] [-q] [-v] [-nv6warn] [-proj <dir>] [Table ...]
spdsbkup -inc -d <dom> -f <file> -h <host> [-hash] [-s <serv>] [-u <user>]
          [-fibfact <n>] [-p <passwd>][-t <mm/dd/yy:hh:mm:ss>] [-r <count>]
          [-a | -aonly] [-q] [-v] [-nv6warn] [-proj <dir>] [Table ...]
spdsbkup -full -d <dom> -f <file> -h <host> [-hash] [-s <serv>][-u <user>]
          [-fibfact <n>] [-p <passwd>] [-r <count>] [-a | -aonly] [-n] [-q]
          [-v] [-nv6warn] [-proj <dir>] [Table ...]
```

spdsbkup
: performs an incremental backup or full backup of SPD Server tables

- 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 -inc
: performs only incremental backups of SPD 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.

  Note: When an incremental backup is restored, only the incremental changes to the rows are applied. Any indexes defined for the table are updated accordingly.

spdsbkup -full
: performs only full backups of SPD Server tables. All of the table observations 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.

  Note: When a full backup is restored, the table is created with those attributes and then all of the rows are added.

spdsbkup Options

-a
: includes the domain ACL files in the backup.

-aonly
includes only the domain ACL files in the backup. No tables are backed up.

-d <*domain*>
the LIBNAME domain of SPD Server.

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
prefix filename for the backup data file. This filename is concatenated with _BK_**`ddmmmyyy_hhmmss`**.0.0.0.spds. 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 SPD Server filename 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 data sets.

-full
performs only full backups of SPD Server tables. All of the table observations and attributes (indexes, definitions, partition size, compression, and sorted) are backed up. After each full table backup, -full resets the last full backup date for the table. See the -n option for dependencies.

-h
the SPD Server host to use for the backup.

-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.

-inc
performs incremental backups of the SPD Server tables.

-n
specifies that index information for SPD 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** 5.1 can back up only SPD Server and SPD Server data sets. If you try to back up earlier versions of SPD Server data sets, a warning message is issued unless you specify the -nv6warn option.

-p
the user password.

-proj <*directory*>
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
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
the port number of the name server. If you do not specify this option, the default value is **spdsname**.

-t
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 only if the table was created after the specified date/time. Otherwise, the utility sets the last full backup date for the table to the specified date/time and performs an incremental backup from the last full system backup date.

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 will not be backed up until a full backup of the table is completed. If **spdsbkup** encounters a table that was created before the specified date/time (that is, the table is in the last full system backup), **spdsbkup** sets the last full backup date for the table to the specified time and performs an incremental backup of the table using the last full system backup date.

You cannot use the -t option with **spdsbkup -full**.

-u
the user name.

-v
includes the full name of the backup file and the backup's table of contents file as part of a **spdsbkup** note.

[Table ...]
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.

Backup Data File

Backup Data File Nomenclature

The **spdsbkup** utility stores backup data in a file named file_BK_**ddmmmyyyy_hhmmss**.0.0.0.spds. 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 filename extension before you can begin a restore session.

Backup Table of Contents File

In addition to the backup file, **spdsbkup** creates a table of contents file using the name file_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 SAS table of contents backup file. 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;
```

spdsbkup User Messages

Three basic types of SPD Server **spdsbkup** backup user 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 observations 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.

The SPD Server Table Restore Utility spdsrstr

spdsrstr Description

The restore utility **spdsrstr** uses a backup file to restore a specified set of SPD Server tables. Tables must meet restore requirements or the **spdsrstr** utility bypasses them. The **spdsrstr** utility can also provide a list of the tables in the backup file that are eligible for restoration.

The restore process is different depending on the type of backup that is being restored:

- When an incremental backup is restored, only the incremental changes to the observations are applied.
- When a full backup is restored, the table is created with the attribute settings that existed when the full backup was performed, and then all of the rows are added.

spdsrstr Requirements

You must meet the following requirements when you use the **spdsrstr** utility to restore a table:

- 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, which assures 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.

spdsrstr Syntax

```
spdsrstr -d <dom> -h <host> {-f <fullfile> | -e <extfile>} [-hash]
 [-r <count>] [-a | -aforce] [-aonly] [-n] [-q] [-s <serv>]
 [-u <user>] [-p <passwrd>] [-proj <dir>] [table ...]

spdsrstr -v -d <dom> -h <host> {-f <fullfile> | -e <extfile>}
 [-s <serv>] [-u <user>] [-p <passwrd>] [-proj <dir>] [table ...]

spdsrstr -t {-f <fullfile> | -e <extfile>} [table...]

spdsrstr -help
```

spdsrstr
: restores all or selected tables from a backup file.

spdsrstr -t
: prints a table of contents for a backup file. This file indicates 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.

spdsrstr Options

-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
the SPD Server LIBNAME domain.

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*>
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.spds. The backup files are restored in order from oldest to newest as determined by the **ddmmmyyyy_hhmmss** component of the filename.

-f <*fullfile*>
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
the host SPD Server to use for the backup.

-hash
prints 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
the user password.

-proj <*dir*>
the domain project directory.

-q
Runs **spdsrstr** in quiet mode, which includes only error and warning messages in the output during a backup operation.

-r
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
the port number of the name server. If you do not specify this option, the default value is **spdsname**.

-u
the user name.

-v
verifies which tables in the backup file can be restored, but do not actually perform the restore operations.

[Table ...]
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.

spdsrstr Return Values

When **spdsrstr** exits, it generates a return value. If the return value is 0, the utility was successful. If the return value is 1, one or more data sets could not be restored. In that case, examine your SAS log for warning information. If the return value is 2, a critical error caused the process to terminate early. Examine your SAS log for warning and error information.

spdsrstr User Messages

Three basic types of SPD Server **spdsrstr** backup user 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 observations 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.

spdsbkup and spdsrstr Usage Examples

Backup and Restore Utility Example Scenario

The backup and restore examples in this scenario use Sunday, February 3, 2008 as the starting date for the backup cycle. The weekly schedule includes the following backup schedule:

- a full backup of the SPD Server domain test every Sunday at 23:30
- an incremental backup of the domain at 23:30 on the remaining days of the week

The scenario includes the following examples:

- Perform Exclusive SPD Server Full and Incremental Backups on page 224
- Perform SPD Server Incremental and Full Backups, and System Full Backups on page 224
- Restore a Single SPD Server Table on page 225
- Restore an SPD Server Domain on page 226

Example 1: Perform Exclusive SPD Server Full and Incremental Backups

You can use the SPD Server backup and restore utilities exclusively to perform full and incremental table backups and restores.

This example outlines the steps that you use 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.

1. On Sunday, February 3, 2008 at 23:30, run the SPD Server backup utility to do a full backup of the domain **test**:

   ```
   spdsbkup -full -a -d test -h host -s serv -f backup
   ```

   The backup creates the backup data file backup_BK_05Feb2006_233000.0.0.0.spds and the backup table of contents file backup_TC_03Feb2008_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 the SPD Server backup facility to perform incremental backups:

   ```
   spdsbkup -a -d test -h host -s serv -f backup
   ```

   This statement performs incremental SPD Server 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.

Example 2: Perform SPD Server Incremental and Full Backups, and System Full Backups

You can use SPD Server utilities to perform incremental backups on data sets that you have previously archived, and to perform full backups on new data sets that have never been backed up. You can also back up your SPD Server data sets by using a system utility from your native operating environment. Which one should you use? The advantage to using system full backups is that 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. For example, 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.

This example outlines the steps that you use to perform a full system backup of the domain **test** by using operating system utilities once a week, and then to use SPD Server to perform a domain back up on the remaining nights.

1. On Sunday, February 3, 2008 at 23:30, run the SPD Server list utility **spdsls -l** to produce a listing of the tables that belong to the domain **test** in preparation for a full backup.

   ```
   spdsls -l -a <physical_path_of_domain>
   ```

2. On Monday, February 4, 2008 at 23:30, run the SPD Server backup utility **spdsbkup** and set 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.

```
spdsbkup -d test -h host -s serv -t 02/04/08:23:30:00 -f backup
```

The utility creates the backup data file backup_BK_04Feb2008_233000.0.0.0.spds and a backup table of contents file backup_TC_04Feb2008_233000.

The backup file contains incremental changes for tables that were modified after 23:30:00 on February 3, 2008, and full backups of tables created after 23:30:00 on February 4, 2008. Only the tables that were modified or created since the last full backup date are included in the backup file. The table of contents file contains information about each table that was either incrementally or fully backed up.

3. Archive the SPD Server backup file and source in the table of contents file into a SAS table of contents table.
4. On Tuesday night through Saturday night, use the SPD Server backup facility to do incremental backups of previously backed up tables and full backups of the newly created tables:

   ```
   spdsbkup -d test -h host -s serv -f backup
   ```

   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.
5. Archive the incremental data file and source in the table of contents file into a SAS table of contents table.

Example 3: 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, 2008.

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, 2008.) If the table was backed up fully by the SPD Server backup utility, skip this step.
2. Run a SAS query on the backup table of contents table bkup.toc.

   ```
   select bk_file from foo.bkup_toc
   where domain = "test"
   and table = "results"
   and dttime >= '03Feb2008:23:30:00'd;
   ```

   The query results indicate which SPD Server backup files are required to restore the table to its last full backup state.
3. Restore 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, 2008. 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 host -s serv -f backup_BK_03Feb2008_233000.0.0.0.spds results

spdsrstr -d test -h host-s serv -f backup_BK_05Feb2008_233000.0.0.0.spds results

spdsrstr -d test -h host -s serv -f backup_BK_07Feb2008_233000.0.0.0.spds results
```

TIP You can also use the -e option of **spdsrstr** and restore all of the files with one command:

```
spdsrstr -d test -h hostname -s serv -e backup results
```

Note: When you restore a single table, you do not need to restore the ACL files, because they were not deleted.

Example 4: Restore an SPD Server Domain

Use the following steps to restore an SPD Server domain named **test**. This domain was lost due to a system media failure that occurred on Friday, February 15, 2008.

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, 2008. 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 >= '10FEB2008:23:30:00'd;
   ```

   The query results identify which SPD Server backup files are required to restore the domain.
3. Restore 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 host -s serv -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.

Use PROC SPDO to Back Up and Restore SPD Server Tables

You can use the SAS **PROC SPDO spdscmd** command to run the SPD Server backup and restore utilities. In order to use this command, you must submit the command using an SPD Server LIBNAME that has special privileges. Backup and restore utilities require privileged access. To grant special privileges, you must specify the LIBNAME option ACLSPECIAL=YES.

When you execute commands using the PROC SPDO **spdscmd** command, the current working directory is the root directory of SPD Server. Messages generated by the commands are echoed to the SAS log. In the next example, the SPD Server 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.

Use the following steps to use PROC SPDO to execute SPD Server backup and restore utilities:

1. Create an SPD Server LIBNAME, and specify special privileges.

```
LIBNAME backup sasspds 'test'
 host='sunny'
 serv='5150'
 user='admin'
 passwd='admin'
 ACLSPECIAL=YES;
```

   This example creates the LIBNAME backup for the domain **test** on the host machine **sunny**. The port number of the name server is 5150, and **admin** is the SPD Server user ID and password.

2. Invoke PROC SPDO for the LIBNAME:

```
PROC SPDO lib=backup;
```

3. Use PROC SPDO remote system commands to issue backup and restore statements on the server. The following example performs a full SPD Server backup of the domain **tstdomn** at 23:30 on February 3, 2008.

```
spdscmd 'spdsbkup -a -full -d tstdomn -h sunny -s 5150 -f /spdsadm/bkup/test';
```

   The example statement creates the backup file **/spdsadm/bkup/test_BK_03Feb2008_233000.0.0.spds** and the table of contents file **/spdsadm/bkup/test_TC_03Feb2008_233000** on the server.

4. If a later restore operation is necessary, run the SPD Server restore utility to restore the domain to its last full backup state:

```
spdscmd 'spdsrstr -aforce -d tstdomn -h sunny -s 5150 -e /spdsadm/bkup/test
```

Back Up and Restore Table Indexes Using SPD Server Full Backups

When you perform an SPD Server full backup of a table, by default the utility saves information to re-create the indexes. This information is subsequently used to re-create the indexes if the table is fully restored.

The SPD Server full backup utility does not save the index data. The utility 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:

- The SPD Server restore utility can re-create the indexes when the table is created. In this method, the index is updated dynamically as each observation is added to the table.
- Use the -n option on the SPD Server restore utility. 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 SPD Server Table Indexes Using System Full Backups

Overview of Backing Up and Restoring SPD Server Table Indexes Using System Full Backups

Restoring indexes from system full backups and restores is not as clean as restoring indexes from SPD Server full backups and restores. To understand why, consider the two methods that you can use to restore indexes from a system full backup:

- **restore the index dynamically as the table is restored**
- **recreate the index after the table is restored**

How do you decide which method to use? You must balance the time and resources that are needed to back up the index against the time needed to re-create the index when the table is restored.

Restoring the Index Dynamically

To restore the index 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 use this method, 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, and then run **spdsrstr** to restore the table. After the table is restored, use PROC DATASETS again to modify the table and create the indexes.

This method trades the resources that you save by not fully backing up the index files against the greater amount of time it can take to re-create the indexes fully if the table must be restored.

Chapter 25

SAS Scalable Performance Data (SPD) Server Directory Cleanup Utility

Overview of the SPD Server Directory Cleanup Utility spdsclean

You use the SPD Server cleanup utility **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 uses a simple command-line interface. You can control the level of cleanup performed and control the behavior of elements that are used in the utility.

CAUTION:
Use the spdsclean command line utility only when SPD Server is not running. Do not run **spdsclean** when the SPD Server host 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, which permits concurrent access to file structures that were not designed to support concurrent access.

Using the Directory Cleanup Utility spdsclean

The **spdsclean** program is a command-line utility. It supports a set of command-line options and parameters that you use to specify the location and names of tables to convert, and to specify behaviors that you want to control during the conversion process.

Here is the command-line syntax:

```
spdsclean <-options>
```

You can specify options in any order. All options are global in scope.

spdsclean Wildcards and Pattern Matching

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 LIBNAME domain name from the LIBNAME 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 LIBNAME domain name in the LIBNAME file.
- The asterisk (*) character terminates the pattern and finds wildcard matches to all remaining characters in the LIBNAME domain name in the LIBNAME file.

For example, the -domains pattern **`?test*`** matches the domains ATEST1, ATEST123, ATESTXYZ, CTEST1, and so on, from a LIBNAME file. The -domains pattern **`test*`** matches only the domain name TEST from the LIBNAME 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. For example:

```
spdsclean -domains "?test*"
```

You can also disable command shell globbing for the execution of the **spdsclean** command.

spdsclean Options

spdsclean options are divided into two classes. The options in the first class define actions. The options in the second class modify behavior.

Options That Define Actions

The **spdsclean** utility uses the following option settings to define specific actions:

-parmfile parmFile
runs cleanup on the specified SPD Server environment, which is defined in the specified SPD Server parameter file. The cleanup action empties all directory resources that are defined in the SPD Server parameter file. All files in the WORKPATH= path list are deleted. Options that modify -parmfile cleanup actions are described in "Options That Modify Behavior" on page 233

-libnamefile libnameFile
runs a cleanup on the SPD Server environment that is specified by libnameFile (the LIBNAME parameter file). The cleanup process empties directory resources that are defined in LIBNAME statements in the LIBNAME parameter file. Cleaning up directory resources removes files and file types that you specify in the **spdsclean** action. The **spdsclean** utility always deletes residual lock files that were left behind in the domain directory. The cleanup process removes any temporary (TEMP=yes) directories and files. Residual temporary files in the allocated domain directories are deleted by default. You can include ACL files and the LIBNAME state file in the files to be deleted, and you can suppress the default deletion of residual temporary files. Use the -domains option with pattern matching to filter the domains that you want to clean in the LIBNAME parameter file. For more information, see the description for the -domains option.

-logdir logPath
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, which tells SPD Server how long to keep the log files using the `-logage` option. When **spdsclean** finds server log files that have a creation date that is older than `-logage` days, **spdsclean** deletes the files. Files that are equal to or less than the specified age are retained. For more information, see the -logage option under "Options That Modify Behavior" on page 233.

Options That Modify Behavior

The **spdsclean** utility uses the following option settings to modify specific behaviors:

-all
equivalent to specifying the following options:

```
-tmp -acl -lib11
```

.

-tmp
enables deletion of residual temporary files in the LIBNAME domain path list. Deletion is enabled by default.

+tmp
disables the deletion of residual temporary files in the LIBNAME domain path list.

-acl
enables the deletion of ACL files in the LIBNAME domain path list. Enabling the -acl setting deletes .spres11* and .sppro11* files from the LIBNAME domains that are specified by the -libnamefile option. The -acl option applies to all of the LIBNAME domains that you specify on the -libnamefile option. Deleting the ACL files does not grant broader access to a given resource. Deleting the ACL files restricts the access to the resource owner. Deletion is enabled by default.

+acl
disables the deletion of ACL files in the LIBNAME domain path list.

-lib11
enables the deletion of the domain state file, .spdslib11. The default setting is +lib11.

+lib11
disables the deletion of the domain state file. This setting is the default setting for the lib111 variable.

-verbose
enables logging of the resource cleanup process from WORKPATH, system workspace directories, and LIBNAME domain directories. Specifying -verbose is equivalent to specifying -vwork and -vdomain.

+verbose
disables logging of the resource cleanup process from WORKPATH, system workspace directories, and LIBNAME domain directories. 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 setting is the default setting.

-vdomain
enables logging of the resource cleanup process from LIBNAME domain directories.

+vdomain
disables logging of the resource cleanup process from LIBNAME domain directories. This setting is the default setting.

-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 LIBNAME domains from the LIBNAME file when it is processed. Standard pattern matching rules and wildcards apply.

-logage ageDays
sets the age threshold, in days, for keeping .spdslog files in the SPD Server log directory when the -logdir option is specified. If the -logdir option is specified and the .spdslog files in the SPD Server log directory are older than the threshold value, the files are deleted. The default value for ageDays is 7.

spdsclean 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 LIBNAME Domain Files

The following **spdsclean** command cleans all of the residual temporary files from all of the LIBNAME domains that are defined in the specified LIBNAME 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 Specific LIBNAME Domains

The following **spdsclean** command cleans all residual temporary files from the LIBNAME 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 LIBNAME file, 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 LIBNAME Domain File Classes

The following **spdsclean** command cleans only the ACL files from LIBNAME domains that begin with UJOE that are defined in the specified LIBNAME file. Because this command specifies the +tmp option, the deletion of residual temporary files is suppressed. To log the LIBNAME 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 LIBNAME file, issue the following command:

```
spdsclean -libnamefile /opt/spds45/site/libnames.parm
 -domains trial9? -lib11 +tmp
```

To log the LIBNAME domains that were cleaned and the files that were deleted, add the -verbose option.

Cleaning WORKPATH and LIBNAME Combinations

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 LIBNAME 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, LIBNAME 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 LIBNAME 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
```

Glossary of spdsclean Terms

ACL files
: When you create SPD Server access control lists (ACLs), hidden ACL files are created in the primary directory of the LIBNAME domain. The hidden files are named .spres11* and .sppro11*. The hidden ACL files retain the state of the ACLs that were defined for the LIBNAME 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 LIBNAME domain. The directory path information is stored as an ordered list for each of the following SPD Server domain storage classes:

  - METAPATH=
  - DATAPATH=
  - INDEXPATH=

  As you make LIBNAME 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.

libnames.parm file
: The libnames.parm file defines the SPD Server LIBNAME domains for the SPD Server environment. The libnames.parm file is a collection of LIBNAME statements.

Each LIBNAME statement defines a storage domain that SPD Server uses with clients. You modify the libnames.parm file by using the -libnamefile option with the **spdsserv** command.

residual lock file
: When SPD Server accesses a data resource or table that is within a LIBNAME 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 LIBNAME proxy process terminates unexpectedly, the residual lock files remain in the LIBNAME 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 LIBNAME domain. If the SPD Server LIBNAME proxy process terminates unexpectedly while you are creating a new file, the residual temporary files remain in the LIBNAME 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.

spdsserv.parm file
: The spdsserv.parm 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. To specify the spdsserv.parm file, use the **spdsserv** command with the -parmfile option.

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).

Chapter 26

SAS Scalable Performance Data (SPD) Server Debugging Tools

Overview of SPD Server Debugging Tools

SPD Server includes debugging tools that are useful for system administrators. The debugging tools allow SPD Server 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 server parameter file options.

SPD Server 5.1 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 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= parameter file debug 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=y;
```

SPD Server 5.1 Server Parameter File Debug Option

SPD Server provides the following server parameter file 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'
```