

SAS[®] Scalable Performance Data Server[®] 4.53 Administrator's Guide



The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2011. *SAS® Scalable Performance Data Server® 4.53: Administrator's Guide*. Cary, NC : SAS Institute Inc.

SAS® Scalable Performance Data Server® 4.53: Administrator's Guide

All rights reserved. Produced in the United States of America.

For a hardcopy book: No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

For a Web download or e-book: Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

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

U.S. Government Restricted Rights Notice: Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227-19 Commercial Computer Software-Restricted Rights (June 1987).

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

1st electronic book, July 2011

SAS® Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at

support.sas.com/publishing or call 1-800-727-3228.

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

Contents

PART 1 Product Notes 1

Chapter 1 • SPD Server 4.5 Product Notes	3
Overview	3
What's New in SPD Server 4.53?	3
What's New in SPD Server 4.52?	3
What's New in SPD Server 4.5?	5

PART 2 Installation 7

Chapter 2 • SPD Server Pre-Installation and System Requirements Guide	9
AIX Requirements and Tuning for 64-Bit SPD Server	9
HP-UX Requirements and Tuning for 64-Bit SPD Server	10
Solaris on SPARC Requirements and Tuning for 64-Bit SPD Server	11
Solaris on X64 Requirements and Tuning for 64-Bit SPD Server	11
Linux on X64 Requirements and Tuning for 64-Bit SPD Server	12
Windows Requirements and Tuning for 32-Bit SPD Server	12
SPD Server 4.5 Client Requirements	12
Chapter 3 • SPD Server UNIX Installation Guide	13
Before You Install: Precautions and Required Permissions	14
Packing List for SPD Server Distribution	15
SAS Scalable Performance Data Server 4.5 and SAS Deployment Wizard	18
Upgrading and Reinstalling SPD Server	18
Configuring SPD Server Host Software for Your Site	19
Configuring the rc.spds File	22
Verify That SPD Server 4.5 Is Running	25
Configure SPD Server Client Software	26
Test Your SPD Server Installation Using SAS	27
SPD Server Command Reference	29
SPD Server 4.5 and the SAS Management Console Utility	33
SPD Server Lightweight Directory Access Protocol (LDAP) Authentication	33
Notes for SPD Server Administrators	34
Troubleshooting	36
Chapter 4 • SPD Server Windows Installation Guide	39
Before You Install: Precautions and Required Permissions	40
Packing List for SPD Server Distribution	40
SAS Scalable Performance Data Server 4.5 and SAS Deployment Wizard	43
Upgrading and Reinstalling SPD Server	44
Validate Default Port and Library Assignments	44
Verify the Name Server Port Assignment and TMP Path	45
Initialize the Password Manager Database	45
Install SPD Server as a Service	46
Start and Stop SPD Server	47
Configure SPD Server Software on Your Windows Host	48
Install and Configure SPD Server Clients	49

Use SAS to Test Your SPD Server Installation	50
SPD Server Command Reference	52
SPD Server and the SAS Management Console	55
SPD Server Lightweight Directory Access Protocol (LDAP) Authentication	55
SPD Server User IDs	56
Troubleshooting	57

PART 3 Migration 61

Chapter 5 • SPD Server 3.x to SPD Server 4.5 Conversion Utility	63
Introduction	63
Before You Convert	63
Overview of the SPDSCONV Utility	64
Using SPDSCONV	65
SPDSCONV Utility Examples	66

PART 4 Configuration 69

Chapter 6 • Using the SPD Server Name Server to Manage Resources	71
Managing Computing Resources with a Name Server	71
Configuring SPD Server on a Corporate Network	71

Chapter 7 • Administering and Configuring SPD Server Using the SAS Management Console	75
Overview of SAS Management Console	75
Accessing SPD Server Services in SAS Management Console	76
Connect to an SPD Server	76
Password Manager	78
ACL Manager	82
Server Manager	85
Process Profiler	88
Proxy Manager	90

Chapter 8 • SPD Server SQL Query Rewrite Facility	93
Overview of the SQL Query Rewrite Facility	93
Configuring Storage Space for the SQL Query Rewrite Facility	93
SQL Query Rewrite Facility Options	94

Chapter 9 • Using SPD Server with Other Clients	97
Introduction	97
Using Open Database Connectivity (ODBC) to Access SPD Server Tables	98
Using JDBC (Java) to Access SPD Server Tables	103
Using htmSQL to Access SPD Server Tables	106
Using the SQL C API to Access SPD Server Tables	107

Chapter 10 • Configuring Disk Storage for SPD Server	109
Introduction	109
SPD Server Component File Types and Sizes	109
Configuring LIBNAME Domain Disk Space	110

Chapter 11 • Setting Up SPD Server Parameter Files	113
---	------------

Introduction	113
Syntax for the -parmfile Option	113
Syntax for the spdsserv.parm Options	114
Server Performance Levels	114
spdsserv.parm Options	115
SPD Server Parameter File Configurations for LDAP	121
SPD Server Parameter File Configurations for Auditing	122
Chapter 12 • Setting Up SPD Server Libname Parameter Files	125
Introduction	125
Domain Naming Syntax for libnames.parm	126
Domain Path Options	126
Consistency in Nomenclature	128
Domain Access Options	129
Controlling the Precedence of Permission Checks with the LIBACLINHERIT= Option	129
Dynamic Locking	132
Organizing Domains for Scalability	133
Domains and Data Spaces	135
Example libname.parm File Configurations	137
Chapter 13 • Setting Up SPD Server Performance Server	143
Introduction	143
Starting SPD Server Performance Server	144
Performance Server Log File	148
 PART 5 Security 149	
Chapter 14 • ACL Security Overview	151
ACL Security Overview	151
SPD Server ACL Security Model	152
Controlling SPD Server Resources with PROC SPDO and ACL Commands	156
Chapter 15 • Symbolic Substitution	187
Symbolic Substitution	187
Chapter 16 • Managing SPD Server Passwords, Users, and Table ACLs	191
Introduction	191
The Password Manager Utility psmgr	191
SAS Management Console	203
LDAP Authentication Notes	203
Chapter 17 • DICTIONARY.PWDB and DICTIONARY.ACLS	205
DICTIONARY.PWDB and DICTIONARY.ACLS	205
Chapter 18 • Using SPD Server with an Internet Firewall	209
Using SPD Server with an Internet Firewall	209
Chapter 19 • SPD Server Auditing	213
SPD Server Auditing	213
Chapter 20 • SPD Server Table WHERE Constraints	217
SPD Server Table WHERE Constraints	217

PART 6 System Management 221

Chapter 21 • SPD Server Operator Interface Procedure (PROC SPDO)	223
Special SPDO Commands	223
LIBNAME Proxy Commands	224
Privileged OPER Commands	227
TRUNCATE Command and Example	228
Refreshing SPD Server Parameter and LIBNAME Files	229
Commands to Nonexistent Users	230
Chapter 22 • SPD Server Index Utility Ixutil	233
The Index Utility Ixutil	233
Ixutil Examples	235
Chapter 23 • SPD Server Table List Utility Spdsls	241
SPD Server Table List Utility Spdsls	241
Chapter 24 • SPD Server Backup and Restore Utilities	245
Introduction	246
Overview of the SAS Scalable Performance Data Server Backup and Restore Utilities	246
Using Utilities with SPD Server	247
Compatibility with Previous Versions	247
Privileged Access Protection	247
Spdsbkup - The SPD Server Table Backup Utility	248
Backup Requirements	249
Backup Usage	250
Backup Options	250
Backup Return Values	252
Backup Data File	253
Backup Table of Contents File	253
Backup User Messages	254
Spdsrstr - The SPD Server Table Restore Utility	255
Using PROC SPDO to Back Up and Restore SPD Server Tables	261
Back Up and Restore Table Indexes Using SPD Server Full Backups	262
Back Up and Restore SPD Server Table Indexes Using System Full Backups	262
Chapter 25 • SPD Server Directory Cleanup Utility	265
Introduction	265
Using the Directory Cleanup Utility Spdsclean	266
Spdsclean Wildcards and Pattern Matching	266
Spdsclean Options	266
Spdsclean Examples	268
Chapter 26 • SPD Server Debugging Tools	273
Introduction	273
SPD Server 4.5 LIBNAME Statement Debug Options	273
SPD Server 4.5 Server Parameter File Debug Options	274

Part 1

Product Notes

Chapter 1

<i>SPD Server 4.5 Product Notes</i>	<i>3</i>
--	-----------------

Chapter 1

SPD Server 4.5 Product Notes

Overview	3
What's New in SPD Server 4.53?	3
What's New in SPD Server 4.52?	3
What's New in SPD Server 4.5?	5

Overview

This document summarizes enhancements and changes in SAS Scalable Performance Data (SPD) Server 4.53, as well as changes and enhancements for SPD Server 4.5. The most recent changes in SPD Server are presented first.

- The SPD Server 4.53 installation includes client modules that are compatible with SAS 9.3.
- SPD Server 4.53 is not compatible with SAS versions earlier than SAS 9.3. See the appropriate SPD Server UNIX or Windows installation guide for more information about SAS software requirements for use with SPD Server 4.53.

What's New in SPD Server 4.53?

The SAS server parameter AUTHDOMAIN= is available in SPD Server 4.53. Use the AUTHDOMAIN= server parameter to allow connection to a server by specifying the name of an authentication domain metadata object. For more information, see “Options to Identify the SPD Server Client” in the *SAS Scalable Performance Data Server: User's Guide* for SPD Server 4.53.

What's New in SPD Server 4.52?

SPD Server 4.52 has a new server parameter named MAXSORTTHREADS=. Use MAXSORTTHREADS= to provide more control over the number of concurrent threads that are used during an SPD Server parallel sort. For more information, see [“MAXSORTTHREADS=” on page 118](#).

The NOMISS= index support parameter is no longer recognized in SPD Server 4.52. If you submit the NOMISS= parameter to SPD Server 4.52, the parameter declaration is ignored.

The SPD Server parameter file is enhanced to include performance-level parameters that you can specify for different classes of users. You can associate each class of user with a specific set of SPD Server parameter settings to regulate SPD Server resources for each class. A user's resource class is defined by the performance class attribute that is specified in the user's SPD Server Password Manager database record. The configurable user resource class attribute levels are LOW, MEDIUM, HIGH, and LOCKING. For more information about how to assign SPD Server users to configurable user resource classes, see [“Server Performance Levels” on page 114](#).

The spds-clean utility now removes temporary directories and files that were created using the TEMP=YES LIBNAME parameter. In the past, orphaned directories and files were not removed if the SPDSBASE process terminated abnormally or was killed. The spds-clean utility checks for temporary directories in LIBNAME domains during the standard -LIBNAMEFILE cleanup. The orphan directory check and cleanup function is integrated into the spds-clean utility. You do not need to specify an argument for spds-clean to remove orphaned TEMP=YES LIBNAME directories and files. For more information, see [“SPD Server Directory Cleanup Utility” on page 265](#).

The 2 GB size limit on SPD Server files on Windows Win32 platforms has been removed. Win32 platforms can now create and access files (.dpf partitions, index files, and sort bins) that are greater than 2 GB in size. SPD Server on Win32 platforms now supports files up to $(2^{63}-1)$ bytes in size, the same file size that is supported by SPD Server on 64-bit UNIX platforms.

SPD Server 4.52 for Windows supports the Windows 2008 R1 operating system running in 32-bit compatibility mode.

Automatic maintenance and cleanup of SPD Server directory .spdslibll files have been implemented. The .spdslibll directory files are cleaned up when SPD Server starts and reads the libnames.parm file, and whenever SPD Server refreshes the libnames.parm file. In this way, the path information that is contained in .spdslibll files remains current. SPD Server creates a .spdslibll file in a domain if a user connects to a domain in which no .spdslibll file exists. The enhanced .spdslibll file management strategy supports changing data partition paths and index file paths for existing LIBNAMEs in SPD Server.

A new SPD Server reset option called SQLHIMEM has been added to SPD Server SQL. Use SQLHIMEM to allocate and deallocate large blocks of memory for SQL queries. For more information about using SQLHIMEM in SPD Server SQL, see [“SQLHIMEM” in the SAS Scalable Performance Data Server 4.52: User's Guide](#).

The following SPD Server messages have been changed from warnings to notes:

- Directory created from LIBNAME assignment. This message is displayed when a TEMP=YES LIBNAME assignment is created.
- Read-only access to LIBNAME domain restricted by *LIBNAM* ACL. This message appears if you try to make a LIBNAME assignment within a read-only domain.
- Parallel WHERE evaluation suppressed due to sort order on table. This message appears when a sorted table is filtered by a WHERE clause.
- WHERE clause requires SAS filtering support which could affect performance because SPD Server could not fully evaluate the expression.

- Because ASYNC operations create indexes in parallel, the status for all defined indexes will be determined at closing time. This message appears when an asynchronous index is being created.

Base SAS software can now read views that are created by SPD Server, and SPD Server can read views created in Base SAS software. That is, Base SAS software can read SQL views that are created in SPD Server by using explicit pass-through SQL. And, SPD Server can read SQL views that are created in Base SAS software by using PROC SQL. Some restrictions exist for how Base SAS software can use some SPD Server SQL views. For more information, see “SPD Server Views” in the *SAS Scalable Performance Data Server 4.52: User's Guide*.

SPD Server 4.52 includes a new security feature called table WHERE constraints. Table WHERE constraints enable the owner of an SPD Server table to use a WHERE clause to filter or control how the table can be accessed by other SPD Server users. Table WHERE constraints can be used with SPD Server symbolic substitution to implement row-level security by filtering table rows based on User ID, Group ID, or ACLSPECIAL attribute settings. For more information, see “[SPD Server Table WHERE Constraints](#)” on page 217.

What's New in SPD Server 4.5?

The operating system requirements for SPD Server 4.5 are different from the requirements for SPD Server 4.4. For more information about operating system requirements for SPD Server 4.5, see “[SPD Server Pre-Installation and System Requirements Guide](#)” on page 9.

SPD Server 4.5 now supports the Linux x64 platform.

SPD Audit logging has been enhanced to include the user LIBNAME in the proxy and SQL audit logs. For more information, see “[SPD Server Auditing](#)” on page 213 .

You can now specify recycle times for the SPD Server name server log and the SPD Server SNET log. For additional information about configuring SPD Server log cycle times for Windows installations, see “[Configure SPD Server Software on Your Windows Host](#)” on page 48. For additional information about configuring SPD Server log cycle times for UNIX installations, see “[Configuring SPD Server Host Software for Your Site](#)” on page 19.

SPD Server now supports user formats with the put() function that are more than 8 characters in length. An SPD Server host can read user format catalog files that were created by SAS running on Windows, or that were running on the same machine as the SPD Server host. The spdsls list utility has been enhanced to add a -verbose option. The -verbose option provides information such as the number of observations, observation length, index segment size, partition size, and whether the table is compressed, encrypted, or is a member of a cluster. For more information about SPD Server list utilities, see “[SPD Server Table List Utility Spdsls](#)” on page 241.

SAS implicit pass-through SQL now permits SQL queries to SPD Server that include supported SPD Server functions. To see a list of the functions that SPD Server supports via implicit pass-through SQL, see “Differences between SAS SQL and SPD Server SQL” in Chapter 8 of *SAS Scalable Performance Data Server: User's Guide* .

When you install SAS, the SPD Server 4.5 client components for SAS are automatically installed and delivered. For more information about installing SPD Server 4.5 on a Windows platform, see “[Before You Install: Precautions and Required Permissions](#)” on

[page 40](#). For more information about installing SPD Server 4.5 on a UNIX platform, see “[Before You Install: Precautions and Required Permissions](#)” on [page 14](#).

When you install SAS Management Console, SAS Management Console components for SPD Server 4.5 are automatically installed and delivered. For more information about installing SAS Management Console components for SPD Server 4.5 on a Windows platform, see “[Before You Install: Precautions and Required Permissions](#)” on [page 40](#). For more information about installing SAS Management Console components for SPD Server 4.5 on a UNIX platform, see “[Before You Install: Precautions and Required Permissions](#)” on [page 14](#).

Part 2

Installation

Chapter 2

SPD Server Pre-Installation and System Requirements Guide 9

Chapter 3

SPD Server UNIX Installation Guide 13

Chapter 4

SPD Server Windows Installation Guide 39

Chapter 2

SPD Server Pre-Installation and System Requirements Guide

AIX Requirements and Tuning for 64-Bit SPD Server	9
HP-UX Requirements and Tuning for 64-Bit SPD Server	10
System Requirements	10
Kernel Tuning Requirements	10
Required Patches	11
Solaris on SPARC Requirements and Tuning for 64-Bit SPD Server	11
System Requirements	11
Kernel Tuning Requirements	11
Solaris on X64 Requirements and Tuning for 64-Bit SPD Server	11
System Requirements	11
Kernel Parameter Requirements	11
Linux on X64 Requirements and Tuning for 64-Bit SPD Server	12
System Requirements	12
Windows Requirements and Tuning for 32-Bit SPD Server	12
System Requirements	12
SPD Server 4.5 Client Requirements	12
System Requirements	12

AIX Requirements and Tuning for 64-Bit SPD Server

For complete information about AIX tuning for SAS SPD Server, see the white paper selection available on SAS Institute's external Web site at www.sas.com/partners/directory/ibm/papers.html.

HP-UX Requirements and Tuning for 64-Bit SPD Server

System Requirements

- Required OS level: HP-UX 11i 64-bit OS (HP-UX 11.11 for PA-RISC) or HP-IA64 64-bit OS (HP-UX 11i v2, HP-UX 11.23 for Itanium)
- Minimum system configuration: HP-PA 2.0 server system with minimum 2 GB memory

Kernel Tuning Requirements

The following kernel parameters are for HP-UX 11.11 and HP-UX 11.23. You must adjust these parameters on the HP server system on which you will run SPD Server. After you make these changes, reboot the system before you attempt to use the SPD Server. In the following example, MAX(a,b) means to use the maximum of the values a or b.

```
dbc_max_pct = 10%
dbc_min_pct = 2%
max_thread_proc = 512
maxdsiz_64 = 1Gb + MAX(SORTSIZE, INDEX_SORTSIZE)
maxuprc = 4 + #concurrent SPD Server users
nproc = current nproc value + 4 + #concurrent SPD Server users
```

SORTSIZE and INDEX_SORTSIZE are SPD Server parameters in the `spdsserv.parm` file. If you increase the values of these SPD Server parameters, you might also need to adjust the HP-UX kernel parameters. For more information about SPD Server parameters, see [Chapter 3, “SPD Server UNIX Installation Guide,”](#) on page 13 .

Depending on the way that you use the SPD Server, you might also need to increase the value of the following HP-UX kernel parameters:

`ninode=`

maximum number of open inodes in memory. Change this value to the maximum number of SPD tables that will be open concurrently multiplied by the maximum number of partitions in an SPD Server table.

`nfile=`

maximum number of files that can be open system-wide. Change this value to the maximum number of SPD Server tables that will be open concurrently multiplied by the maximum number of partitions in an SPD Server table.

`nflocks=`

maximum number of files that can be locked system-wide. Change this value to the maximum number of SPD Server tables that will be open concurrently.

`maxfiles_lim=`

process hard limit for open files. Change this value to the maximum number of SPD Server tables that will be open concurrently multiplied by the maximum number of partitions in an SPD Server table. The minimum recommended setting is 8192.

Required Patches

Apply the following HP-UX 11.23 for Itanium (IA-64) patches for SPD Server 4.4:

- PHCO_30543 s700_800 11.23 Pthread library cumulative patch
- PHCO_30531 s700_800 11.23 libc cumulative patch
- The HP September 2004 Base Patch Bundle for HP-UX 11.23

Solaris on SPARC Requirements and Tuning for 64-Bit SPD Server

System Requirements

- Required OS level: Solaris Version 5.9

Kernel Tuning Requirements

Adjust the following kernel parameter on Solaris server systems on which you will run SPD Server.

`rlim_fd_max=`
process limit for open files. Change this value to accommodate the maximum number of SPD tables that will be open concurrently multiplied by the maximum number of partitions in an SPD Server table. The minimum recommended setting is 8192.

Solaris on X64 Requirements and Tuning for 64-Bit SPD Server

System Requirements

- Required OS level: Solaris Version 5.10, Update 3

Kernel Parameter Requirements

Adjust the following kernel parameter on Solaris server systems on which you will run SPD Server.

`rlim_fd_max=`
process limit for open files. Change this value to the maximum number of the number of SPD tables that will be open concurrently multiplied by the maximum number of partitions in an SPD Server table. The minimum recommended setting is 8192.

Linux on X64 Requirements and Tuning for 64-Bit SPD Server

System Requirements

- Required OS level: Red Hat Enterprise Linux 4 and 5, SuSE Linux Enterprise Server 9 and 10.

Windows Requirements and Tuning for 32-Bit SPD Server

System Requirements

- Required OS level: Windows NT 4.0 Service pack 3 or later
- Minimum System Configuration: NT server system

SPD Server 4.5 Client Requirements

System Requirements

- Required SAS level: SPD Server 4.5 requires SAS 9.2

Chapter 3

SPD Server UNIX Installation Guide

Before You Install: Precautions and Required Permissions	14
Packing List for SPD Server Distribution	15
SAS Scalable Performance Data Server 4.5 and SAS Deployment Wizard	18
Upgrading and Reinstalling SPD Server	18
Upgrading from SPD Server 3.x to SPD Server 4.5	18
Upgrading from SPD Server 4.x to SPD Server 4.5	18
Upgrading or Reinstalling SPD Server 4.5	18
Configuring SPD Server Host Software for Your Site	19
Configure SPD Server Ports	19
Configure the SPD Server System	19
Configuring the rc.spds File	22
Example rc.spds File	22
SNET	23
SNET Port Assignments	23
Logging	24
Audit File Facility	25
User Password and Parameter Files	25
Verify That SPD Server 4.5 Is Running	25
Configure SPD Server Client Software	26
Test Your SPD Server Installation Using SAS	27
SPD Server Command Reference	29
SPD Server Name Server Commands	29
SPD Server Host Commands	30
SNET Server Commands	32
Password Utility Reference	32
Performance Server Reference	33
SPD Server 4.5 and the SAS Management Console Utility	33
SPD Server Lightweight Directory Access Protocol (LDAP) Authentication	33
Notes for SPD Server Administrators	34
UNIX User IDs	35
SPD Server User IDs	35
LDAP Password Authentication	36
Troubleshooting	36
Name Server Start-Up Failed	36
SPD Server Host Start-Up Failed	37

SAS LIBNAME Assignment Failed	37
Problems Renewing Your SPD Server License	38

Before You Install: Precautions and Required Permissions

Before you install, see [Chapter 1, “SPD Server 4.5 Product Notes,”](#) on page 3 for important information about features in this release. Also read [Chapter 2, “SPD Server Pre-Installation and System Requirements Guide,”](#) on page 9.

Review the following precautions and required permissions:

- SPD Server 4.5 is distributed only as a 64-bit environment application for Solaris by Sun, AIX by IBM, Linux by SUSE or Red Hat, and HP-UX by Hewlett-Packard.
- Use a UNIX user ID other than root to run your production SPD Server environment. Although there are no known security or integrity problems with SPD Server 4.5, root access is not required to run SPD Server. After you correctly configure UNIX directory ownership and set permissions on your LIBNAME domains, root access to SPD Server does not confer any benefits. For more information and a list of options to use when you are configuring SPD Server, see [“Notes for SPD Server Administrators”](#) on page 34.
- Install SPD Server in a location that is adequately mirrored and backed up to assure reliability. The SPD Server installation location should use system space in which the SPD Server Administrator has full rights.
- General familiarity with the UNIX language is required to install SPD Server 4.5. At a minimum, you should be familiar with basic UNIX shell entities (such as sh, csh, and ksh), Bourne shell scripts, the UNIX **tar** command, and how to modify 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. For more information, see [“Notes for SPD Server Administrators”](#) on page 34.
- 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**.
- Insert the **WORKPATH=** server option in your **spdsserv.parm** file. Use the **WORKPATH=** option to configure your server to use a high-performance file system. Ideally this system has RAID-structured volumes with sufficient disk space to accommodate the transient storage needs for SPD Server. The **spdsserv.parm** file is located in the root directory of your SPD Server host installation. For more information about the **WORKPATH=** option and configuring servers for performance, see the *SAS Scalable Performance Data (SPD) Server 4.53: User's Guide*.

Packing List for SPD Server Distribution

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 **bin/** subdirectory contains the following binary files:

ixutil

the data set index utility

psmgr

the password file utility

spdsaud

the audit logger

spdsbase

the LIBNAME proxy

spdsbkup

performs full or incremental table backups

spdsconv

the SPD Server 3.x to SPD Server 4.x table conversion utility

spdseng

the SQL pass-through engine

spdslog

the message logger

spdsls

gives physical file listings for a LIBNAME domain

spdsnsrv

the SPD Server name server

spdsperf

the Performance server

Note: Currently, spdsperf is not available for Linux on X64.

spdsrstr

restores full or incremental table backups

spdserv

the SPD Server host

spdsnet

the ODBC, JDBC, and htmSQL gateway

spqldrive

a stand-alone SQL pass-through driver

spds-clean

the SPD Server disk cleanup utility

dulibv3

the SPD Server 3.x 64-bit version of the shared library used by spdsconv

Note: The `dulibv3` file is included only if your system previously supported SPD Server 3.x tables.

`spdsbased`
the debug version of `spdsbase`

`spdsengd`
the debug version of `spdseng`

`spdsiotest`
the stand-alone SPD Server I/O scalability test

`spdsnslib`
the national language support (NLS) library

`spdsnslibd`
the debug version of `spdsnslib`

The `lib/` subdirectory contains the following SPD Server library files:

`spds.dll`
the application extension library that is accessed via the SAS ODBC driver

`spdslib`
the run-time library that performs SQL pass-through from C and C++ applications to SPD Server

The `samples/` directory contains the following files of interest:

`accolrw.sas`
shows how to use ACL row and column security features.

`auditraw.sas`
used to read proxy audit files that do not include WHERE clause auditing.

`audit.sql.sas`
used to read an SQL audit file.

`doc_examples.sas`
contains sample SAS code that is used in the *SAS Scalable Performance Data (SPD) Server 4.53: User's Guide*. This guide provides SPD Server LIBNAME and data set usage, and syntax options.

`dynamic_cluster*.sas`
shows how to use dynamic clusters with a MIN and MAX variable list.

`fmtgrpby.sas`
shows how to use formatted parallel GROUP BY statements in SQL pass through.

`index_scan*.sas`
shows how to use the SQL index scan performance enhancement.

`killrc`
a shell script that kills all processes related to a run of `rc.spds`. The `killrc` script is selective. It does not kill SPD Server processes that are not related to core processes that were started by `rc.spds`. Those core processes are initially started when `rc.spds` runs. The core processes are typically `spdsnsrv`, `spdserv`, `spdsbase`, `spdslog`, and `spdsnet`, based on the `rc.spds` script in the `/samples` directory.

`killspds`
a shell script that kills all processes that begin with the letters **spds** for a UNIX user. Do not use the `killspds` script if you have any processes running in UNIX that do not belong to SPD Server, but whose executable names begin with the letters **spds**.

libnames.parm
a sample SPD Server host LIBNAME configuration file. Use with the -libname file option for the **spdsserv** command.

libsamp.parm
a more advanced example of a LIBNAME configuration file.

materialize_view*.sas
shows how to use the SQL materialized view performance enhancement.

minmax*.sas
shows how to use a MIN and MAX variable list with an SPD Server table.

paralleljoin*.sas
shows how to use the SQL Parallel Join performance enhancement.

paraload.sas
shows how to perform parallel loads from an existing table into an SPD Server table. This technique exploits a parallel load capability in the LIBNAME proxy.

passthru.sas
demonstrates the use of SQL pass through. This file gives examples of simple, single-level pass through, and secondary libref and connection scenarios.

PerfDataSample.sas
reads a processed Performance Server log into a SAS data set.

process_perf_log
a PERL script that processes a Performance server log and a server log into data that can be read into a SAS data set for post-performance analysis.

pwdb
a script that starts the Password Manager executable.

rc.perf
a Bourne shell script that starts a standard Performance Server.

rc.spds
a Bourne shell script to start a standard SPD Server environment.

scale.sas
can be used to benchmark the scalability of your SPD Server.

spdsinst.sas
demonstrates the simple use of WHERE clauses and WHERE planner output.

spdsserv.par
a sample SPD Server host parameter file. This file sets the defaults for SPD Server options. Use this file with the -PARMFILE option for the **spdsserv** command.

starjoin*.sas
shows how to use the SQL Star Join performance enhancement.

symsub.sas
shows how symbolic substitution in SQL pass-through statements can provide row-level security in tables.

tempwork.sas
demonstrates temporary LIBNAME domain support. Files that are created in a temporary LIBNAME domain are automatically deleted when the SAS session ends.

verify.sas
a SAS installation verification job. You should run this job after you install SPD Server

The **doc/** directory contains information about the location of the online PDF of the *SAS Scalable Performance Data (SPD) Server 4.53: Administrator's Guide* and the *SAS Scalable Performance Data (SPD) Server 4.53: User's Guide*.

The **msg/** directory contains SPD Server message files. Some SPD Server components use the collection of *.m files to generate message text.

The **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.

The **spdsasmc/** directory contains the SAS Management Console files that support SPD Server.

SAS Scalable Performance Data Server 4.5 and SAS Deployment Wizard

You can install SAS Scalable Performance Data Server (SPD Server) 4.5 as part of your initial SAS 9.3 installation or as an add-on product to an existing SAS 9.3 installation. In either case, the SPD Server 4.5 installation is facilitated by the SAS Deployment Wizard. The SAS Deployment Wizard installs SPD Server to the following location on your computer: `<SASROOT>/SASScalablePerformanceDataServer/4.5/`.

Note: `<SASROOT>` is a placeholder for the full path specification to the base directory of your SAS 9.3 installation.

Upgrading and Reinstalling SPD Server

Upgrading from SPD Server 3.x to SPD Server 4.5

SPD Server 3.x tables are not compatible with SPD Server 4.x tables, including tables in SPD Server 4.5. If you want to use your SPD Server 3.x tables with SPD Server 4.5, you must first convert your SPD Server 3.x tables to SPD Server 4.x format. For more information about converting SPD Server 3.x tables for use with SPD Server 4.5, see [“ACL Security Overview” on page 151](#).

Upgrading from SPD Server 4.x to SPD Server 4.5

SPD Server 4.x tables are compatible with SPD Server 4.5. No conversion is required to use SPD Server 4.x tables with SPD Server 4.5. You can start SPD Server 4.5 using domains that include tables created by any SPD Server 4.x host.

Upgrading or Reinstalling SPD Server 4.5

If you need to reinstall SPD Server 4.5, or if you install a newer release of SPD Server 4.5, the installation process does not alter any files that have been modified in the **/site** directory. Any custom modifications that were made to installation files are retained when you upgrade SPD Server to a subsequent release.

Configuring SPD Server Host Software for Your Site

Configure SPD Server Ports

During installation, you can configure the data server port that is used to make connections from the SAS SPD Server LIBNAME engine to the SPD Server. You can also configure the SNET Server port that is used to make connections from a JDBC client to the SPD Server.



The UNIX installation values are stored in the `<InstallDir>/site/rc.spds` start-up script as: `NSPORT=<data server port>` and `SNSPORT=<snet server port>`.

To change these values after installation, you must edit the above script to their new values and restart SPD Server.

After you install SPD Server 4.5, you must configure SPD Server to run on your server machine. The SPD Server 4.5 installation contains only 64-bit components.

Configure the SPD Server System

Complete the following steps to configure the SPD Server system:

1. Issue one of the following commands:
 - For ksh users: `export PATH=$PATH:InstallDir/bin`
 - For csh users: `set path = ($path InstallDir/bin)`
2. Invoke the `pwdb` script from the `/site` subdirectory to create an initial SPD Server password file. Execute the `pwdb` to create the password file in `InstallDir/`.
3. Issue the Password Manager `groupdef` command to define a group called `admingrp`. Next, issue the Password Manager `add` command to add an SPD Server user ID for yourself (assuming that you are the SPD Server administrator). Both the

groupdef and **add** commands prompt you to enter values. Use the following transcript file from a typical command sequence for reference. Notice that the password prompt does not echo any characters as you type. If you want to verify your work, issue the Password Manager **list** command to print the contents of the SPD Server password file, after you issue the **add** command.

You should see content similar to the following example:

```
SAS Scalable Performance Data Server Host 4.50
Password Manager Utility
Copyright (c) 1996-2009 by SAS Institute Inc, Cary NC 27513 USA

Enter command
> groupdef admingrp
Group admingrp defined

Enter command
> add
Enter username to add
> admin
Enter password for admin
>
Verify password
>
Enter authorization level (0 to 7) for admin:
> 7
Enter IP Address or <Return>
>
Enter password expiration time in days
> 365
Enter group name or <Return>
> admingrp
Enter the maximum allowed time (in days) between successful logins <Default
= infinite>
>
Enter the maximum allowed login failures <Default = infinite>
>
Enter admin's performance class(1=LOW 2=MED 3=HIGH carriage return for LOW)
>
User admin added
Enter command
> quit
```

These commands initialize the user password database.

4. Add other user IDs before you open the SPD Server system for use. Authorization level 7 is privileged. Authorization level 7 allows users to circumvent SPD Server ACL security measures. Unlike in the previous example, you should give most or all users authorization level 0 (which is non-privileged), so that users cannot bypass SPD Server security. For more information, see [“Notes for SPD Server Administrators” on page 34](#).

Note: The administrator password expires during the first logon to the SPD Server host. For more information about passwords, see the **psmgr** utility reference documentation.

5. In the **InstallDir/site** directory, edit the **libnames.parm** file to add the site-specific LIBNAME domains that your SPD Server supports. Decide how to organize

your existing disk storage to best exploit the capabilities of the SPD Server. For more information, see [“SPD Server Host Commands” on page 30](#) and the `libsamp.parm` file. For more information about managing resources, see [“Managing Computing Resources with a Name Server” on page 71](#).

6. Edit and configure the resource script file `rc.spds`. In the `/InstallDir/site` directory, use a UNIX text editor to open the `rc.spds` file. The tasks to configure the `rc.spds` file include the following:
 - Specify the SQL audit file cycle time and the file prefix using `AUDTIME=` and `AUDFILESQL=`.
 - Confirm settings for the `INSTDIR=` pointers to your installation directory.
 - Confirm that `LICFILE=` settings point to your `<SASHOME>/licenses/<SPD License File>` directory.
 - Specify whether to start the SNET Server.
 - If you use SNET, check SNET port assignments.
 - Reassign SNET ports if there are conflicts.
 - Specify whether to create a log using `LOGDIR=`.
 - Specify the log cycle time and the file prefix using `LOGTIME=` and `LOGFILE=`.
 - Specify whether to create an audit file facility using `AUDDIR=`.
 - Specify the audit file cycle time and the file prefix with `AUDTIME=` and `AUDFILE=`.
 - Specify the location of your server user password database and parameter files.

For more information about configuring these settings in the `rc.spds` file, see [“Configuring the rc.spds File” on page 22](#).

7. After you have finished making your changes, save and close the `rc.spds` file.
8. Add the following services to your `/etc/services` or `/etc/inet/services` file on the SPD Server host machine. The following example assumes that you want to use registered ports for your SPD Server host, and you choose to use the default SPD Server name server port 5190 and the SNET Server port 5191.

- Service declaration for the SPD Server name server:

```
spdsname 5190/tcp # SPDS Name Server
```

- Server Service declaration for the SNET Server.

```
spdssnet 5191/tcp # SPDS SNET Server
```

You need the SNET service only if you plan to run the SNET Server. By default, the sample `rc.spds` script runs `spdssnet`.

If you choose to use different port addresses, replace the `????` strings in the following examples with unused 4-digit port addresses. Also remember to update your `rc.spds` script accordingly. Determine unused port addresses by scanning the existing addresses and then choosing a number that does not appear. Choose a port number greater than 5000 to avoid conflicts with reserved and system-defined port addresses.

- service declaration for the SPD Server name server:

```
spdsname ????/tcp # SPDS Name Server
```

- service declaration for the SNET Server:

```
spdssnet ???/tcp # SPDS SNET Server
```

Note: If you installed a previous version of SPD Server software and you have the service name **spdsoper** defined, you should remove it from your **/etc/services** or **/etc/inet/services** file on the SPD Server system.

9. Start SPD Server. Execute the **InstallDir/site/rc.spds** script that you customized in the previous steps. The SPD Server environment starts in the context of your current UNIX user ID.

Customizing the **rc.spds** script is important because it defines UNIX ownership and file access permissions on SPD Server resources. Ownership and file permissions are set in the context of the SPD Server run-time environment. If you plan to execute **rc.spds** when your system starts, the **rc.spds** script should be executed in the context of the appropriate UNIX user ID. Using the appropriate UNIX user ID ensures that the resources created in the start-up configuration meet the necessary file ownership and permission requirements for SPD Server.

If you do not use SPD Server user IDs, you still need the SPD Server password file. Without the SPD Server password file, the SPD Server host process does not function correctly. To disable the use of SPD Server user IDs at your site, specify the **-NOACL** option when you start SPD Server.

Configuring the rc.spds File

As part of the installation procedure, you need to configure the **rc.spds** file.

Example rc.spds File

Here is an example of a typical unmodified **rc.spds** file:

```
#!/bin/sh -x
# Sample startup script for SPDS.
# This script starts the SPDS Name Server
# data server and ODBC server processes
# using assumed install directories. Most
# paths are controlled through shell variables
# defined at the beginning of the script.
# If you change this script, copy it to
# the SPDS site directory and modify that
# copy just to make sure that a subsequent
# SPDS software upgrade doesn't wipe out
# your site modifications to the script.
#-----
#
# Define some primary variables. INSTDIR is the
# root directory of your installation. INSTDIR is
# initialized to run rc.spds from the site dir
# of your installation.
#
# NSPORT is the SPDS name server listen port;
#         if omitted uses "spdsname" service entry.
# SNSPORT is the SPDS ODBC server listen port;
#         if omitted uses "spdssnet" service entry.
#
```

```

# If you are running through a firewall the NSPORT and
# SNSPORT must be surfaced through the firewall. In
# addition, the SPDS server listen port and operator port
# must be surfaced through the firewall. If you are not
# running through a firewall allow the server to choose
# these ports.
#
# SRVLPORT is the SPDS server listen port;
#       leave as 0 if NOT running through a firewall.
#
# SRVOPORT is the SPDS server operator port;
#       leave as 0 if NOT running through a firewall.
#
# Refer to the SPDS Admin Doc section on Security for
# more information on running SPDS through a firewall.
#
NSPORT=5190
SNSPORT=5191
SRVLPORT=0
SRVOPORT=0
INSTDIR=<InstallDir>
PARMDIR=$INSTDIR/site
ACLDIR=$INSTDIR/site
LICFILE=<SASHOME>/licenses/<SPD License File>

```

The following sections describe the configuration changes that you might want to make in the rc.spds file.

SNET

The rc.spds script assumes that you want to start the SNET Server (spdssnet) to support ODBC, JDBC, or htmSQL access to SPD Server data stores. If this is not what you want, you can delete or comment out the following lines near the bottom of the rc.spds script.

```

# Startup the spdssnet server. This server supports
# ODBC access to SPDS data. Note the
# only parameter is the optional spdssnet listen
# port number. If not explicitly specified it
# will default to the "spdssnet" service in /etc/services
#

/bin/sleep 2
if [ -z "$SNSPORT" ]; then
    spdssnet 1>$SNSLOG 2>&1 &
else
    spdssnet -listenport $SNSPORT 1>$SNSLOG 2>&1 &

```

SNET Port Assignments

The rc.spds script assumes that you are running SPD Server concurrently with an SPD Server 3.x environment. The script also assumes that the SPD Server name server and the SNET Server will run using explicit port number assignments. The following lines at the beginning of rc.spds assign the ports numbers:

```
NSPORT=5190 (name server port for spdsnsrv)
SNSPORT=5191 (SNET Server port for spdssnet)
```

If these ports are in use, or are otherwise unavailable, choose new port numbers. If you omit these assignments, rc.spds uses the name services entries SPDSNAME and SPDSSNET. If you do not run the SNET Server, you do not need to modify the SNSPORT definition.

Logging

The rc.spds script assumes that you want to keep the logs from messages written to STDOUT or STDERR of the spdsnsrv (SPD Server name server) and spdsserv (SPD Server host) processes. The shell variable LOGDIR= defines the directory in which these logs are kept. If you do not want to keep these logs, change LOGDIR=, and the rc.spds script will use `/dev/null`. If you want to keep the logs in another location besides `InstallDir/log`, change LOGDIR=.

The DSRVFILE= and DSRVTIME= spdsserv options, NSRVFILE= and NSRVTIME= spdsnsrv options, and SNSFILE= and SNSTIME= spdssnet options are enabled with the following shell variables by default:

```
DSRVFILE=spdsserv
    specifies the spdsserv process log file prefix

DSRVTIME=00:00
    specifies the time of day to cycle the spdsserv log file

NSRVFILE=spdsnsrv
    specifies the spdsnsrv process log file prefix

NSRVTIME=00:00
    specifies the time of day to cycle the spdsnsrv log file

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

SNSTIME=00:00
    specifies the time of day to cycle the spdssnet log file
```

Specify the log file prefix and the log file cycle time of day to enable automatic log filename generation and cycling of the log file. The file path for the -logfile option is generated by concatenating the LOGDIR= and LOGFILE= variables. For more information about these options, see [“SPD Server Host Commands” on page 30](#). When automatic log filename generation and cycling are enabled, the only messages that are sent to the default log file are those written to STDERR. If you want to disable automatic log filename generation and cycling, change the settings to empty pointers, such as the DSRVFILE= and DSRVTIME= options.

The LOGFILE= and LOGTIME= spdsserv options are enabled by default with the following shell variables:

- **LOGFILE=spdsserv** specifies the spdsserv process log file prefix
- **LOGTIME=00:00** specifies the time of day to cycle the log file

These settings enable automatic log filename generation and cycling by specifying the log file prefix and the log file cycle time of day. The file path for the -logfile option is generated by concatenating the LOGDIR= and LOGFILE= variables. For more information about these options, see [“SPD Server Host Commands” on page 30](#). When automatic log filename generation and cycling are enabled, the only messages that go to the `InstallDir/log/spdsserv.log` file are those written to STDERR. If you want

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

Audit File Facility

The rc.spds script enables you to use the SPD Server audit file facility, but the audit file facility is not enabled by default. Use the following shell 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

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 the audit file facility, see [“SPD Server Host Commands” on page 30](#).

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 the ACLDIR= and PARMDIR= assignments. You can include this script into your system start-up file so that it is executed as part of starting the system. Otherwise, the SPD Server administrator must manually start SPD Server after the system starts.

Verify That SPD Server 4.5 Is Running

If you connected to SPD Server through a SAS connection, verify that both the SPD Server name server (spdsnsrv) and the SPD Server host (sdpserv) processes are running.

1. Issue the UNIX **ps** command. You should see processes for spdsnsrv, spdsserv, spdsbase (row-level integrity proxy), and spdssnet as shown in the following example:

PID	TTY	TIME	CMD
24012	pts/26	00:00:00	ksh
24114	pts/26	00:00:00	spdsnsrv
24116	pts/26	00:00:00	spdslog
24117	pts/26	00:00:00	spdsserv
24119	pts/26	00:00:00	spdslog
24120	pts/26	00:00:00	spdssnet
24130	pts/26	00:00:00	spdslog
24136	pts/26	00:00:00	spdsbase
24139	pts/26	00:00:00	ps

2. If the spds* processes are not running, check the logs for errors. Unless you change the log file defaults in rc.spds, the following log paths apply:

- `InstallDir/log/spdsnsrv.log`
- `InstallDir/log/spdsnsrv_mmddyyyy_hh:mm:ss.spdslog`
- `InstallDir/log/spdsserv.log`
- `InstallDir/log/spdsserv_mmddyyyy_hh:mm:ss.spdslog`
- `InstallDir/log/spdssnet.log`
- `InstallDir/log/spdssnet_mmddyyyy_hh:mm:ss.spdslog`

3. If there were problems during start-up and any processes failed to initialize, terminate the remaining SPD Server processes before re-invoking the rc.spds script. Use the **killspds** shell script in the `\samples` directory, or terminate the process manually using the UNIX **kill** command as shown in the following example:

```
$ kill 834 831 832 836 835
```

Upgrade Notice: If you have upgraded from SPD Server 3.x to SPD Server 4.5 and you are satisfied with your SPD Server installation, you should copy the libnames.parm file from your SPD Server 3.x location to your SPD Server 4.5 location. The new libnames.parm file overwrites the temporary file that was created when you verified your SPD Server 4.5 installation. The new file provides you with access to all of the SPD Server 3.x LIBNAME domains from your previous environment.

Configure SPD Server Client Software

Use the SPD Server client software to make SAS LIBNAME connections and perform user-specified operations on the SPD Server host. The SPD Server client software is installed with SAS 9.3 Foundation at `<SASROOT>/SASFoundation/9.3/sasexe`. The SAS 9.3 configuration file automatically includes your SPD Server client software directory in its required path list. The client software contains the following SAS modules:

- `sasspds` is the LIBNAME engine that is required to access the SPD Server environment from SAS 9.3.
- `sasspdo` is the SPD Server operator procedure that is required to access the SPD Server 4.5 environment from SAS 9.3.
- `spds.msg` is the message file for the SPD Server LIBNAME engine and SPD Server operator procedure. This file is compatible with SAS.

After you have configured the SPD Server environment and the SPD Server is running, you need to complete other installation tasks on the SAS clients that will use SPD Server. Because some of these tasks might involve the system that is actually running SPD Server, you might have already completed some of the following steps when you installed the SPD Server host. If that is the case, skip the duplicated steps.

Complete the following steps on each SAS client that will access SPD Server:

1. 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 ???/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.

2. The SPD Server can be accessed with the SAS 9.3 ODBC driver, JDBC driver, and htmSQL driver. Download these drivers on the **Support** tab at support.sas.com.
3. ODBC drivers only: ODBC client applications require you to install the spds.dll application extension. To install the ODBC client application extension:
 - a. Install the SAS 9.3 ODBC driver
 - b. Copy the `InstallDir/lib/spds.dll` directory to the `<drive letter>:\Program Files\sas\shared files\general` directory.
 - c. Configure an ODBC data source for direct SPD Server access.

Test Your SPD Server Installation Using SAS

Testing your SPD Server installation is simple. To verify your installation, you make two SAS LIBNAME assignments using the SPD Server LIBNAME engine. The examples in this section refer to the SASSPDS engine, which is the engine for SAS 9.3.

1. Start the SPD Server environment by executing your customized rc.spds script. Execute this script from the UNIX user ID that owns the LIBNAME directories that are configured in the SPD Server LIBNAME file. For more information about the rc.spds script and SPD Server LIBNAME files, see “[Notes for SPD Server Administrators](#)” on page 34.
2. On a client system that is configured correctly, invoke SAS, and make the following LIBNAME assignments:

```
LIBNAME test sasspds 'tmp'
      server=serverNode.port
      user='anonymous' ;
```

serverNode is the server's node name. port is either the numeric value assigned to NSPORT from the rc.spds file, or the service name that you use to access the SPD Server name server. If you used the sample rc.spds, your LIBNAME assignment looks similar to the following example:

```
LIBNAME test sasspds 'tmp'
      server=serverNode.5190
      user='anonymous' ;
```

If you use the spdsname service, your LIBNAME assignment looks similar to the following example:

```
LIBNAME test sasspds 'tmp'
      server=serverNode.spdsname
      user='anonymous' ;
```

In addition, you should verify that the row-level integrity LIBNAME assignment works correctly:

```
LIBNAME testrl sasspds 'tmp'
      server=serverNode.port
      user='anonymous' locking=YES;
```

When you verify these statements, you confirm the connectivity between the SAS client and the SPD Server environment. When you successfully make these LIBNAME assignments, the network configuration is correct and most of the SPD Server configuration is correct.

Substitute `serverNode` with the node name that runs the SPD Server environment that you want to test. This node is the node that invokes `rc.spds`. The test assumes the temporary LIBNAME definition in the sample `libnames.parm` file was not changed during installation.

3. Watch the SAS log for error messages. You might see messages that indicate that one or more required SPD Server components are not configured correctly.

If you receive one of the following error messages, check your `-PATH` option to confirm that the directory in which you installed SAS components is correctly set.

```
ERROR:      Protocol version mismatch. Proxy version
            is 4.5 while engine version is 3.x.
```

```
ERROR:      Module TEST not found in search paths.
```

```
ERROR:      Error in the LIBNAME or FILENAME
            statement.
```

If you receive the following error message (or a similar message) that describes failures to access messages, check your `-SASMSG` option. Confirm that the directory in which you installed SPD Server components is correctly set:

```
ERROR: unable to access message 608.108
```

If the attempted connection to the SPD Server does not respond for several minutes, check the `-PATH` option and confirm that the directory in which you installed SAS components is correctly set. The `spds45` client component directory must be at the beginning of the path option.

4. After you have assigned every LIBNAME, you can further verify your installation by running the sample SAS program, `InstallDir/samples/verify.sas`. Issue the SAS command to execute the program: `%include 'InstallDir/samples/verify.sas'/source2;`

This test exercises many features of the SPD Server LIBNAME engine and proxy, and verifies that your installation is configured correctly. The test performs a sequence of DATA and PROC steps using a generated data set. It checks the results expected from various DATA step queries. If any query fails to produce the expected result, the SAS job is terminated. The job `verify.sas` requires that the SAS librefs TEST and TESTRL are assigned as shown in [Step 2 on page 27](#).

5. Verify that SQL pass-through services are working in SPD Server by issuing the following SAS commands:

```
%let spdshost=serverNode;
```

```
%let spdsport=port;
%include 'InstallDir/samples/verptsq1.sas'/source2;
```

Note: **serverNode** and **port** are described in [Step 2 on page 27](#).

SPD Server Command Reference

SPD Server operation revolves around the executable files described in the packing list. The executables are in the **/bin** subdirectories. Each executable supports a set of command-line options that override default features, or it provides site-dependent configuration information. The command-line options for each executable are described in the following sections.

SPD Server Name Server Commands

The SPD Server LIBNAME engine connects to the SPD Server name server. The name server resolves LIBNAME domain names into physical file system paths for librefs. The name server also resolves host node and end-point (TCP port) addresses for each LIBNAME. Each SPD Server (spdssrv) host process registers LIBNAME domain information from its configuration file with its appointed name server (spdsnsrv) process. Multiple SPD Server hosts can use the same name server to register their LIBNAME domains. The only requirement is that the combination of the LIBNAME= option values from the SPD Server host's LIBNAME configuration file must be unique across all SPD Server hosts that connect to the name server.

Part of the function of the name server process is to start an SPD Server logging process. The spdslog process performs message logging functions. Message logging functions include automatic log filename generation and periodic log file cycling. You can periodically switch to a new name server log file without stopping and restarting SPD Server, which improves the availability of the name server.

The spdsnsrv command-line options control automatic log filename generation and cycling properties. The default rc.spds script in the **samples/** directory of your SPD Server installation provides examples of spdsnsrv command-line options.

When you are using automatic log filename generation and cycling, remember to periodically clean the log files. Proper log file maintenance includes archiving logs using secondary or long-term storage. Many users retain only a few generations of log files for quick reference. A shell script that runs on a regular basis (such as CRONTAB) is a good way to perform log maintenance on your server machine.

Invoke the SPD Server name server by using the following command-line syntax:

```
spdsnsrv [-option [optval]...]
```

The **spdsnsrv** command supports the following options:

-listenport *port#*

specifies the explicit TCP port number that the name server uses to accept connections from the SPD Server LIBNAME engine and its SPD Server hosts. If no port is specified, the name server queries the system for port addresses using the service name spdsname. If no such service has been registered, SPD Server chooses a dynamic port number for the name server to use.

-licensefile *lic-file*

License file keys are generated by SAS and provided to you. With this release of SPD Server, you receive an SPD Server license key for each machine on which you

license SPD Server. The SPD Server administrator must enter each key into the license file. The SPD Server will not run on a machine whose valid license key has not been entered into the license file. License keys are plain text strings that include product, site, and machine information, and the password that is required for using Server in this specific environment.

-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 `\DOWNlogs\spdsnsrv`, the name `\DOWNlogs\spdsnsrv_mmddyyyy_hh:mm:ss.spdslog` is generated. The values `mmddyyyy` and `hh:mm:ss` indicate the time when the system created the log file.

-logtime *hh:mm*

specifies the time of day to cycle a new generation of the name server log file. At this time each day, the previous log file will be closed and a new log file will be opened.

SPD Server Host Commands

The SPD Server LIBNAME engine connects to the SPD Server host to access data in the server environment. The SPD Server host uses the SPD Server password file to validate each SPD Server user, and then creates a LIBNAME proxy process on behalf of each of them.

Part of the function of the SPD Server host process is to start SPD Server logging processes. The `spdslog` process performs message logging functions. The `spdsaud` process performs audit logging functions. The `spdserv` command-line options control the message and audit logging functions. Message and audit logging functions include automatic log filename generation and log file cycling support. You can periodically switch to a new name server log file without stopping and restarting SPD Server, which improves the availability of the name server.

The `spdserv` command-line options control automatic log filename generation cycling properties. The default `rc.spds` script shipped in the `samples/` directory of your SPD Server installation provides examples of the command-line options.

Audit log records are kept for all resources that are accessed by each LIBNAME proxy process. The audit log saves records in its own separate space, away from other server log files. A sample SAS job that processes the audit log and generates a report is provided. Check `samples/audit.sas` for information about processing the audit log and generating the report. To enable the audit log, use the `spdserv` command with the `-AUDITFILE` option.

When you are using automatic server log cycling or audit log cycling, remember to periodically clean the log files. Proper log file maintenance includes archiving logs using secondary or long-term storage. Many users retain only a few generations of log files for quick reference. A shell script that runs on a regular basis (such as CRONTAB) is a good way to perform log maintenance on your server machine.

Invoke the SPD Server host by using the following command-line syntax:

```
spdserv [-option [optval]...]
```

The `spdserv` command supports the following options:

-acldir *pwd-dir-path*

specifies the directory path to the SPD Server host SPD Server password file. You can omit this option if the PASSPATH option is declared in the SPD Server host's -PARMFILE option. A valid SPD Server password file is required even when you specify the -NOACL option. You must use the SPD Server psmgr utility to create the password file and to populate it with the set of valid SPD Server user IDs.

-auditfile *fileSpec*

enables audit logging for the server and automatic audit log file creation by the audit process. *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.

-auditime *hh:mm*

specifies the time of day to cycle a new generation of the audit log file. At this time each day, the previous log file will be closed and a new log file will be opened.

-libnamefile *file-spec*

specifies the name of the file that contains the logical LIBNAME domain definitions that the SPD Server host supports. LIBNAME definitions can span multiple lines and must begin with the LIBNAME=*name* keyword. For more information about SPD Server LIBNAME parameter files, see [“Setting Up SPD Server Libname Parameter Files” on page 125](#).

-logfile *fileSpec*

specifies that the logger process automatically creates a server log file. The *fileSpec* value specifies a partial path or filename that is used to generate the complete log file path. For example, if you specify *fileSpec* as **/logs/spds**, the generated name will be **/logs/spds_mmddyyyy_hh:mm:ss.spdslog**, where **mmddyyyy** and **hh:mm:ss** indicate the system time when the log file was created.

-logtime *hh:mm*

specifies the time of day to cycle a new generation of the server log file. At this time each day, the previous log file will be closed and a new log file will be opened.

-nameserver *node-name*

specifies the node name where the name server process is running. This node does not need to be the same node that is hosting the SPD Server host processes. This option is required.

-nameserverport *port#*

specifies an explicit TCP port number for the SPD Server host to use to connect to its name server. If you do not specify a port, the name server queries the system for a registered port address that is using the service name spdsname.

-noacl

disables SPD Server login validation for SPD Server LIBNAME engine connections to the SPD Server host.

-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 spdserv.parm is in the process's current working directory. Option declarations in this file are of the following form:

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

The recognized -parmfile option names are listed, but full descriptions are available only in online documentation. You do not need to modify the default values in **InstallDir/site/spdserv.parm** for most environments. For more

information about the parameter file and setting up server parameters, see [“Setting Up SPD Server Parameter Files”](#) on page 113.

SNET Server Commands

The SNET Server is the connection point for clients accessing SPD Server data through ODBC, JDBC, or htmSQL applications. Part of the function of the SPD Server SNET process is to perform SPD Server logging. The `spdsslog` process manages SPD Server message logging functions. You configure SPD Server message logging functions using `spdssnet` command-line options.

Message logging facilities include automatic log filename generation and periodic log file cycling support. SNET server availability improves because you can periodically switch to a new name server log file without stopping and restarting SPD Server.

The default `rc.spds` script in the `samples/` directory provides examples of the command-line options.

When you are using automatic SNET server log cycling, remember to periodically clean the log files. Proper log file maintenance includes archiving logs using secondary or long-term storage. Many users retain only a few generations of log files for quick reference. A shell script that runs on a regular basis (such as CRONTAB) is a good way to perform log maintenance on your server machine.

Invoke the SNET Server by using the following command-line syntax:

```
spdssnet [-listenport listen_port]
```

The **spdssnet** command supports the following options:

-listenport *listen_port*

specifies the listen port number that `spdssnet` uses to accept connections from ODBC, JDBC, or htmSQL clients. If you do not specify a listen port number, `spdssnet` uses the name service `spdssnet` from the `/etc/services` file to determine its listen port.

-logfile *fileSpec*

specifies that the logger process automatically creates a server log file. The *fileSpec* value specifies a partial path or filename that is used to generate the complete log file path. For example, if you specify *fileSpec* as `\DOWNlogs\spdssnet`, the generated name will be `\DOWNlogs\spdssnet_mmddyyyy_hh:mm:ss.spdsslog`, where `mmddyyyy` and `hh:mm:ss` indicate the system time when the log file was created.

-logtime *hh:mm*

specifies the time of day to cycle a new generation of the SNET log file. At this time each day, the previous log file will be closed and new log file will be opened.

Password Utility Reference

The SPD Server administrator can use the SPD Server `psmgr` utility to create and maintain the data set that contains the authorized SPD Server user IDs. This utility is the SPD Server analog of the UNIX user ID facility. If you choose to run SPD Server ACL support, you need to use this utility to create and populate the SPD Server password file before you start the SPD Server. For more information about the Password Manager utility, see [“The Password Manager Utility `psmgr`”](#) on page 191.

Performance Server Reference

The SPD Performance Server is available to monitor and log the activity of the SPD Server processes. The SAS Management Console SPD Server Manager can connect to the Performance Server to provide real-time feedback about the SPD Server process. For more information about SPD Server performance monitoring, see [“Accessing SPD Server Services in SAS Management Console” on page 76](#).

SPD Server 4.5 and the SAS Management Console Utility

The SAS Management Console offers a standardized, common management tool that enables enterprises to support many technologies from a single point of administration. This client application uses an extensible plug-in architecture that enables you to customize the console to support a wide range of administrative capabilities. When you install SAS 9.3 and SPD Server 4.5, you are prompted to install the SAS Management Console utility. For more information about SAS Management Console, see [“Overview of SAS Management Console” on page 75](#).

SPD Server Lightweight Directory Access Protocol (LDAP) Authentication

In SPD Server for Solaris, AIX, HP-UX, and HP Integrity Itanium, clients can be authenticated by psmgr, or by an LDAP Server such as Microsoft Active Directory, Sun Java System Directory Server, or OpenLDAP (for more information, go to www.openldap.org). 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 psmgr or the SAS Management Console utility so that all other SPD Server user information is recorded.

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

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, the 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.

Note: If you use the LDAP_HOST value for LDAPSERVER, SPD Server might fail during start-up.

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 0-65536. The default setting is the LDAP_PORT value.

LDAPBINDMETH

controls the way 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.

The value LDAP_AUTH_SASL forces secure authentication and is the default authentication method. LDAPBINDMETH="LDAP_AUTH_SASL" tells the LDAP server to authenticate SPD Server clients with the Simple Authentication and Security Layer (SASL) method. SASL is the preferred authentication method for secure environments. When the SPD Server authenticates with SASL, the SPD Server specifies that the DIGEST-MD5 mechanism is used. DIGEST-MD5 is the most common LDAP authentication and is a requirement for all Version 3 LDAP server products.

LDAPBINDDN

the distinguished name (DN), or the location in the LDAP Server's database where the client's information is stored. LDAPBINDDN is a Lightweight Directory Access Protocol term. LDAPBINDDN is the combination of the user name and the network domain in which they operate. 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 **uid**. 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.

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 needs a UNIX login ID on the SPD Server machine. Other SPD Server users do not need UNIX login IDs. You can control other users' access to SPD Server data resources without giving them special login accounts by using the SPD Server password facility. Withholding login accounts to these users adds a measure of security and control, and SPD Server users are permitted physical access to the SPD Server machine.

You should add the **InstallDir/bin** directory to your PATH environment variable by using your shell's login script. If you use ksh, modify the `.profile` or `.kshrc` files. If you use csh, modify the `.login` or `.cshrc` files, depending on where you currently set the PATH environment variable. This modification makes invoking the various SPD Server utility programs much easier.

Run your SPD Server environment using the same UNIX user ID that was used to install SPD Server on the server 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 to the SPD Server data resources.

Regardless of how the SPD Server run-time environment is configured, SPD Server processes always run with a UNIX user ID. 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 are starting SPD Server processes and running SPD Server administrator utilities. Otherwise, it is possible to create files that have ownership and permissions that deny required access to the SPD Server processes. If you perform all SPD Server installation and administration tasks from the same UNIX user ID, subsequent use of the SPD Server is much easier.

Here are some options for establishing the appropriate UNIX user ID for your SPD Server processes:

- Establish a dedicated UNIX account for the SPD Server administrator. Always execute the `rc.spds` script from that account.
- The `rc.spds` script that starts the SPD Server processes should use the `setuid` bit. It does not matter who executes the script, the user ID of the shell executing the script is the script owner. This ensures that SPD Server processes run with the correct UNIX user ID.
- When you start the system, use the UNIX **su** command to establish the correct UNIX user ID for the shell that executes the `rc.spds` script. To start the environment manually, you must enter the password for each UNIX account in your **su** command, unless you are the root when you execute the **su** command.

SPD Server User IDs

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

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.

The SPD Server password file allows better access control to SPD Server's data resources than a native UNIX user ID does. Many sites do not want to give UNIX

accounts to SPD Server system users, but still want to retain protection of and ownership of the data resources that were created in the SPD Server environment. In this case, SPD Server user IDs provide an extra layer of access control.

If you do not use SPD Server user IDs, you still need the SPD Server password file. Without the SPD Server password file, the SPD Server host process does not function correctly. To disable the use of SPD Server user IDs at your site, specify the `-NOACL` option when you start SPD Server.

If you use SPD Server user IDs, add them to the SPD Server password file that was created during installation. The `psmgr` command reads its commands from stdin, so you can pipe commands to it from another command, script, or input file.

LDAP Password Authentication

LDAP authentication causes SPD Server to authenticate an SPD Server user password using LDAP, rather than using the password in the password database. LDAP authentication allows an SPD Server user to have the same user ID and password as their UNIX logon, as long as the UNIX logon meets the SPD Server character restrictions for user IDs and passwords.

You can select the mode of password authentication with server parameters. You can choose between using `psmgr` or LDAP. After you select the mode, all authentication is performed using the that mode. When you use LDAP authentication, an SPD Server user must be entered in the SPD Server password database in order to maintain other information that SPD Server requires, such as a user's groups and access levels.

For more information about SPD Server LDAP authentication, see [“LDAP Authentication Notes” on page 203](#).

Troubleshooting

Troubleshooting networked applications is often difficult. Key information for SPD Server troubleshooting can be found in the name server and SPD Server host process log files. With those two log files, you can reconstruct SAS interaction with SPD Server components. Entries in these 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:

- The LIBNAME engine selector that is specified in the LIBNAME statement is invalid. The SPD Server engine name is **sasspds** and is misspelled in the following LIBNAME statement:

```
LIBNAME foo sasspps 'test'
      server=sunspot.spdsname
      passwd='xxx';
ERROR: Module FOO not found in search paths.
ERROR: Error in the LIBNAME or FILENAME statement.
```

- The logical LIBNAME domain name that is specified in the LIBNAME statement is invalid. The domain name **test** is not defined in the SPD Server name server sunspot.spdsname.

```
LIBNAME foo sasspds 'test'
      server=sunspot.spdsname
      passwd='xxx';
ERROR: ERROR: Libname path info not found in SPDS name server..
ERROR: Error in the LIBNAME or FILENAME statement.
```

- No name server is running on the specified node name or no name server is available at the specified port address. In the following example, no name server is running on the specified node **stelling**. This same message is generated if the port address is incorrect.

```
LIBNAME foo sasspds 'test'
      server=stelling.spdsname
```

```
passwd='xxx';
ERROR: Unable to connect to SPDS name server.
ERROR: Connection refused.
ERROR: Error in the LIBNAME or FILENAME statement.
```

- An invalid or unknown node name is specified in the LIBNAME statement. In the following example, node **xxx** is not accessible in the network.

```
LIBNAME foo sasspds 'test'
server=xxx.spdsname
passwd='xxx';
ERROR: Unable to connect to SPDS name server.
ERROR: xxx.
ERROR: Error in the LIBNAME or FILENAME statement.
```

- An invalid SPD Server user password is specified in the LIBNAME statement. In the following example, the SPD Server user ID is derived from the UNIX user ID that is running the SAS session. The SPD Server password file has an entry for this SPD Server user ID, but the password is not **xxx**.

```
LIBNAME foo sasspds 'test'
server=sunspot.spdsname
passwd='xxx';
ERROR: Error on server LIBNAME socket.
ERROR: SPD server has rejected login from user
sasetb.. ERROR: Error in the LIBNAME or FILENAME
statement.
```

- An invalid SPD Server user ID is specified in the LIBNAME statement. In the following example, the SPD Server user ID **xxx** does not exist in the SPD Server host's password file. The resulting message is the same if the password is invalid.

```
LIBNAME foo sasspds 'test'
server=sunspot.spdsname
user='xxx' passwd='xxx';
ERROR: Error on server LIBNAME socket.
ERROR: SPD server has rejected login from user xxx..
ERROR: Error in the LIBNAME or FILENAME statement.
```

Problems 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.

Note: Do not change the licensing information unless you are logged in with the user ID of the owner of SPD Server. You designated the owner of these files when you licensed the software.

Chapter 4

SPD Server Windows Installation Guide

Before You Install: Precautions and Required Permissions	40
Packing List for SPD Server Distribution	40
SAS Scalable Performance Data Server 4.5 and SAS Deployment Wizard	43
Upgrading and Reinstalling SPD Server	44
Upgrading SPD Server 3.x to SPD Server 4.5	44
Upgrading SPD Server 4.x to SPD Server 4.5	44
Upgrading or Reinstalling SPD Server 4.5	44
Validate Default Port and Library Assignments	44
Verify the Name Server Port Assignment and TMP Path	45
Initialize the Password Manager Database	45
Install SPD Server as a Service	46
Start and Stop SPD Server	47
Configure SPD Server Software on Your Windows Host	48
Install and Configure SPD Server Clients	49
Use SAS to Test Your SPD Server Installation	50
SPD Server Command Reference	52
SPD Server Name Server Commands	52
SPD Server Host Commands	53
SNET Server Commands	54
PSMGR Utility	55
SPD Server and the SAS Management Console	55
SPD Server Lightweight Directory Access Protocol (LDAP) Authentication	55
SPD Server User IDs	56
Troubleshooting	57
Name Server Start-Up Failed	57
SPD Server Host Start-Up Failed	57
SAS LIBNAME Assignment Failed	58
Problems Renewing Your SPD Server License	59

Before You Install: Precautions and Required Permissions

Review the following precautions and list of required permissions before you install SPD Server:

- SPD Server is compatible with the Microsoft Win32 API. SPD Server 4.5 runs on computers using Windows NT, Windows XP, Windows Vista, and Windows 2000 operating environments.
- You need Write access to your server machine's `\etc\services` file if you want SPD Server clients to connect to the SPD Server host by using name services instead of by specifying port numbers at invocation. Name services require you to define registered ports that will 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 `C:\winnt\system32\drivers\etc\services`.
- Insert the `WORKPATH=` server option in your `spdsserv.parm` file. Use the `WORKPATH=` option to configure your server to use a high-performance file system. Ideally, this system has a RAID-structured volume with sufficient disk space to accommodate the transient storage needs of SPD Server. The `spdsserv.parm` file is located in the `InstallDir\DOWN` directory of your SPD Server host installation.

Packing List for SPD Server Distribution

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 `bin\` subdirectory contains the following binary files:

`dulibv3.dll`

the SPD Server 3.x 64-bit version of the shared library that is used by `spdsconv`

Note: The `dulibv3` file is included only if your system previously supported SPD Server 3.x tables.

`ixutil.exe`

the data set index utility

`psmgr.exe`

the password file utility

`spdsaud.exe`

the audit logger

`spdsbase.exe`

the LIBNAME proxy

`spdsbased.exe`

the debug version of `spdsbase`

spdsbat.exe
an executable that makes SPD Server function as a service

spdsbkup.exe
performs full or incremental table backups

spds-clean.exe
the SPD Server disk cleanup utility

spdsconv.exe
the SPD Server 3.x to SPD Server 4.x table conversion utility

spdseng.dll
the SQL pass-phrough engine

spdsengd.dll
the debug version of spdseng

spdsiotest.exe
the standalone SPD Server I/O scalability test

spdslog.exe
the message logger

spds-ls.exe
gives physical file listings for a LIBNAME domain

spds-makesvc450.exe
an executable that installs SPD Server as a service

spdsnlslib.dll
the national language support (NLS) library

spdsnlslibd.dll
the debug version of spdsnlslib

spdsnsrv.exe
the SPD Server name server

spds-serv.exe
the SPD Server host

spds-net.exe
the ODBC, JDBC, and htmSQL gateway

spdsrstr.exe
restores full or incremental table backups

spqldrive.exe
a standalone SQL pass-through driver

The **lib** subdirectory contains the following SPD Server library files:

spds.dll
the application extension library that is accessed with the SAS ODBC driver

spdslib.dll
the run-time library that performs SQL pass-through from C and C++ applications to SPD Server

The **samples** directory contains the following files of interest:

audit.sql.sas
used to read an SQL audit file.

auditraw.sas
used to read proxy audit files that do not include WHERE clause auditing.

- auditwh.sas
used to read proxy audit files that include WHERE clause auditing.
- libnames.parm
a sample SPD Server host LIBNAME configuration file. Use this file with the -LIBNAMEFILE option for the **spdsserv** command.
- libsamp.parm
a more advanced example of a LIBNAME configuration file.
- spdsserv.parm
a sample SPD Server host parameter file. This file sets the defaults for SPD Server options. Use this file with the -PARMFILE option for the **spdsserv** command.
- rc.spds
a Bourne shell script to start a standard SPD Server environment.
- doc_examples.sas
contains sample SAS code that is used in the *SAS Scalable Performance (SPD) Data Server 4.53: User's Guide*. This guide provides SPD Server LIBNAME and data set usage, and syntax options.
- verify.sas
a SAS installation verification job. You should run this job after you install SPD Server.
- spdsinst.sas
demonstrates the simple use of WHERE clauses and WHERE planner output.
- passthru.sas
demonstrates SQL pass-through usage. This file gives examples of simple, single-level pass-through and secondary libref, and connection scenarios.
- tempwork.sas
demonstrates temporary LIBNAME domain support. Files that are created in a temporary LIBNAME domain are automatically deleted when the SAS session ends.
- paraload.sas
shows how to perform parallel loads from an existing table into an SPD Server table. This technique exploits a parallel load capability in the LIBNAME proxy. The LIBNAME proxy uses the same technology as the SQL LOAD TABLE statement.
- aclcolrw.sa
shows how to use ACL row and column security features.
- symsub.sas
shows how symbolic substitution in SQL Pass-Through statements can provide row-level security in tables.
- fmtgrpby.sas
shows how to use formatted parallel GROUP BY statements in SQL pass-through.
- scale.sas
can be used to benchmark the scalability of your SPD Server.
- dynamic_cluster*.sas
shows how to use dynamic clusters with a MIN and MAX variable list.
- minmax*.sas
shows how to use a MIN and MAX variable list on an SPD Server table.
- paralleljoin*.sas
shows how to use the SQL Parallel Join performance enhancement.

starjoin*.sas

shows how to use the SQL Star Join performance enhancement.

index_scan*.sas

shows how to use the SQL index scan performance enhancement.

materialize_view*.sas

shows how to use the SQL materialized view performance enhancement.

The **doc/** directory contains information about the location of the online PDF of the *SAS Scalable Performance Data (SPD) Server 4.53: Administrator's Guide* and the *SAS Scalable Performance Data (SPD) Server 4.53: User's Guide*.

The **msg** directory contains SPD Server message files. Some SPD Server components use the collection of *.m files to generate message text.

The **site** directory is a storage directory for a user's site-specific customization of the sample SPD Server start-up and configuration files. The files in this directory include:

spdsmakesvc.bat

batch file that makes SPD Server a service

spdsremovesvc.ba

batch file that removes the SPD Server service

spdsstartsvc.bat

batch file that starts the SPD Server service

spdsstopsvc.bat

batch file that stops the SPD Server service

spdsnsrv.bat

batch file that starts the SPD Server name server

spdsnet.bat

batch file that starts the SPD Sever SNET Server

spdserv.bat

batch file that starts SPD Server

The **spds\smc** directory contains the SAS Management Console files that support SPD Server.

SAS Scalable Performance Data Server 4.5 and SAS Deployment Wizard

You can install SAS Scalable Performance Data Server (SPD Server) 4.5 as part of your initial SAS 9.3 installation or as an add-on to an existing SAS 9.3 installation. In either case, your SPD Server 4.5 installation is facilitated by the SAS Deployment Wizard. The SAS Deployment Wizard installs SPD Server to the following location on your computer: **<SASROOT>\SAS\scalablePerformanceDataServer\4.5**. See your SAS documentation for more information about the SAS Deployment Wizard.

Note: **<SASROOT>** is a placeholder for the full path specification to the base directory of your SAS 9.3 installation.

Upgrading and Reinstalling SPD Server

Upgrading SPD Server 3.x to SPD Server 4.5

SPD Server 3.x tables are not compatible with SPD Server 4.5. If you want to use your SPD Server 3.x tables with SPD Server 4.5, you must first convert your SPD Server 3.x tables. For more information about converting SPD Server 3.x tables for use with SPD Server 4.5, see [SPD Server 3.x to SPD Server 4.5 Conversion Utility on page 63](#).

Upgrading SPD Server 4.x to SPD Server 4.5

SPD Server 4.x tables are compatible with SPD Server 4.5. No conversion is required to use SPD Server 4.x tables with SPD Server 4.5. You can start SPD Server 4.5 using domains that include tables that were created by any SPD Server 4.x host.

Upgrading or Reinstalling SPD Server 4.5

If you need to reinstall SPD Server 4.5 or you need to install a newer release of SPD Server 4.5, the install process does not alter any files that have been modified in the `\site` directory. Any custom modifications that were made to installation files are retained when you upgrade SPD Server to a subsequent release.

Validate Default Port and Library Assignments

During installation, you can configure the data server port that is used to make connections from the SAS SPD Server libname engine to the SPD Server. You can also configure the SNET Server port that is used to make connections from a JDBC client to the SPD Server.



SAS Deployment Wizard

Specify SAS SPD Ports

Specify the port numbers for the SAS SPD Server connections.

SPD Data Server:
5400

SPD Snet Server:
5401

Help < Back Next > Cancel

For Windows, the specified values are contained in the following **<InstallDir>\site** batch files:

- spdsnsrv.bat: -listenport<data server port>
- spdsserv.bat: -nameserverport<data server port>
- spdssnet.bat: -listenport<snets server port>

To change these values after installation, you must edit these batch files to contain the new values, and then restart SPD Server.

Verify the Name Server Port Assignment and TMP Path

After you install SPD Server on your Windows server, you are almost ready to verify your installation. Before you verify, you must be sure that you have assigned a port for your name server, and you have specified a valid path for the temporary LIBNAME domain called TMP.

Note: SPD Server documentation uses **InstallDir** as a placeholder. Any SPD Server configuration files that you must edit are located in the **InstallDir\site** directory.

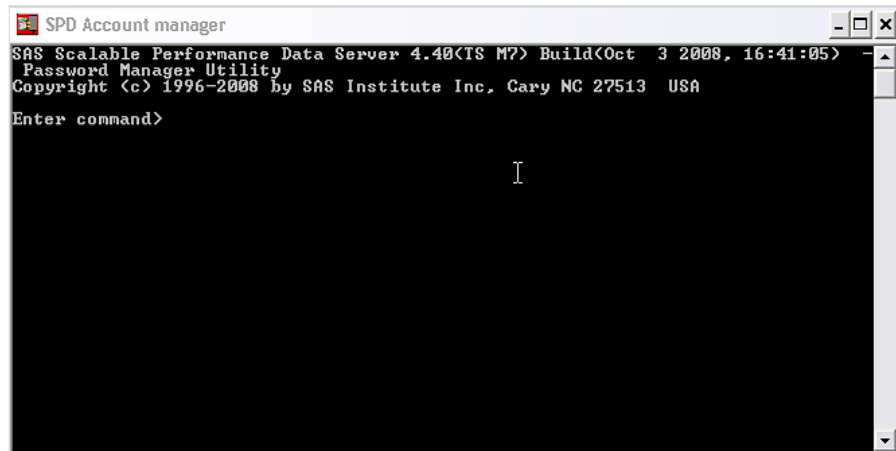
Ensure that your Windows server environment meets the following requirements:

1. During installation, SPD Server creates a sample library parameter file called **libnames.parm** in **InstallDir\samples**. Copy the sample **libnames.parm** file to **InstallDir\site**. The **libnames.parm** file contains a LIBNAME domain definition for a temporary workspace called TMP. The TMP library uses the Windows temporary directory **C:\TEMP**. If your Windows installation does not include a **C:\TEMP** directory, you need to create the directory or specify an existing directory path to replace **C:\TEMP**. If you want to use a different path for TMP, you must modify the SPD Server **libnames.parm** configuration file in **InstallDir\site** to specify the new TMP domain path.
2. During installation, SPD Server creates a sample server parameter file called **spdsserv.parm**, in **InstallDir\samples**. Copy the sample **spdsserv.parm** file to **InstallDir\site**. The **spdsserv.parm** file contains server parameters. You should use the sample **spdsserv.parm** file only to verify that SPD Server is running. You should edit your **spdsserv.parm** file immediately to specify the unique server parameters for your site.

Initialize the Password Manager Database

Before you can start SPD Server, you must initialize the SPD Server Password Manager database from the SPD Account Manager.

1. To open the SPD Account Manager, click the Windows **Start** button and select **Programs ⇒ SAS ⇒ SPD Server 4.5 ⇒ SPD Account Manager**. The SPD Account Manager is launched in a command window.



2. To initialize the SPD Server Password Manager database, add the group ADMINGRP to your site by issuing the following SPD Account Manager commands:

```
Enter Command> groupdef
Enter group name to define> admingrp
Enter Command> quit
```

Install SPD Server as a Service

You must use Windows Services to start SPD Server.

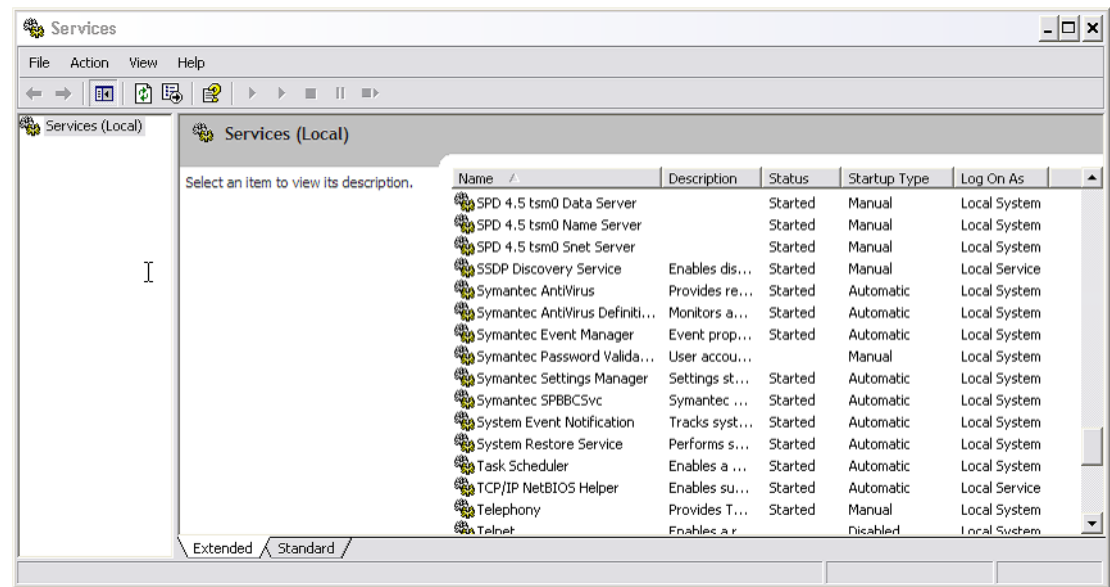
To install the services that start the SPD Server name server, the SPD Server host, and the SPD Server SNET server, select **Start** ⇒ **Programs** ⇒ **SAS** ⇒ **SPD Server 4.5** ⇒ **Install SPD as a Service**.

Select **Start** ⇒ **SAS** ⇒ **SPD Server 4.5** ⇒ **Start SPD Service** to manually start SPD Server.

Select **Start** ⇒ **Programs** ⇒ **SAS** ⇒ **SPD Server 4.5** ⇒ **Stop SPD Service** to manually stop SPD Server.

After you install SPD Server as a service, you can verify SPD Services through Windows Services. To open the Windows Services window, select **Start** ⇒ **Settings** ⇒ **Control Panel** ⇒ **Administrative Tools** ⇒ **Services**.

The main panel of the Services window contains a sortable list of Windows services. Scroll down the **Services** list to find entries for SPD 4.5 name server and the SPD 4.5 data server.



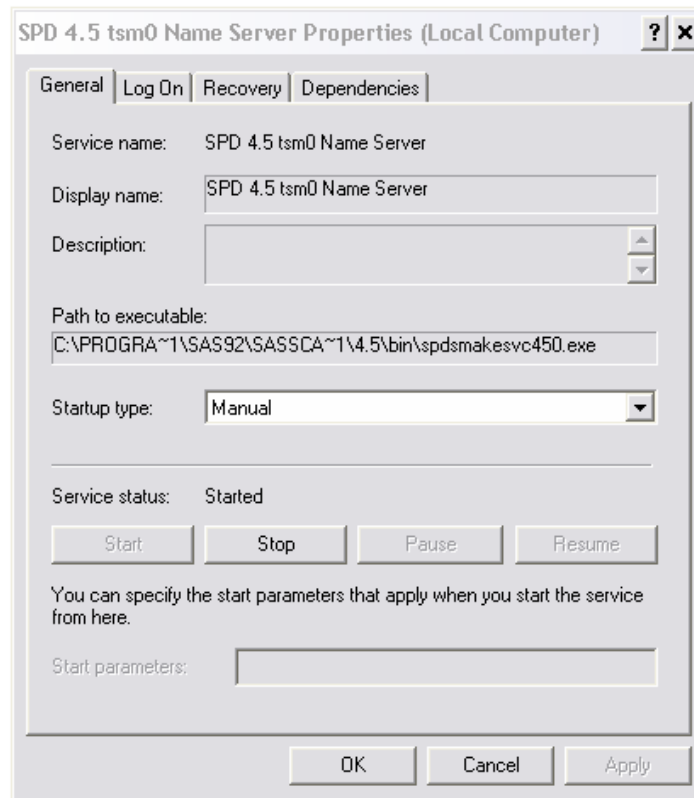
When you configure SPD Server in the Services window for the first time, the **Status** column for the servers is blank and the items in the **Startup Type** column are set to **Manual**.

Start and Stop SPD Server

You must install SPD Server as a service before you can start SPD Server. If you have not already completed this task, see [“Install SPD Server as a Service”](#) on page 46.

Most users want to configure SPD Server to automatically start and stop the name server, data server, and SNET server when they start and stop Windows. The **Automatic** setting loads the name server, data server, and SNET server without prompting. It stops the services without prompting when you close Windows.

1. In the Window Services window, scroll down the **Services** list to find the entry for SPD 4.5 Name Server.
2. Select the name server service in the list, and then right-click on it and select **Properties**. The Properties window opens.



3. Select **Automatic** from the **Startup type** list. This setting configures the 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 **Startup type** setting for the data server and 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** ⇒ **Programs** ⇒ **SAS** ⇒ **SPD Server 4.5** ⇒ **Start SPD Service**. After you manually start the services, the name server, data server, and SNET server will automatically start and stop with Windows.

Configure SPD Server Software on Your Windows Host

After you validate port and library assignments and start the name server and data server, you can begin to configure the LIBNAME domains and user password files.

1. You must configure the libnames.parm file with all of the LIBNAME domains that you use to store SAS tables and catalogs. The best way to exploit the parallel processing capabilities of SPD Server is to declare all of your LIBNAME domains. For more information about the format of the libnames.parm file, see “[SPD Server Command Reference](#)” on page 52 and the libsamp.parm file in **InstallDir**. Any changes that you make to the libnames.parm file when SPD Server is running do not take effect until you stop and restart SPD Server.

2. Add your SPD Server user IDs to the SPD Server password file. You should run the SPD Account Manager to perform this function. You can add to or modify the SPD Server password file at any time, even when SPD Server is running.
3. Use the SPD Account Manager to add users and groups. For more information, see “Using SAS Scalable Performance Data (SPD) Server with Other Clients” in Chapter 6 of *SAS Scalable Performance Data Server: User's Guide*.
4. After you set up and configure your SPD Server host environment, examine the files in your **InstallDir** directory. This file contains SAS programs that help you understand how to use various SPD Server features. The following sample files are included:

auditraw.sas	symsub.sas
auditwh.sas	fmtgrphy.sas
audit.sql.sas	dynamic_cluster*.sas
verify.sas	minmax*.sas
passthru.sas	paralleljoin*.sas
tempwork.sas	starjoin*.sas
paraload.sas	index_scan*.sas
accolrw.sas	materialize_view*.sas

For more information about these files, see [“Packing List for SPD Server Distribution” on page 40](#).

Install and Configure SPD Server Clients

SPD Server client software makes SAS LIBNAME connections and performs user-specific functions on the SPD Server host. SPD Server client software is installed with SAS 9.3. The software contains the following SAS modules:

- sasspds is the LIBNAME engine that is required to access the SPD Server environment from SAS 9.3.
- sasspdo is the SPD Server operator procedure that is required to access the SPD Server 4.5 environment from SAS 9.3.
- spds.msg is the message file for the SPD Server LIBNAME engine and SPDO operator procedure. This file is compatible with SAS.

You install the SPD Server client software with SAS 9.3 Foundation at **<SASROOT>\SASFoundation\9.3\spdclient**. The SAS 9.3 configuration file automatically includes your SPD Server client software directory in its required path list.

After you have configured the SPD Server environment and SPD Server is running, you need to complete other installation tasks on the SAS clients that will use SPD Server. Because some of these tasks might involve the system that is actually running SPD Server, you might have already completed some of the following steps when you installed the SPD Server host. If that is the case, skip the duplicated steps.

SPD Server media contains SAS client software modules for SAS 9.3 installations on Solaris by Sun, AIX by IBM, and HP-UX by Hewlett-Packard. Complete the following steps on each SAS client that will access SPD Server:

1. If you want to access SPD Server through a registered port (name service), add the following service to your client's `\etc\services` file (if this file is not already present):

```
spdsname ????\tcp # SPDS Name service
```

This service defines the port number for the SPD Server name server process (spdsnsrv). Make sure that this port number matches the port number 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.

2. The SPD Server can be accessed with the SAS 9.3 ODBC Driver, JDBC Driver, and htmSQL Driver. Download these drivers from the **Support** tab at <http://support.sas.com>.

For more information about connecting and configuring these applications, see “Using SPD Server with Other Clients” on page 97.

3. ODBC drivers only: ODBC client applications require you to install the spds.dll application extension. To install the ODBC client application extension:
 - a. Install the SAS 9.3 ODBC driver.
 - b. Copy the `InstallDir\lib\spds.dll` directory to the `<drive letter>:\Program Files\sas\shared files\general` directory.
 - c. Configure an ODBC data source for direct SPD Server access.

Use SAS to Test Your SPD Server Installation

Testing your SPD Server installation is simple. To verify your installation, you make two SAS LIBNAME assignments using the SPD Server LIBNAME engine. The examples in this section refer to the SASSPDS engine, which is the engine for SAS 9.3.

1. Start the SPD Server environment.
2. On a client system that is configured correctly, invoke SAS and make the following LIBNAME assignments:

```
LIBNAME test sasspds 'tmp'
server=serverNode.port
user='anonymous' ;
```

In this code, `serverNode` is the server's node name, and `port` is either the numeric value used to start the name server or the service name you use to access the SPD Server name server. Name services enable you to connect to a server using a character string, instead of specifying a port number. If you used the default numeric port assignment of 5400, your assignment looks similar to the following example:

```
LIBNAME test sasspds 'tmp'
server=serverNode.5400
user='anonymous' ;
```

If you used the `spdsname` service, your LIBNAME assignment looks similar to the following example:


```
LIBNAME test sasspds 'tmp'
  server=serverNode.spdsname
  user='anonymous';
```

In addition, you should verify that the row-level integrity LIBNAME assignment works correctly:

```
LIBNAME testrl sasspds 'tmp'
  server=serverNode.port
  user='anonymous'
  locking=YES;
```

When you verify these statements, you confirm the connectivity between the SAS client and the SPD Server environment. When you successfully make these LIBNAME assignments, the network configuration is correct and most of the SPD Server configuration is correct.

Substitute serverNode with the node name that runs the name server and the data server. The test assumes that the temporary LIBNAME definition in the sample libnames.parm file was not changed during installation.

3. Watch the SAS log for error messages. You might see failures to locate one or more of the required SPD Server components. The following messages are common error messages that you might receive:

```
ERROR: Module TEST not found in search paths.
ERROR: Error in the LIBNAME or FILENAME statement.
```

If you receive the following error message (or a similar message) that describes failures to access messages, check your -SASMSG option and confirm that the directory in which you installed SPD Server components is correctly set.

```
ERROR: unable to access message 608.108
```

4. After you have assigned every LIBNAME, you can further verify your installation by running the sample SAS program, **InstallDir\samples\verify.sas**. Issue the following SAS command to execute the program:

```
%include 'InstallDir\samples\verify.sas'\source2;
```

This test exercises many features of the SPD Server LIBNAME engine and proxy, and verifies your installation configuration. The test performs a sequence of DATA and PROC steps using a generated data set. It checks the results expected from various DATA step queries. If any query fails to produce the expected results, the SAS job is terminated. The job verify.sas requires that the SAS librefs TEST and TESTRL are assigned as shown in [step 2 on page 50](#).

5. Verify that SQL pass-through services are working in SPD Server by issuing the following SAS commands:

```
%let spdshost=serverNode;
%let spdsport=port;
%include 'InstallDir\samples\verptsq1.sas'\source2;
```

Note: serverNode and port are described in [step 2 on page 50](#).

SPD Server Command Reference

SPD Server operation revolves around the executables that are described in the packing list. The executables are in the bin\ subdirectories. Each executable supports a set of command-line options that override default features, or provides site-dependent configuration information. The command-line options for each executable are provided in the following sections.

SPD Server Name Server Commands

The SPD Server LIBNAME engine connects to the SPD Server name server. The name server resolves LIBNAME domain names into physical file system paths for librefs. The name server also resolves host node and end-point (TCP port) addresses for each LIBNAME. Each SPD Server (spdssrv) host process registers LIBNAME domain information from its configuration file with its appointed name server (spdsnsrv) process. Multiple SPD Server hosts can use the same name server to register their LIBNAME domains. The only requirement is that the combination of the LIBNAME= option values from the SPD Server host's LIBNAME configuration file must be unique across all SPD Server hosts that connect to the name server.

Invoke the SPD Server name server by using the following command-line syntax:

```
spdsnsrv [-option [optval]...]
```

The **spdsnsrv** command supports the following options:

- listenport *port#*
specifies the explicit TCP port number that the name server uses to accept connections from the SPD Server LIBNAME engine and its SPD Server hosts. If no port is specified, the name server queries the system for port addresses using the service name spdsname. If no such service has been registered, SPD Server chooses a dynamic port number for the name server to use.
- 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 **\DOWNlogs\spdsnsrv**, the name **\DOWNlogs\spdsnsrv_mmddyyyy_hh:mm:ss.spdslog** is generated. The values **mmddyyyy** and **hh:mm:ss** indicate the time when the system created the log file.
- logtime *hh:mm*
specifies the time of day to cycle a new generation of the name log file. At this time each day, the previous log file will be closed and a new log file will be opened.
- licensefile *lic-file*
License file keys are generated by SAS and provided to you. With this release of SPD Server, you receive an SPD Server license key for each machine on which you license SPD Server. The SPD Server administrator must enter each key into the license filer. SPD Server will not run on a machine whose valid license key has not been entered into the license file. License keys are plain text strings that include product, site, and machine information, and the password that is required for using the SPD Server in this specific environment.

SPD Server Host Commands

The SPD Server LIBNAME engine connects to the SPD Server host to access data in the server environment. The SPD Server host uses the SPD Server password file to validate each SPD Server user, and then creates a LIBNAME proxy process on behalf of each of them.

Part of the function of the SPD Server host process is to start SPD Server logging processes. The `spdlog` process performs message logging functions. The `spdsaud` process performs audit logging functions. The `spdsserv` command-line options control automatic log filename generation and cycling properties. The `spdsserv` command-line options also control the message and audit logging functions. Message and audit logging facilities include automatic log filename generation and periodic log file cycling support. You can periodically switch to a new name server log file without stopping and restarting SPD Server, which improves the availability of the name server.

Audit log records are kept for all resources that are accessed by each LIBNAME proxy process. The audit log saves records in its own separate space, away from other server log files. A sample SAS job that processes the audit log and generates a report is provided. Check `samples\audit.sas` for information about processing the audit log and generating the report. To enable the audit log, use the **spdsserv** command with the `-AUDITFILE` option.

When you are using automatic server log cycling or audit log cycling, remember to periodically clean the log files. Proper log file maintenance includes archiving logs using secondary or long-term storage.

Many users retain only a few generations of log files for quick reference. A shell script that runs on a regular basis (such as CRONTAB) is a good way to perform log maintenance on your server machine.

Invoke the SPD Server host by using the following command-line syntax:

```
spdsserv [-option [optval]...]
```

The **spdsserv** command supports the following options:

-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 a parameter file named `spdsserv.parm` is in the process's current working directory. Option declarations in this file are of the following form:

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

You probably do not need to modify the default values in `InstallDir\site\spdsserv.parm`. For more information about the parameter file and the server parameters, see [“Setting Up SPD Server Parameter Files”](#) on page 113.

-acl *dir-pwd-dir-path*

specifies the directory path to the SPD Server host password file. You can omit this option if the `PASSPATH=` option is declared in the SPD Server host's `-PARMFILE` option. A valid SPD Server password file is required even when you specify the `-noacl` option. You must use the SPD Account Manager to create the password file and to populate it with the set of valid SPD Server user IDs.

-noacl

disables SPD Server login validation for SPD Server LIBNAME engine connections to the SPD Server host.

- nameserver *node-name*
specifies the node name where the name server process is running. This node does not need to be the same node that is hosting the SPD Server host processes. This option is required.
- nameserverport *port#*
specifies an explicit TCP port number for the SPD Server host to use to connect to its name server. If you do not specify a port, the name server queries the system for a registered port address that is using the service name spdsname.
- libnamefile *file-spec*
specifies the name of the file that contains the logical LIBNAME domain definitions that the SPD Server host supports. LIBNAME definitions can span multiple lines and must begin with the LIBNAME=*name* keyword. Each LIBNAME definition must be terminated with a ; character.
- logfile *fileSpec*
specifies that the logger process will automatically create a server log file. The *fileSpec* value specifies a partial pathname or filename that is used to generate the complete log file path. For example, if you specify *fileSpec* as \DOWNlogs\spds, the generated name will be: \DOWNlogs\spds_ *mmddyyyy_hh:mm:ss.spdslog*. In this example, *mmddyyyy* and *hh:mm:ss* indicate the system time when the log file was created.
- logtime *hh:mm*
specifies the time of day to cycle a new generation of the server log file. At this time each day, the previous log file will be closed and a new log file will be opened.
- auditfile *fileSpec*
enables audit logging for the server and automatic audit log file creation by the audit process. The *fileSpec* value specifies a partial pathname 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 *mmddyyyy_yyyy.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. At this time each day, the previous log file will be closed and a new log file will be opened.

SNET Server Commands

The SNET Server is the connection point for clients accessing SPD Server data through ODBC, JDBC, or htmSQL applications.

Invoke the SNET server by using the following command-line syntax:

```
spdssnet [-listenport listen_port]
```

The **spdssnet** command supports the following options:

- listenport *listen_port*
specifies the listen port number that spdssnet uses to accept connections from ODBC, JDBC, or htmSQL clients. If you do not specify a port number, spdssnet uses the name service spdssnet from the **etc\services** file to determine its listen port.
- logfile *fileSpec*
specifies that the logger process will automatically create a server log file. The *fileSpec* value specifies a partial path or filename that is used to generate the complete log file path. For example, if you specify *fileSpec* as \DOWNlogs

`\spdssnet`, the generated name will be
`\DOWNlogs\spdssnet_mmddyyyy_hh:mm:ss.spdslog`, where `mmddyyyy`
 and `hh:mm:ss` indicate the system time when the log file was created.

`-logtime hh:mm`

specifies the time of day to cycle a new generation of the SNET log file. At this time each day, the previous log file will be closed and a new log file will be opened.

PSMGR Utility

The SPD Server administrator can use the `psmgr` utility to create and maintain the data set that contains the authorized SPD Server user IDs. If you choose to run SPD Server ACL support, you need to use this utility to create and populate the SPD Server password file before you start SPD Server. You can use the `psmgr` utility or SAS Management Console to manage passwords. For more information about the `psmgr` utility, see [“The Password Manager Utility `psmgr`” on page 191](#).

SPD Server and the SAS Management Console

The SAS Management Console offers a standardized, common management tool that enables enterprises to support many technologies from a single point of administration. This client application uses an extensible plug-in architecture that enables you to customize the console to support a wide range of administrative capabilities. When you install SAS 9.3 and SPD Server 4.5, you are prompted to install the SAS Management Console utility. For more information about SAS Management Console, see [“Overview of SAS Management Console” on page 75](#).

SPD Server Lightweight Directory Access Protocol (LDAP) Authentication

In SPD Server 4.5, clients can be authenticated by `psmgr` or by an LDAP server that is running on the SPD Server host. LDAP authentication integrates with the SPD Server password facility and offers a centralized approach to user ID and password management. SPD Server clients that use LDAP authentication should have user accounts in the domain in which the LDAP and SPD servers are running. 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's password database through `psmgr` or the SAS Management Console utility so that all other SPD Server user information is recorded.

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

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, the 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.

Note: If you use the LDAP_HOST value for LDAPSERVER, SPD Server might fail during start-up.

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 0-65536. The default setting is the LDAP_PORT value.

LDAPBINDDN

the distinguished name (DN), or the location in the LDAP Server's database where the client's information is stored. LDAPBINDDN is a Lightweight Directory Access Protocol term. LDAPBINDDN is the combination of the user name and the network domain in which they operate. The form of this string is "**ID= , rdn1=RDN1, rdn2=RDN2, . . .**", where **ID** is the identifier of 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**", and 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 **uid**. 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 distinguished name 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 it using LDAPBINDDN.

See the LDAP server administrator at 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.

SPD Server User IDs

The SPD Server system uses its own layer of access controls that overlay the file-system 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.

The SPD Server password file gives you better access control to SPD Server's data resources than a native Windows user ID does. Many sites do not want to give Windows accounts to SPD Server system users, but they still want protection and ownership of the data resources that are created in the SPD Server environment. In this case, SPD Server user IDs provide an extra layer of access control.

You need to be familiar with the SPD Account Manager.

If you do not use SPD Server user IDs, you need the SPD Server password file. Without the SPD Server password file, the data server process does not function correctly. To disable the use of SPD Server user IDs at your site, specify the `-NOACL` option when you start the data server process.

If you use SPD Server user IDs, add them to the SPD Server password file that was created during installation. The SPD Account Manager reads its commands from STDIN so you can pipe commands to it from another command, script, or input file.

Troubleshooting

Troubleshooting networked applications can be difficult. Key information for SPD Server troubleshooting can be found in the Name Server and SPD Server host process log files. With those two log files, you can reconstruct SAS interaction with SPD Server components. Entries in these 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 non-standard 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 will be able to diagnose the reason for the failure from these messages. Some common problems include:

- The LIBNAME engine selector that is specified in the LIBNAME statement is invalid. The SPD Server engine name is sasspds and is misspelled in the following LIBNAME statement:

```
LIBNAME foo sasspds 'test'
      server=sunspot.spdsname
      passwd='xxx';
ERROR: Module FOO not found in search paths.
ERROR: Error in the LIBNAME or FILENAME statement.
```

- The logical LIBNAME domain name that is specified in the LIBNAME statement is invalid. The domain name **test** is not defined in the SPD Server name server sunspot.spdsname:

```
LIBNAME foo sasspds 'test'
      server=sunspot.spdsname
      passwd='xxx';
ERROR: ERROR: Libname path info not found in SPDS name server.
ERROR: Error in the LIBNAME or FILENAME statement.
```

- No name server is running on the specified node name, or no name server is available at the specified port address. In the following example, no name server is running on the specified node **stelling**. This same message is generated if the port address is incorrect:

```
LIBNAME foo sasspds 'test'
      server=stelling.spdsname
      passwd='xxx';
ERROR: Unable to connect to SPDS name server.
ERROR: Connection refused.
ERROR: Error in the LIBNAME or FILENAME statement.
```

- An invalid or unknown node name is specified in the LIBNAME statement. In the following example, node **xxx** is not accessible in the network.

```
LIBNAME foo sasspds 'test'
      server=xxx.spdsname
      passwd='xxx';
ERROR: Unable to connect to SPDS name server.
ERROR: xxx.
ERROR: Error in the LIBNAME or FILENAME statement.
```

- An invalid SPD Server user password is specified in the LIBNAME statement. In the following example, the SPD Server user ID is derived from the UNIX user ID that is running the SAS session. The SPD Server password file has an entry for this SPD Server user ID, but the password is not **xxx**:


```
LIBNAME foo sasspds 'test'
  server=sunspot.spdsname
  passwd='xxx';
ERROR: Error on server LIBNAME socket.
ERROR: SPD server has rejected login from user
sasetb.. ERROR: Error in the LIBNAME or FILENAME
statement.
```

- An invalid SPD Server user ID is specified in the LIBNAME statement. In the following example, the SPD Server user ID **xxx** does not exist in the SPD Server host's password file. The resulting message is the same if the password is invalid.

```
LIBNAME foo sasspds 'test'
  server=sunspot.spdsname
  user='xxx'
  passwd='xxx';
ERROR: Error on server LIBNAME socket.
ERROR: SPD server has rejected login from user xxx..
ERROR: Error in the LIBNAME or FILENAME statement.
```

Problems 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.

Note: Do not change the licensing information unless you are logged in with the user ID of the owner of the SPD Server. You designate the owner of SPD Server when you license the software.

Part 3

Migration

Chapter 5

SPD Server 3.x to SPD Server 4.5 Conversion Utility [63](#)

Chapter 5

SPD Server 3.x to SPD Server 4.5 Conversion Utility

Introduction	63
Before You Convert	63
Overview of the SPDS CONV Utility	64
Using SPDS CONV	65
SPDS CONV Utility Examples	66
Example LIBNAME Parameter File	66
Converting a Simple Table	66
Converting Tables and Re-creating Indexes	67

Introduction

SPD Server 4.5 uses improved 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 backwards compatible with SPD Server 3.x software.

SPD Server 4.x tables use architectures that use SAS 9 metadata. SPD Server 3.x tables use architectures that use SAS 8 metadata. These two metadata architectures are not compatible. However, SPD Server 4.5 provides a conversion utility SPDS CONV that SPD Server 3.x customers can use to convert existing tables for use with SPD Server 4.5. The SPDS CONV utility is designed to be run by the SPD Server Administrator.

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

Before You Convert

The SPDS CONV utility changes the format of SPD Server 3.x tables to the format used by SPD Server 4.5 tables. The conversion is not reversible. After you convert a table for use with SPD Server 4.5, 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 4.5, you can use PROC COPY to make copies of the tables before you convert them. You can use SPDS CONV on the copied versions to convert them for use with SPD Server 4.5 and maintain archival versions of the SPD Server 3.x tables.

Overview of the SPDS CONV Utility

The SPDS CONV utility converts SPD Server 3.x metadata files for use with SPD Server 4.5. The conversion process updates the physical structure of the metadata files and renames them. The SPDS CONV 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 4.5 table files end with the filename extension .spds9. All tables that are upgraded to be compatible with SPD Server 4.5 have the filename extension .spds9.

SPD Server 4.5 index files differ from SPD Server 3.x index files. SPD Server 4.5 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 environment.

The SPDS CONV table conversion utility does not re-create index files. When you use the SPDS CONV utility to convert tables from SPD Server 3.x to SPD Server 4.5 format, the utility automatically deletes physical files that were associated with the old 3.x indexes and that are now obsolete. The SPDS CONV utility does offer an option to create a SAS job file that you can run in the SPD Server 4.5 environment to recreate the SPD Server 3.x index files for use with SPD Server 4.5.

If you choose to create the SAS job file to re-create SPD Server 3.x indexes for use in SPD Server 4.5, the code resembles the following example:

```
%let SPDSIASY=YES;
PROC DATASETS lib=<spds4 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 SPDS CONV 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 4.5 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 4.5 to use tables that contain more than 2,000,000 rows of data.

How does SPDS CONV work? When the SPDS CONV utility converts a table, it reads the original SPD Server 3.x metadata file and creates a new SPD Server 4.5 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 4.5 metadata file is deleted, and the original SPD Server 3.x table remains intact.

SPDS CONV reads the SPD Server 3.x metadata file one section at a time, and re-creates each structure in the SPD Server 4.5 metadata file as it is read. After the SPD Server 4.5 metadata file is fully populated, the utility checks the data partition file component to determine whether updates are required.

If SPDS CONV detects the presence of compression block headers, then the data partition file contains SAS 8 compression information that is not compatible in SPD Server 4.5, and SPDS CONV must update the data partition files. SPDS CONV updates the file by overwriting the compression block headers. SPDS CONV 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 SPDS CONV 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 4.5, all remnants of the SPD Server 3.x table are deleted. The SPDS CONV 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 4.5 table retains the same SPD Server owner as the SPD Server 3.x table.

Using SPDS CONV

The SPDS CONV 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, and then you specify the options that you want for your conversion. If your SPD Server software is installed on a UNIX platform, see the [Chapter 3, “SPD Server UNIX Installation Guide,” on page 13](#) for information about setting up your environment before you run SPDS CONV.

The command-line syntax is as follows:

```
SPDS CONV <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 SPDS CONV 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 SPDS CONV 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 4.5.
- 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 4.5 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 Server 4.5. 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 4.5 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 4.5. 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
```


Part 4

Configuration

<i>Chapter 6</i>	
Using the SPD Server Name Server to Manage Resources	<i>71</i>
<i>Chapter 7</i>	
Administering and Configuring SPD Server Using the SAS Management Console	<i>75</i>
<i>Chapter 8</i>	
SPD Server SQL Query Rewrite Facility	<i>93</i>
<i>Chapter 9</i>	
Using SPD Server with Other Clients	<i>97</i>
<i>Chapter 10</i>	
Configuring Disk Storage for SPD Server	<i>109</i>
<i>Chapter 11</i>	
Setting Up SPD Server Parameter Files	<i>113</i>
<i>Chapter 12</i>	
Setting Up SPD Server Libname Parameter Files	<i>125</i>
<i>Chapter 13</i>	
Setting Up SPD Server Performance Server	<i>143</i>

Chapter 6

Using the SPD Server Name Server to Manage Resources

Managing Computing Resources with a Name Server	71
Configuring SPD Server on a Corporate Network	71
Example Scenario	71
Run the Name Server on the namecpu Machine	72
Configure SPD Server on the worldcpu server	72
Set Up asiacpu, the Asia Departmental Server	72
Which SAS Program Statement Runs Where?	73

Managing 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 SPD Server Using the SAS Management Console

Overview of SAS Management Console	75
Accessing SPD Server Services in SAS Management Console	76
Connect to an SPD Server	76
Password Manager	78
Overview of Password Manager	78
Managing Users with Password Manager	78
Managing Groups with Password Manager	81
ACL Manager	82
Overview of ACL Manager	82
List ACL Resources	82
Add an ACL Resource	84
Delete an ACL Resource	85
Add a User or Group to an ACL Resource	85
Change Resource Permissions	85
Server Manager	85
Overview of Server Manager	85
Refresh Domains	86
Refresh Configuration Information	87
Run Commands	87
Process Profiler	88
Proxy Manager	90
Overview of Proxy Manager	90
Refresh Proxies	91
Interrupt Proxies	91
Cancel Proxies	91

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 SAS Management Console's single 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 4.5 is installed with SAS 9.3 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 76](#).

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.

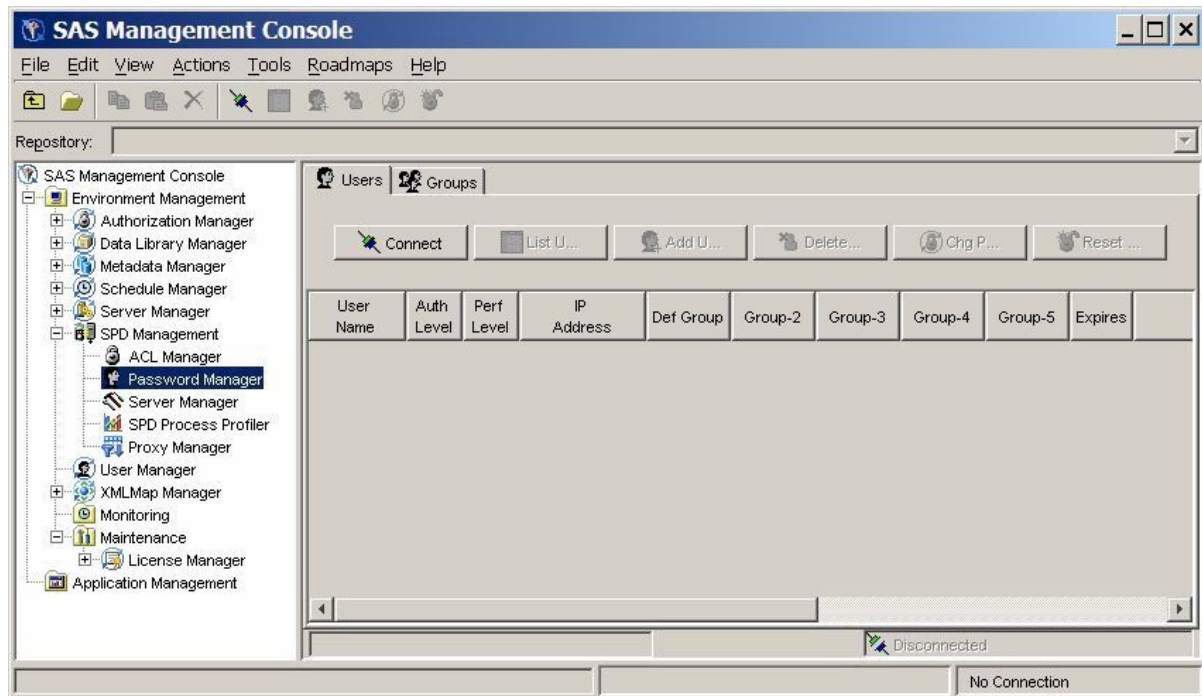
Note: Do not confuse the SAS Management Console status indicator in the lower right corner of the SAS Management Console window with the SPD Server connection status at the bottom of the active tab.

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:

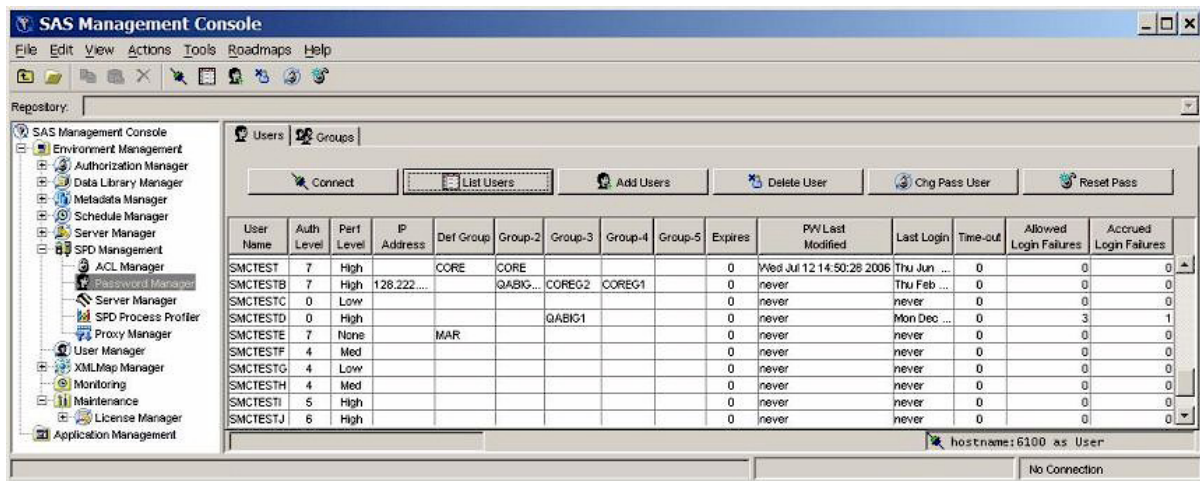


When no server connection exists, no data is displayed. To display data, you must connect to an SPD Server. For more information, see [“Connect to an SPD Server”](#) on page 76.

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 80](#) and [“Add a User” on page 80](#).

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 114](#).

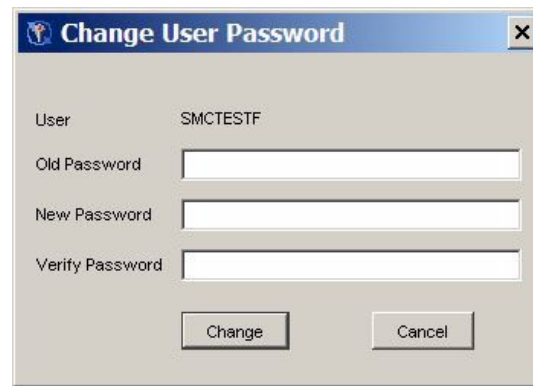
User, **Password**, **Auth Level**, **Def Group**, and **Failed Logins** are required values. The **IP Address**, **PW Expire**, and **Login Timeout** values are optional. If you omit the optional settings, they default to "no limits".

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

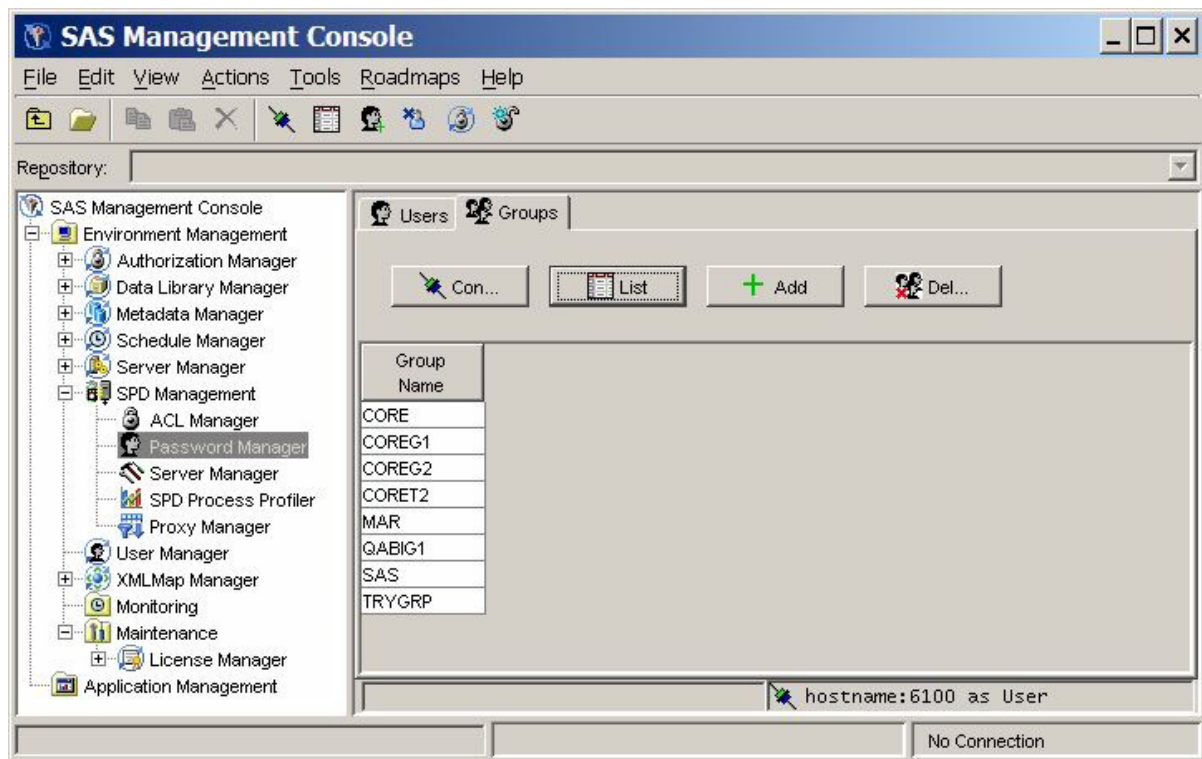
To reset the password for a selected user, click **Reset Pass**. Specify the user's new password and click **Change**. You do not need to know the user's current password to change it. The user must change the password before he can connect to the server.

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.

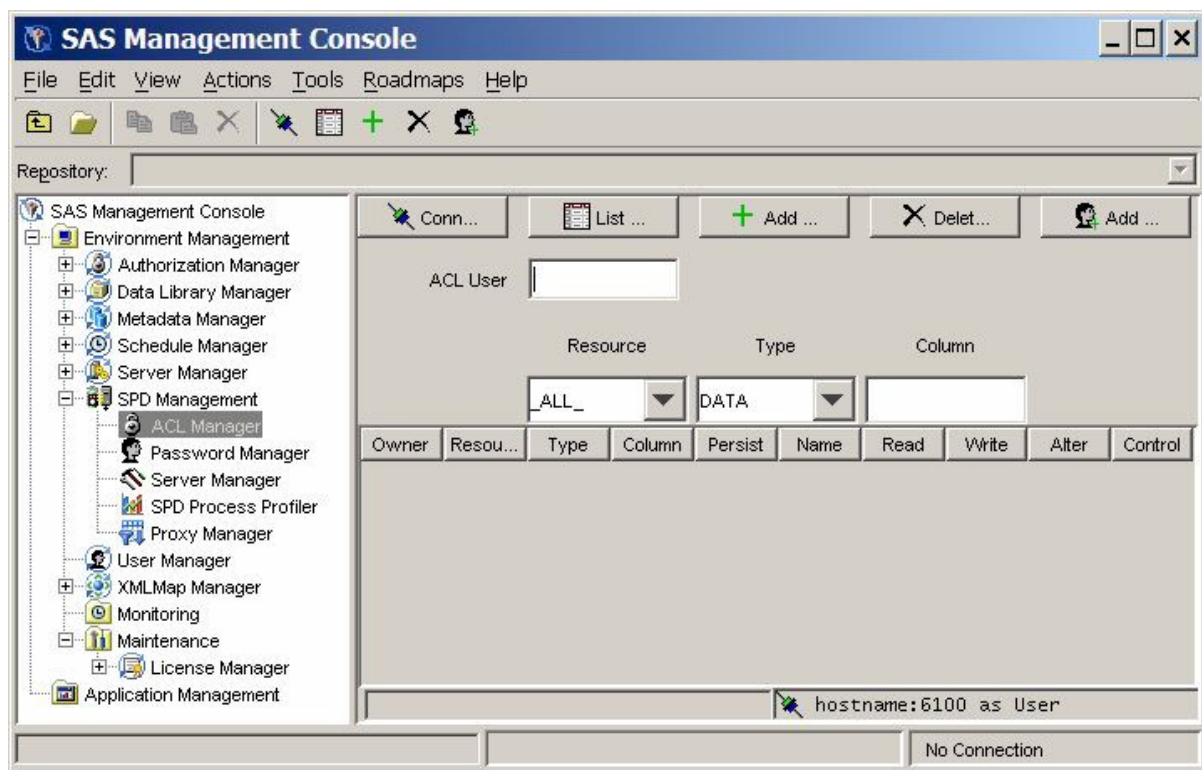
Delete a Group

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

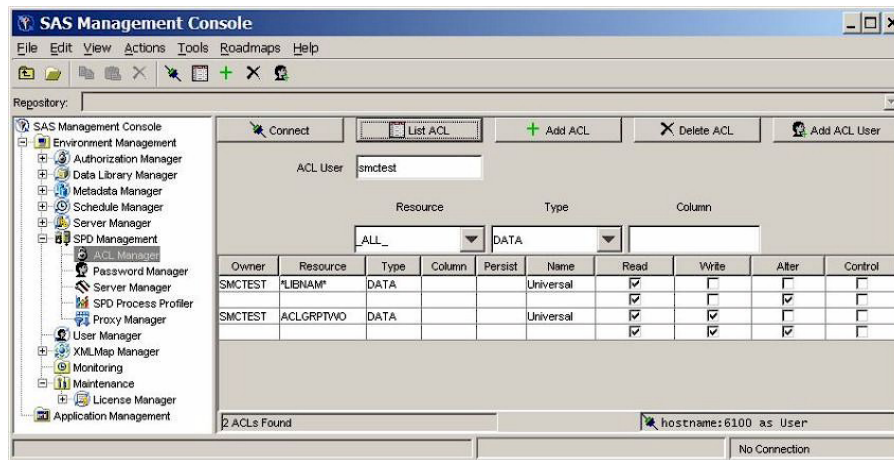
Click the ACL Manager folder in the SAS Management Console window to manage ACL resources. The ACL Manager panel resembles the following display.



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 76](#).

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 MDDDB). 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.

	R	W	A	C
Default	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Group	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Specify the following values in the Add ACL window.

Resource

name of the resource you are adding..

Column

column restrictions for the resource 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

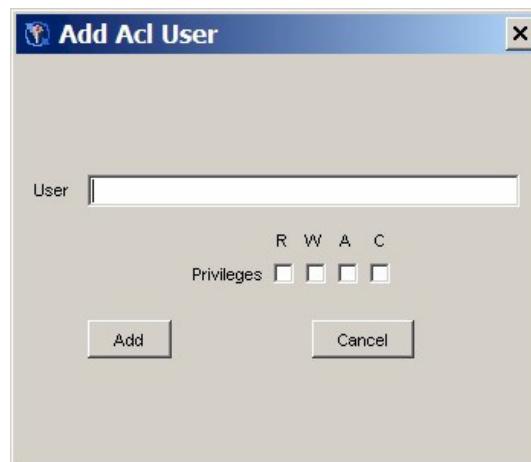
specify the name of an existing ACL resource to use as a model for this ACL resource.

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, and then 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

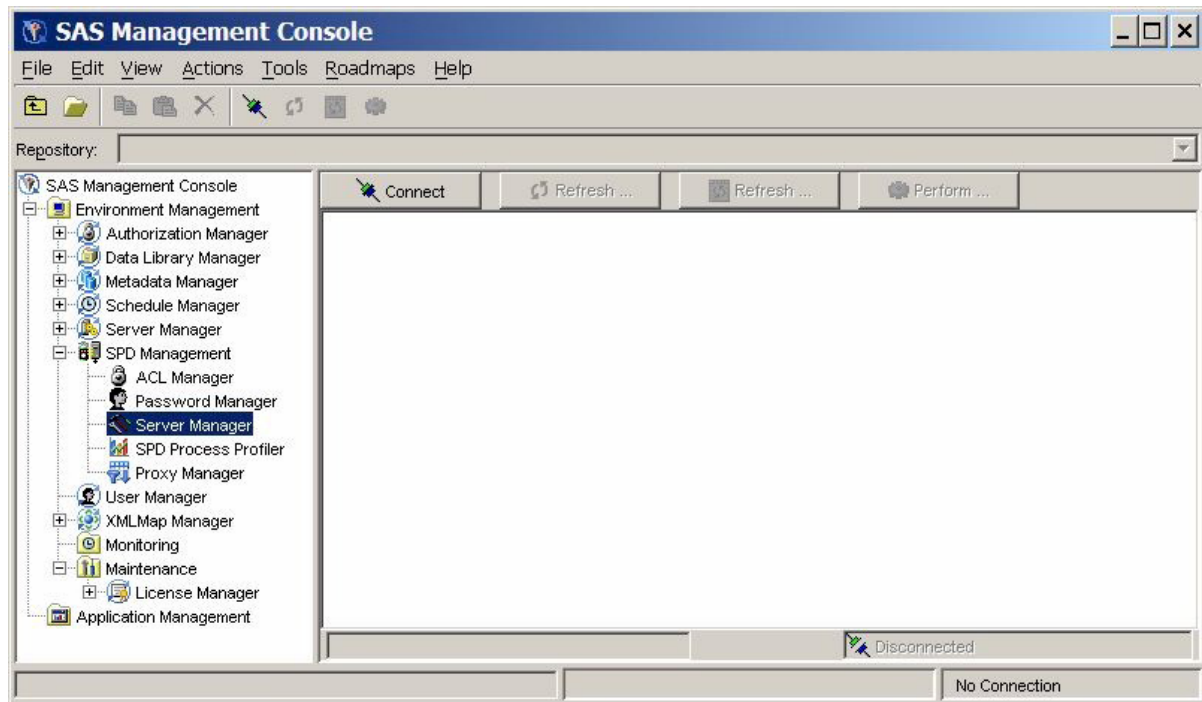
Each ACL resource has at least one set of permissions called universal permissions. Universal permissions are the default permissions for the ACL resource if no other permissions are applied. If any group names or user names have permissions for the ACL resource, they are displayed.

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

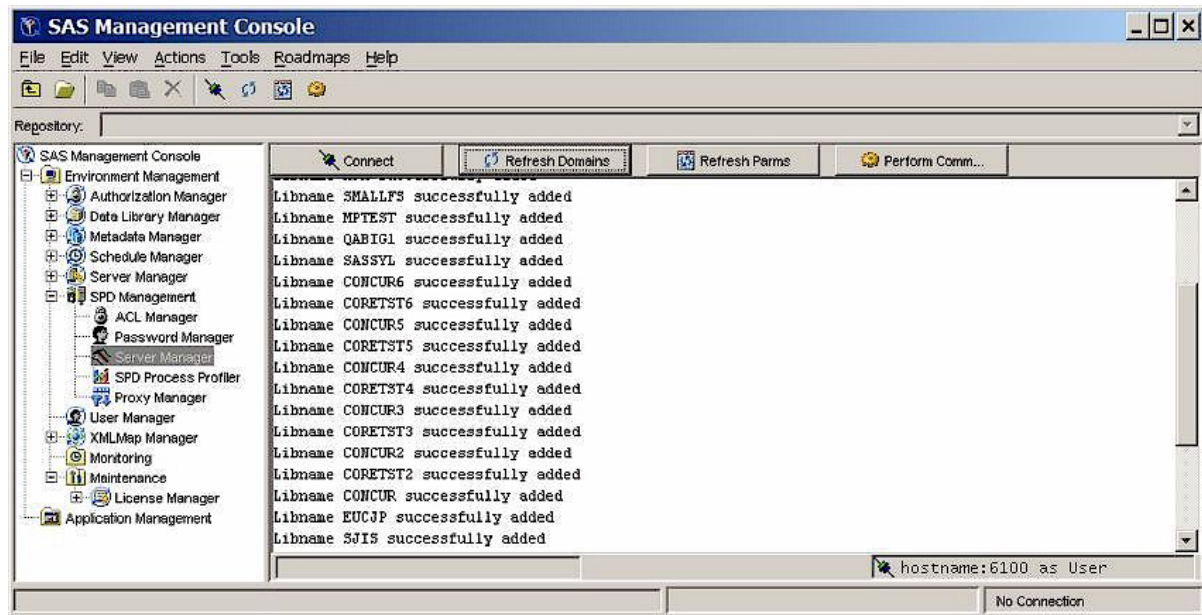
Click the **Server Manager** folder in the SAS Management Console window to refresh the SPD Server configuration and to run selected SPD Server utilities. The Server Manager panel resembles the following display when no server connection exists.



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 76.

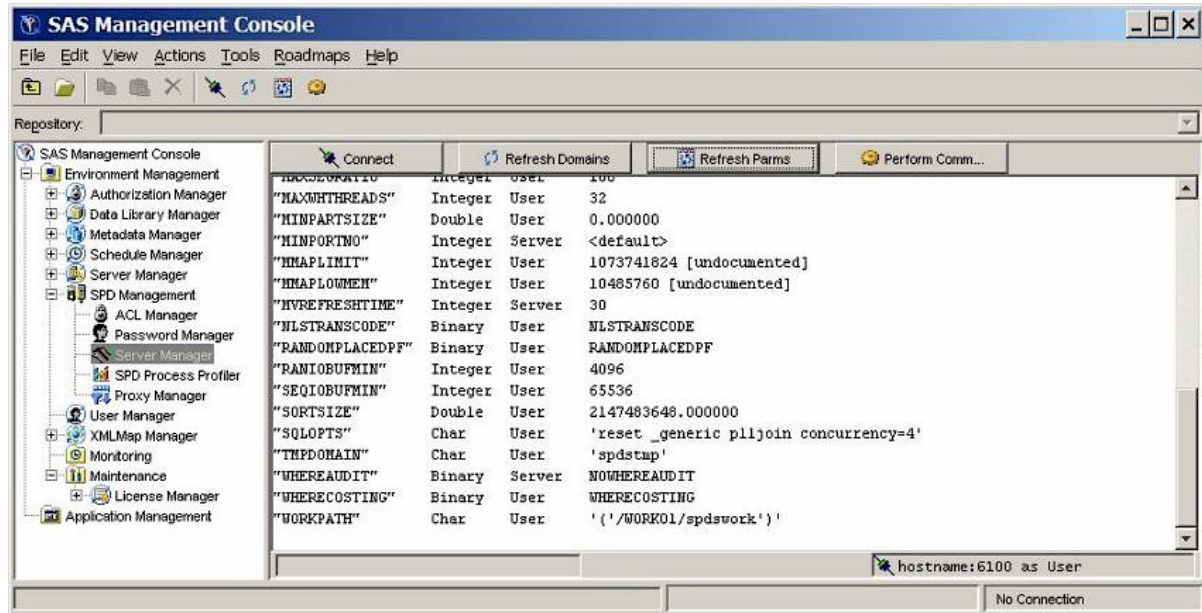
Refresh Domains

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.



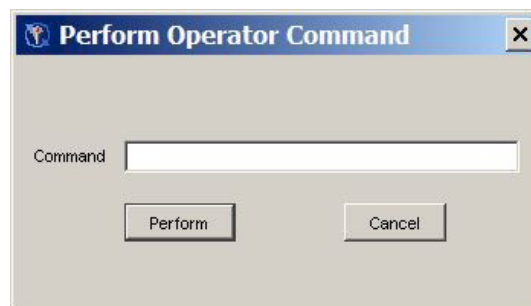
Refresh Configuration Information

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

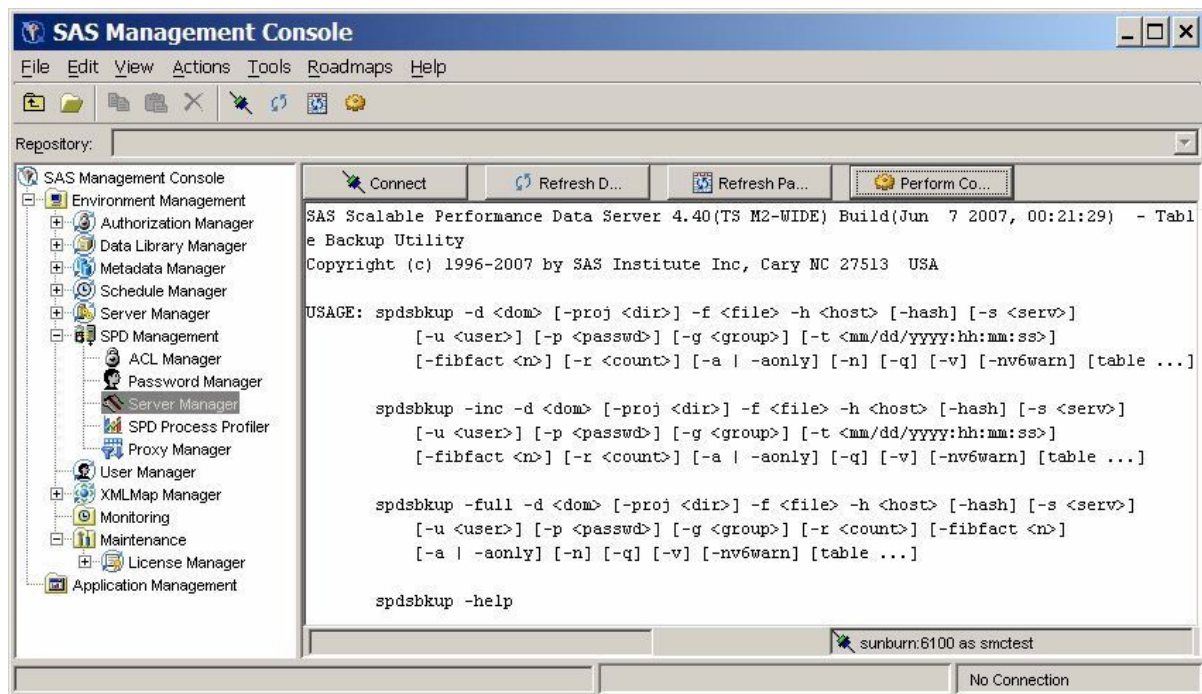


Run Commands

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**.



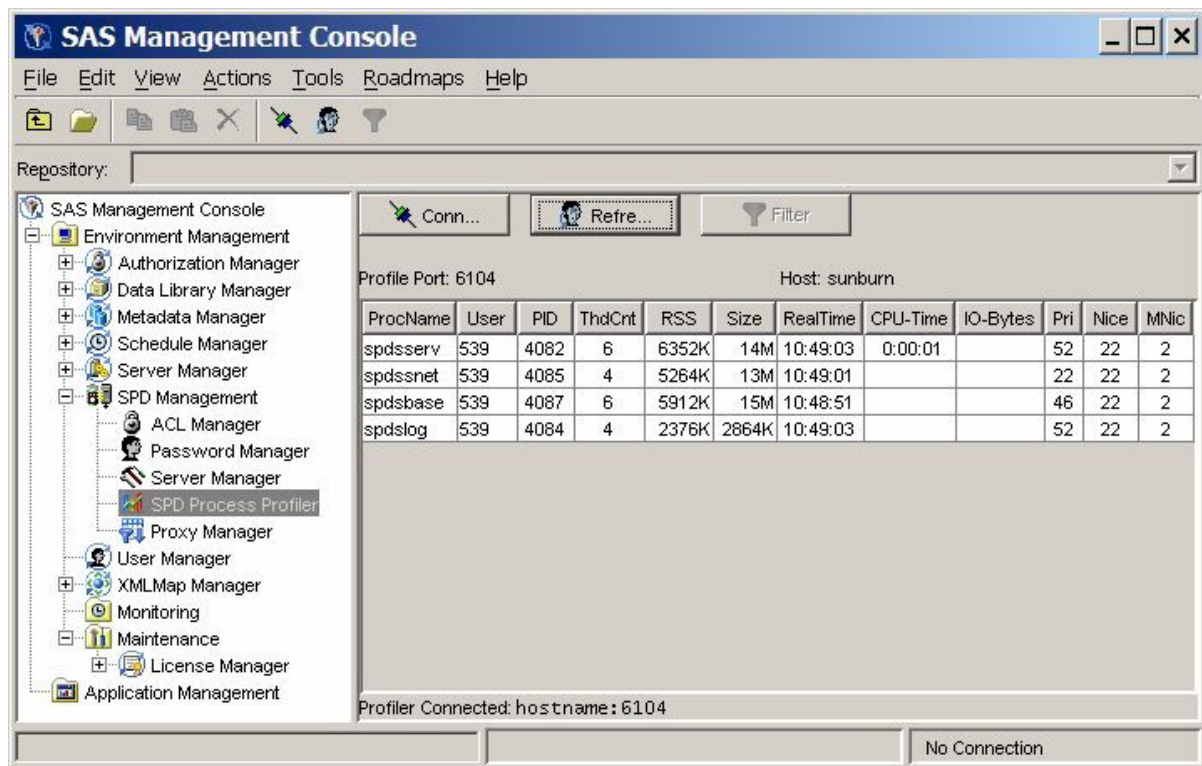
The following example shows the Server Manager panel after the `spdsbkup` command was issued:



Process Profiler

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

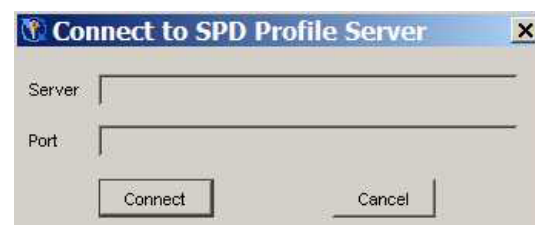
Note: The SPD Server Process Profile Manager is not supported on Windows or on Linux on X64.



The Performance Server gathers SPD Server process performance information and distributes it across the SPD Management section of the 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 `spdperf` 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 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 SPD Server Performance Server,”](#) on page 143.

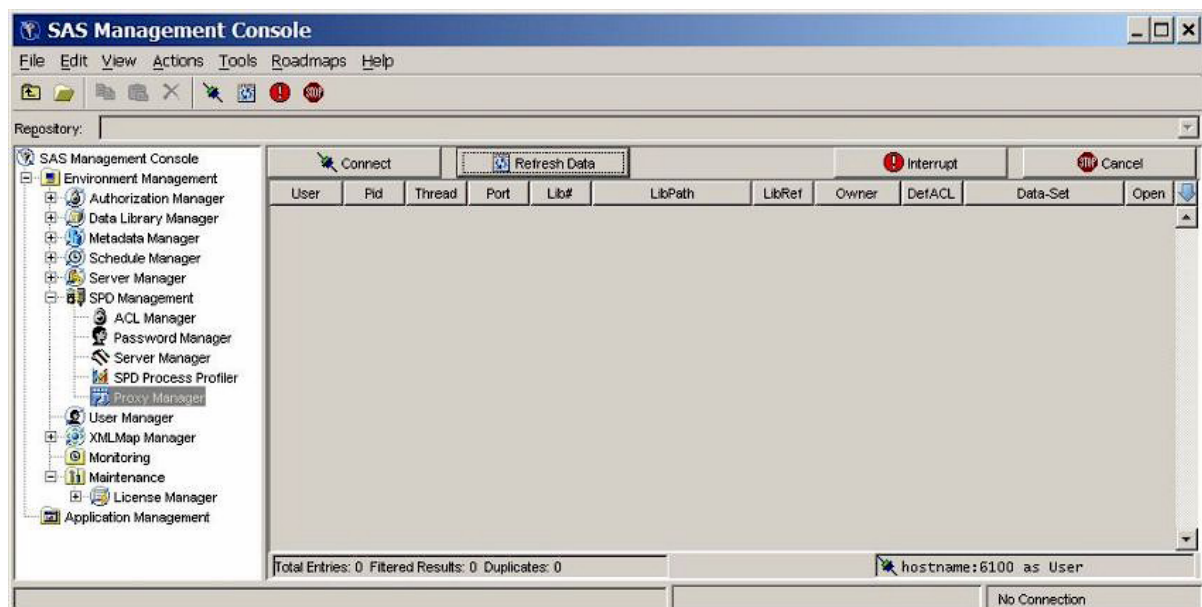
If the SAS Management Console user also connects to the same SPD Server name server through ACL Manager, Password Manager, or Server Manager, the user information that is displayed is the SPD Server user name that was used to connect with the LIBNAME statement. Otherwise, the user name of the user that 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 the 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 76. 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 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

SPD Server SQL Query Rewrite Facility

Overview of the SQL Query Rewrite Facility	93
Configuring Storage Space for the SQL Query Rewrite Facility	93
SQL Query Rewrite Facility Options	94
_QRWENABLE Option	94
_QRW Option	95

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 with the SQL Query Rewrite Facility 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 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 TEMP=YES environment setting that permits concurrent access to tables in the domain by multiple SPD Server users. Multiple SPD Server users can specify and share remapped TMPDOMAIN= locations without experiencing table handling or contention issues.

Note: If the SPD Server parameter TMPDOMAIN is not configured and the SQL query rewrite is enabled, the query rewrite will fail 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 SPD Server with Other Clients

Introduction	97
Using Open Database Connectivity (ODBC) to Access SPD Server Tables	98
Requirements and Considerations for Using ODBC	98
Install ODBC Drivers on the Server	98
Prepare Your Client Machine for ODBC Installation	99
Configure ODBC on the Client	99
Connecting to SPD Server Using an ODBC Connection	100
Create a Query Using an ODBC-Compliant Program	103
Using JDBC (Java) to Access SPD Server Tables	103
Requirements and Considerations for Using JDBC	103
Set Up JDBC on the Server	103
Set Up JDBC on the Client	103
Make a Query with JDBC	104
HTML File Requirement for JDBC	104
Limitations of Using JDBC with SPD Server	105
Using htmSQL to Access SPD Server Tables	106
Requirements and Considerations for Using htmSQL	106
Set Up htmSQL on the Server	106
Set Up htmSQL on the Client	106
Make a Query with htmSQL	107
Set Up an htmSQL Web Page	107
Using the SQL C API to Access SPD Server Tables	107
Requirements and Considerations for using the SQL C API	108

Introduction

SAS Scalable Performance Data (SPD) Server provides ODBC, JDBC, htmSQL, 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.
- **htmSQL** (HyperText Markup Structured Query Language): 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 htmSQL driver in SPD Server, all you need is an htmSQL Web page and a client machine with a browser. 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 you are familiar with.

Install ODBC Drivers on the Server

Instructions for installing the ODBC driver are included in the download package.

Prepare Your Client Machine for ODBC Installation

Before you create an ODBC 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 102.](#)

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

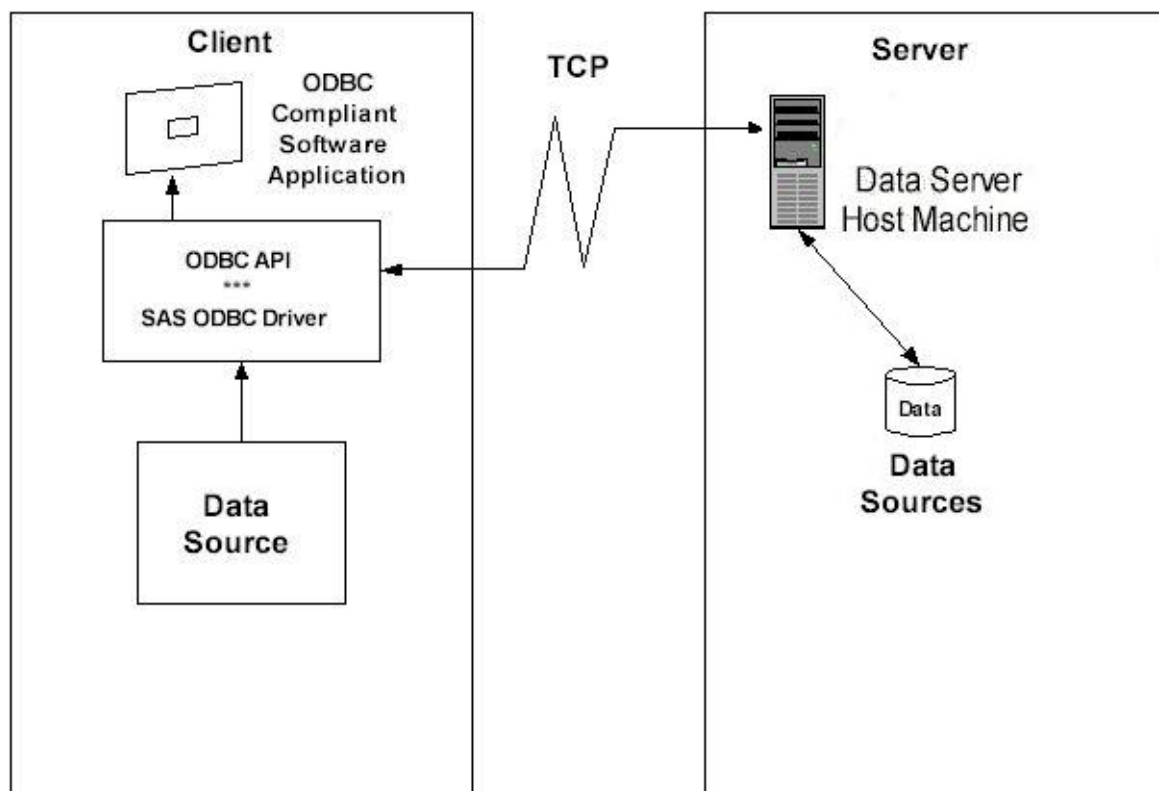
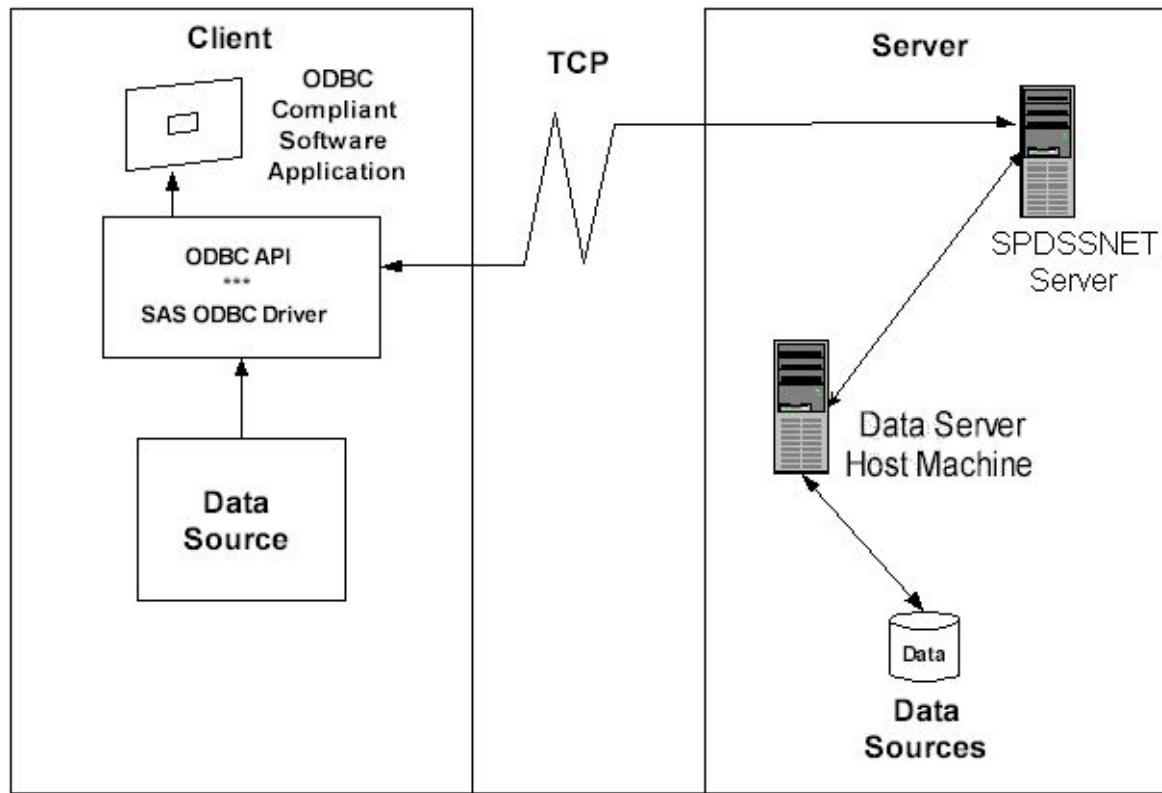


Figure 9.2 Configure ODBC to Connect SPD Server Client to SPD SNET Server

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 ODBC 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 will connect.

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 will connect.

Connection Options

Enter values for the following options. Additional SPD Server LIBNAME options might be listed. For more information, see “SPD Server LIBNAME Options” in Chapter 13 of *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 ODBC 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 102](#). 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 will connect.

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 will connect.

Connection Options

Enter values for the following options. Additional SPD Server LIBNAME options might be listed. For more information, see “SPD Server LIBNAME Options” in Chapter 13 of *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 99.
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 & 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 101](#). 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 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:

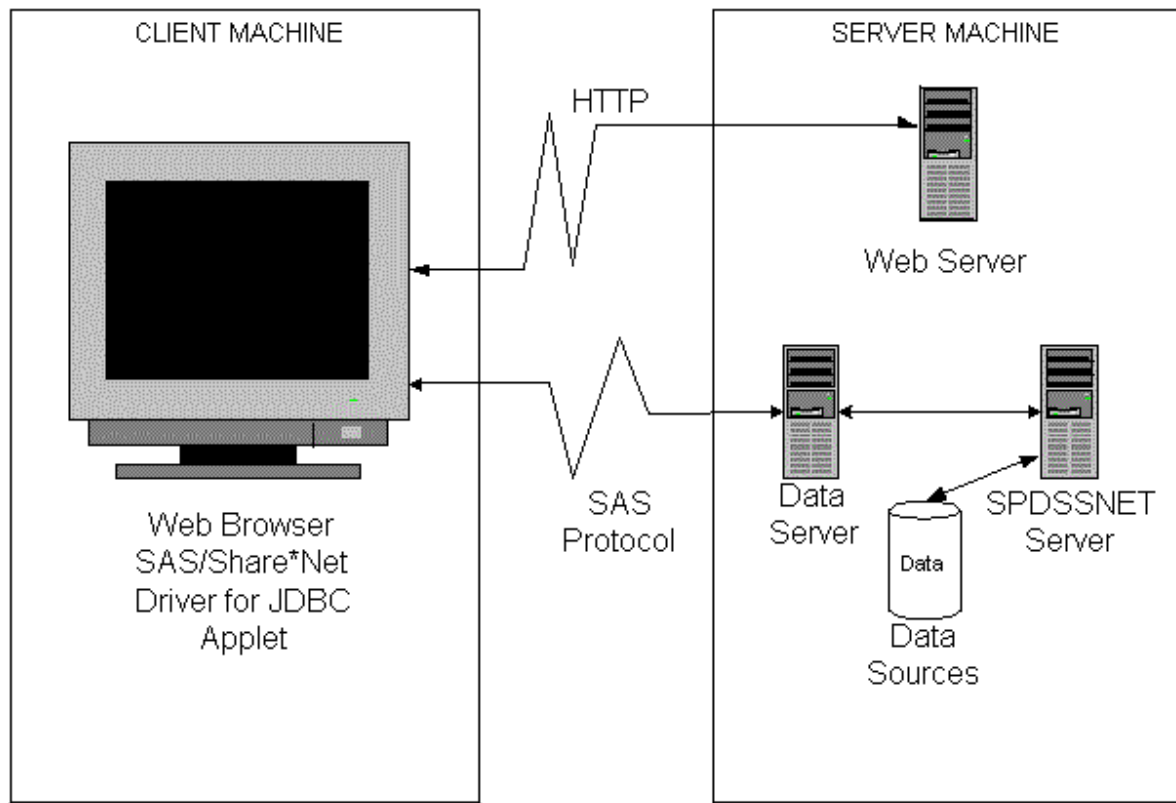
- You do not have Base SAS software on the network client to process the data sets.
- You want to distribute the information across your corporate intranet through a Web page.
- The clients on your network are varied: UNIX machines, Windows PCs, and workstations.
- The audience for the information wants 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 on the Server

JDBC is usually set up on the server at the time the SPD Server is installed. The process is covered in the SPD Server installation manual.

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 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 will 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')

```

Using htmSQL to Access SPD Server Tables

Requirements and Considerations for Using htmSQL

To use htmSQL, you must have SPD Server tables available on your network. htmSQL must be loaded and configured on a UNIX or Windows operating system, and SPD Servers and SPD SNET servers must be running. If you are working in such an environment, you might want to use htmSQL if one or more of the following criterion are true:

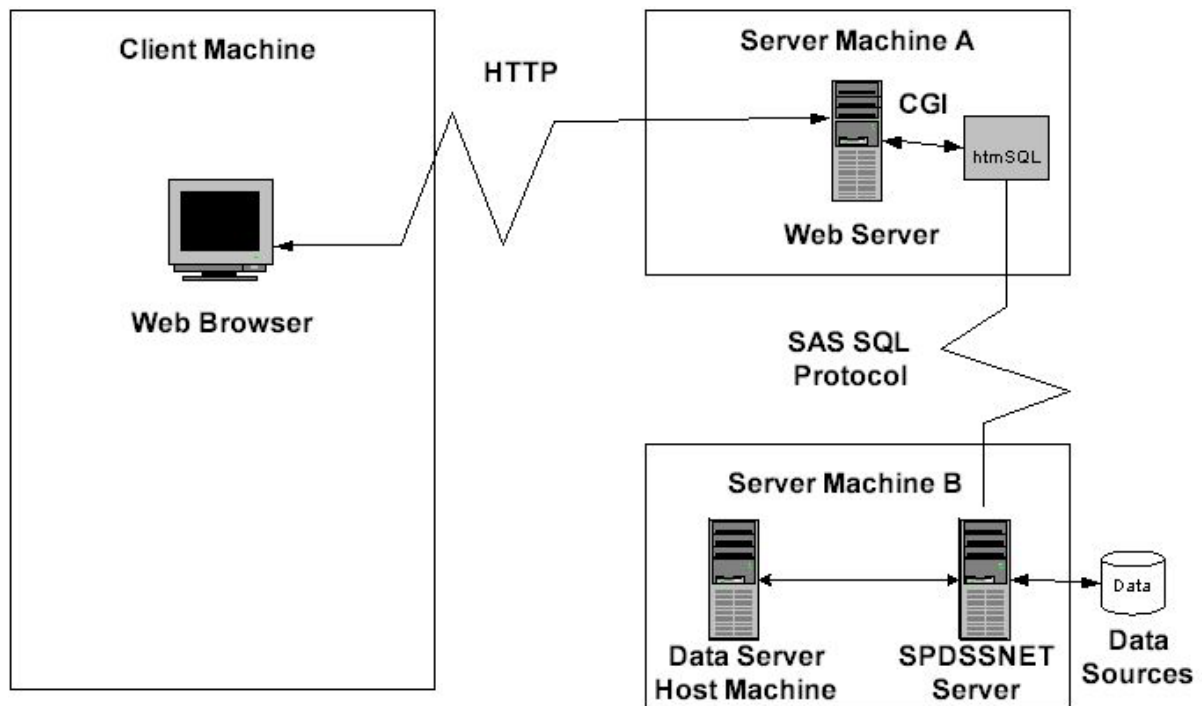
- You do not have Base SAS software on the network client to process the data sets.
- You want to distribute the information across your corporate intranet through a Web page.
- The clients on your network are varied: UNIX boxes, Windows PCs, and workstations.
- The audience for the information wants point-and-click access to the data.
- You would like to use the JDBC option to extract the information but cannot permit Java applets to run on your network browsers.
- You want to distribute the information over the Internet.
- Your developers are familiar with SQL and HTML.

Set Up htmSQL on the Server

htmSQL is usually set up on the server at the time the SPD Server is installed. The process is covered in the SPD Server installation manual. htmSQL must be installed on the Web server. You need the name of a data source that points to the SPD SNET server and to the specific LIBNAME domain that contains the SPD Server data you are interested in.

Set Up htmSQL on the Client

The only requirement for htmSQL is a browser on the network or Web client.

Figure 9.4 *htmSQL Configured on an SPD Server Client***Make a Query with htmSQL**

To use htmSQL to make a query:

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

Set Up an htmSQL Web Page

SAS Institute maintains a Web site that explains the technical details of setting up htmSQL Web pages. In some cases, there are references to the SAS/SHARE product. The rules for setting up htmSQL for either the SPD Server or SAS/SHARE are virtually the same.

The SAS Institute Web page for htmSQL is at http://support.sas.com/documentation/cdl/en/htmsql/59546/HTML/default/main_contents.htm.

Using the SQL C API to Access SPD Server Tables

Read this section if you do not have Base SAS software on the network client but you want to provide your network client machines with the capability of accessing SPD Server tables, using SQL query methods. You must have SPD Server tables available for

use, SPD Servers and SPD SNET servers running, and Network client machines capable of running C/C++ programs.

Requirements and Considerations for using the SQL C API

To use the SQL C API, you must have SPD Server tables available on your network. You must also have SPD Servers and SPD SNET Servers running, and network client machines that can run C/C++ programs. If you are working in such an environment, you might want to use the SQL C API 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.
- The clients on your network are varied: UNIX boxes, Windows PCs, workstations. One thing they might have in common is the ability to run C/C++ programs.
- Your developers are familiar with SQL and C/C++.

For additional information about SQL C API, see “SAS Scalable Performance Data (SPD) Server SQL Access Library API Reference” in Chapter 10 of *SAS Scalable Performance Data Server: User's Guide*

Chapter 10

Configuring Disk Storage for SPD Server

Introduction	109
SPD Server Component File Types and Sizes	109
Configuring LIBNAME Domain Disk Space	110
Example 1: Primary File System Storage for All Component Files	110
Example 2: Using ROPTIONS= to Store SPD Server Table Data and Index Component Files in Other File Systems	111
Example 3: Adding More File Systems to a Path Option When Its File System Is Full	111
Advantages of Using ROPTIONS=	112

Introduction

This section discusses how to manage large SPD Server data stores that consume gigabytes 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
Segmented index	.idx	Medium to large	0 or more
Global index	.hbx	Medium to large	0 or more

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. 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 maintains a segmented view of the index that includes a list of all of the segments that each key is a member of. A bitmap is used to determine the per-segment observations for each key.

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 for the index keys is distributed across segments. An index key that is in many segments requires a larger segment list. Larger segment lists require a larger number of per-segment bitmaps, as compared to an index key that is found only in a small number of segments.

The best-case scenario for creating an optimally sized .idx file occurs when the table is sorted by the indexed columns, in order to minimize the number of segments that the key is in. The worst case scenario for creating an optimally sized .idx file occurs when the index keys are in a large number of segments, with a low cardinality of rows for each segment.

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 the metadata component (.mdf) must start in the primary file system. If all the available space in the primary file system is consumed, SPD Server cannot create new tables until disk space becomes available.

Example 1 stores all the component files (metadata, data, and index data) in the primary file system. This arrangement can cause problems if you use large tables. Large tables can quickly fill up the primary file system. To avoid this problem, you 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).

Advantages of Using ROPTIONS=

ROPTIONS= specifications override any corresponding options that your SAS users include in their programs. In example 3, the system administrator maintains explicit control of disk usage. Even if SAS users specify file systems using the DATAPATH= and INDEXPATH= LIBNAME options during their LIBNAME connection, the administrator's use of DATAPATH= and INDEXPATH= with ROPTIONS= overrides the SAS users' specifications.

In contrast, when you use OPTIONS=, a keyword that is syntactically the same as ROPTIONS=, you are not overriding a user specification. Instead, you are supplementing the user-specified options. In this case, if a user specifies an option, the user's setting is implemented. If the user omits an option, your OPTIONS= specification in the libnames.parm file is used.

CAUTION:

Do not specify DATAPATH=, INDEXPATH=, and METAPATH= with OPTIONS=. If you use OPTIONS=, the disk allocation results cannot be predicted. Use ROPTIONS= to explicitly control disk usage at your site.

Chapter 11

Setting Up SPD Server Parameter Files

Introduction	113
Syntax for the <code>-parmfile</code> Option	113
Syntax for the <code>spdsserv.parm</code> Options	114
Server Performance Levels	114
<code>spdsserv.parm</code> Options	115
SPD Server Parameter File Configurations for LDAP	121
SPD Server Parameter File Configurations for Auditing	122
<code>[NO]WHEREAUDIT</code>	122
<code>SQLAUDLEN=</code>	123
<code>WHAUDLEN=</code>	123

Introduction

The **spdssrv** command, which starts an SPD Server host, requires a parameter file that is named `spdsserv.parm`. You specify the name of the `spdsserv.parm` file with the `-parmfile` command-line option.

Syntax for the `-parmfile` Option

The syntax for the `-parmfile` option is:

```
-parmfile file-spec
```

file-spec is an explicit file path for the SPD Server parameter file. The `spdsserv.parm` 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 `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. All option keywords are case sensitive and must be capitalized. Comments are not 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/spdsserv.parm**. These examples contain the recommended default settings for SPD Server. **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 Table database is assigned a performance class attribute. For more information about user performance classes, see [“The Password Manager Utility psmgr” on page 191](#). 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`.

spdsserv.parm 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.

FMTDOMAIN=

specifies the domain where the user-defined format is stored. To use formats that are user-defined, create a domain that is named FORMATS. Use FMTDOMAIN= with FMTNAMENODE= and FMTNAMEPORT=.

Usage:

```
FMTDOMAIN=FORMATS ;
```

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 ;
```

GRPBYPROWCACHE=

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 GRPBYPROWCACHE 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 GRPBYPROWCACHE 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:

GRPBYPROWCACHE= *<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:

MAXWHTHREADS= *<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]NLSTRANS CODE

enables or suppresses the server-side SPD Server NLS processing. The default value for NLSTRANS CODE is NONLSTRANS CODE if the option is not found in the spdsserv.parm file. The default spdsserv.parm file for SPD Server does not contain the NLSTRANS CODE option. Users must explicitly activate server-side transcoding in SPD Server 4.5.

When you specify NONLSTRANS CODE, 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 4.5 (with SAS 9) 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 NLSTRANS CODE, 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:

NLSTRANSODE ;

NONLSTRANSODE ;

[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 4.5, 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= allocates is divided evenly among the sort threads. For best results, specify SORTSIZE= values in the range 256 MB–1 GB per parallel sort thread, or between 256 * MAXSORTTHREADS= and 1 GB * MAXSORTTHREADS=.

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 8 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 LDAPPOR= 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 LDAPPOR= 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 Configurations for Auditing

[NO]WHEREAUDIT

Enables audit logs for WHERE clauses that are submitted to SPD Server. Specify the WHEREAUDIT option in the spdserv.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_mmmddyyyy.spdsaudit*. In the output, *mmmddyyyy* 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 SPD Server Libname Parameter Files

Introduction	125
Domain Naming Syntax for libnames.parm	126
Domain Path Options	126
Consistency in Nomenclature	128
Domain Access Options	129
Controlling the Precedence of Permission Checks with the LIBACLINHERIT= Option	129
Dynamic Locking	132
Overview of Dynamic Locking	132
Benefits of Using Dynamic Locking	132
How Dynamic Locking Works	132
Organizing Domains for Scalability	133
Overview of Organizing Domains	133
Data Table Space	133
Index Table Space	134
Metadata Table Space	134
SPD Server Workspaces	134
Domains and Data Spaces	135
Overview of Domains and Data Spaces	135
Permanent Table Space	135
Semi-Permanent Table Space	136
Temporary Table Space	136
Example libname.parm File Configurations	137
Example 1: Minimum Configuration for Domain Declaration	137
Example 2: Specify Domain Paths for Data, Index, and Workspace Tables	137
Example 3: Query-Rewrite Domain Configuration	138
Example 4: Multiple Domain Types and Paths Configuration	139

Introduction

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>
  OPTIONS="option-1 <...option-n>"
  ROPTIONS="option-1 <...option-n>"
  OWNER=owner-id
  LIBACLINHERIT=<YES/NO>
  DYNLOCK=<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 will contain the metadata tables that are associated with the domain. By default, the PATHNAME= specification will also contain the data tables, index tables, and intermediate tables that the domain creates. SPD Server administrators and users can use the options described in [“Domain Path Options” on page 126](#) 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';
```

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. Specify ROPTIONS (reserved option) to ensure that the domain options that you declare in libnames.parm cannot be modified by subsequent libref statements submitted to SPD Server using SAS code.

All options that are specified in libnames.parm files must be either standard options or reserved options (ROPTIONS). You cannot specify a combination of reserved and non-reserved options in the libnames.parm file.

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

```
LIBNAME=domain-name PATHNAME=primary-metadata-path ;
    OPTIONS="<option-1 ... option-n>" ;
```

```
LIBNAME=domain-name PATHNAME=primary-metadata-path ;
    ROPTIONS="<option-1 ... option-n>" ;
```

The following list describes the LIBNAME domain path options for SPD Server.

DATAPATH=

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

Usage:

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

INDEXPATH=

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

Usage:

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

WORKPATH=

specifies a list of paths that will contain temporary SPD Server work tables and temporary SPD Server intermediate files that are associated with the declared domain. You can use OPTIONS or 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 ' )
```

Consistency in Nomenclature

Although not required, you should match or closely match the LIBNAME, pathname, and additional optional pathnames for consistency. The following examples illustrate a domain declaration that is easy to follow, and a domain declaration that is more difficult to follow.

In the following example of intuitive names in a libnames.parm file, the declared domain name, pathname, data pathname, and index pathname are **spds123**,

```
LIBNAME=SPDS123 PATHNAME=c:\data\spds123
OPTIONS="
  DATAPATH= ('d:\data\spds123'
             'e:\data\spds123')
  INDEXPATH= ('f:\idx\spds123') " ;
```

In this example of non-intuitive names in a libnames.parm file, each pathname is different. The structure is technically valid, but unnecessarily complex.

```
LIBNAME=BADEXAMPL PATHNAME=c:\data\myspds
OPTIONS="
  DATAPATH= ('d:\data\datapath1'
             'e:\data\datapath2')
  INDEXPATH= ('f:\idx\index') " ;
```

The directories that are specified in domain pathname, data path, and index path statements should correspond to one and only one domain. In the first example, the pathname, data path, and index path specifications point to separate, unique paths that end with the directory name **spds123**, which corresponds to the domain **spds123**. If a domain **spds456** exists, it should have its own unique domain pathname, data path, and index path specifications. Also, it should not share a specified path with **spds123** or with any other domain.

Domain Access Options

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.

OWNER=

specifies the owner of a domain. The SPD Server owner controls the resources of the domain and can create a LIBNAME ACL on the domain to grant or deny privileges 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
- 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”](#) on page 129.

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 132.

Usage:

DYNLOCK=<YES/NO>

Controlling the Precedence of Permission Checks with the LIBACLINHERIT= Option

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 give READ or WRITE 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 permissions are defined on the object for the user, the user gets these permissions.
2. If group-specific permissions 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, the user gets the LIBNAME ACL permissions on the object.
4. If LIBNAME ACL permissions are defined for the user's group, the user gets the LIBNAME ACL permissions on the object.
5. Otherwise, the user gets UNIVERSAL ACLs on the resource.

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 the way SPD Server stores files. “[Domain Access Options](#)” on page 129 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 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.

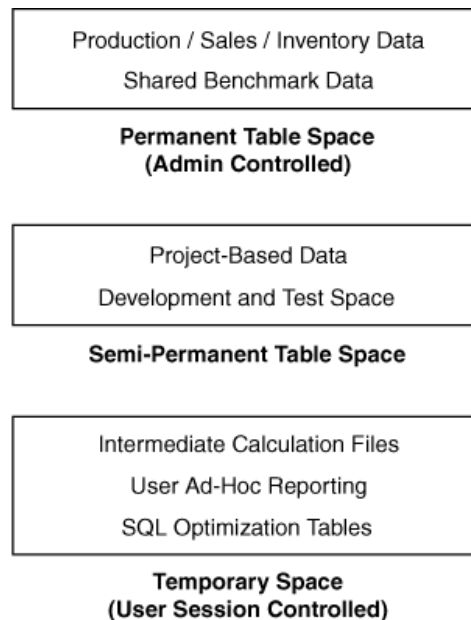
You typically configure a workspace as part of a large striped file system that spans as many disks and I/O channels as possible. Workspace I/O can critically impact the performance behavior of an SPD Server host.

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, and 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 best suited to 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 modifies the way 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 best suited to 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
options="
  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 4.5, 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
options="
  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 spdserv.parm file code (the TMPDOMAIN=QRW statement references the domain created for query rewrite tables):

```
SORTSIZE=128M;
INDEX_SORTSIZE=128M;
GRPBYROWCACHE=128M;
BINBUFSIZE=32K;
INDEX_MAXMEMORY=8M;
NOCOREFILE;
SEQIOBUFMIN=64K;
RANIOBUFMIN=4K;
MAXWHTHREADS=8;
WHERECOSTING;
RANDOMPLACEDPF;
MINPARTSIZE=128M;
TMPDOMAIN=QRW;
WORKPATH= ('c:\temp\work1');
```

Example 4: Multiple Domain Types and Paths Configuration

This example uses a combination of libnames.parm, spdserv.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
options="
  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
options="
    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
options="
    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 SPD Server Performance Server

Introduction	143
Starting SPD Server Performance Server	144
Overview of Starting the Performance Server	144
Start Performance Server from the Command Line	145
Start Performance Server by Using the rc.perf Script	145
Sample rc.perf Script	146
Performance Server Log File	148

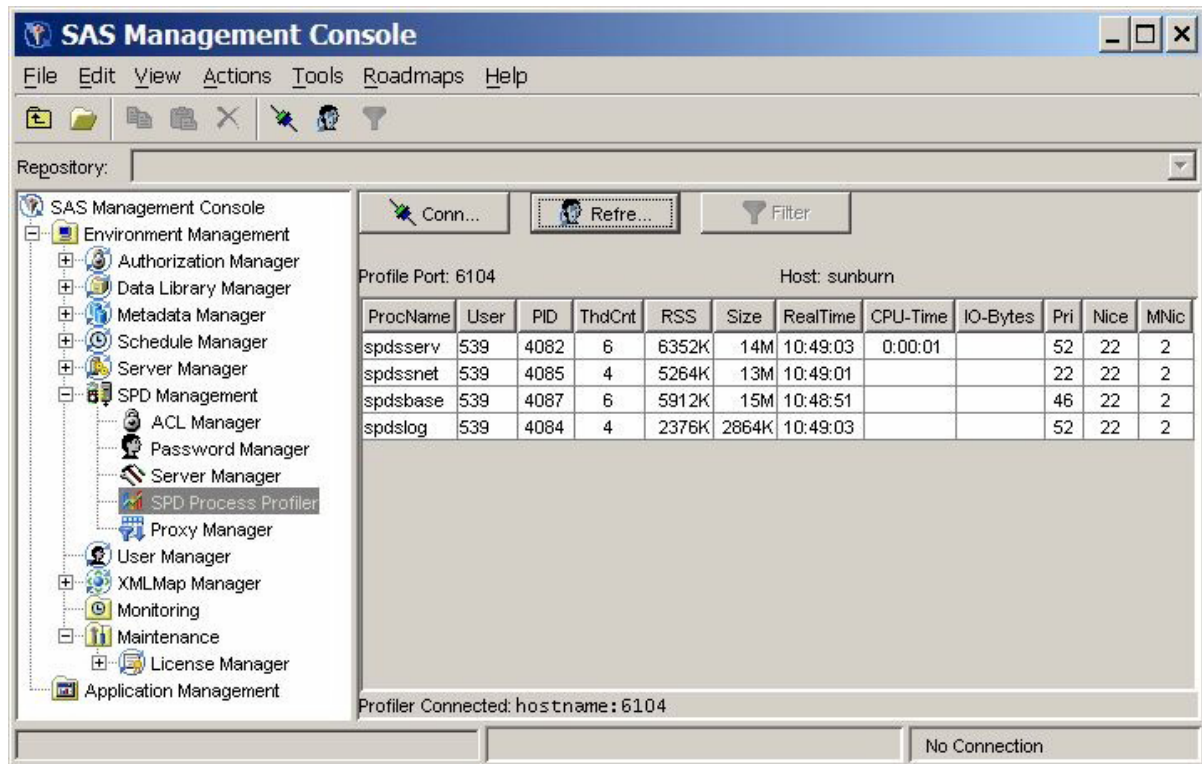
Introduction

SAS SPD Server 4.5 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 ships 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 146](#) 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 4.5 also ships 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 145](#) 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

Security

<i>Chapter 14</i>	
ACL Security Overview	151
<i>Chapter 15</i>	
Symbolic Substitution	187
<i>Chapter 16</i>	
Managing SPD Server Passwords, Users, and Table ACLs	191
<i>Chapter 17</i>	
DICTIONARY.PWDB and DICTIONARY.ACLS	205
<i>Chapter 18</i>	
Using SPD Server with an Internet Firewall	209
<i>Chapter 19</i>	
SPD Server Auditing	213
<i>Chapter 20</i>	
SPD Server Table WHERE Constraints	217

Chapter 14

ACL Security Overview

ACL Security Overview	151
SPD Server ACL Security Model	152
Overview of the ACL Security Model	152
Enabling ACL Security	152
Disabling ACL Security	155
Controlling SPD Server Resources with PROC SPDO and ACL Commands . . .	156
Using PROC SPDO	156
Using ACL	156
ACL Concepts	157
Overview of the ACL Command Set	159
SET ACLTYPE memtype;	159
SET ACLUSER [name];	160
ADD ACL acl1 acl2... [C=cat T=type] [/options]	160
ADD ACL Examples	161
MODIFY ACL and MODIFY ACL _ALL_	162
MODIFY ACL Examples	163
LIST ACL and LIST ACL _ALL_	164
LIST ACL Examples	164
DELETE ACL and DELETE ACL _ALL_	165
DELETE ACL Examples	166
ACL Security Examples	167

ACL Security Overview

SPD Server uses Access Control Lists (ACLs) and SPD Server user IDs to secure domain resources. You obtain your user ID and password from your SPD Server administrator.

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

ACL file security is turned on by default when an administrator brings up SPD Server. ACL permissions affect all SPD Server resources, including domains, tables, table columns, catalogs, catalog entries, and utility files. When ACL file security is enabled, SPD Server only grants access rights to the owner (creator) of an SPD Server resource. Resource owners can use PROC SPDO to grant 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

universal READ access to the resource (read or query).

WRITE

universal WRITE access to the resource (append to or update).

ALTER

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

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

GROUPREAD

group READ access to the resource (read or query).

GROUPWRITE

group WRITE access to the resource (append to or update).

GROUPALTER

group ALTER access to the resource (rename, delete, or replace a resource and add, delete indexes associated with a table).

SPD Server ACL Security Model

Overview of the ACL Security Model

SPD Server provides an Access Control List (ACL) based security system. The ACL-based security is enabled by default. You are encouraged to run SPD Server using ACLs. ACLs add little overhead to SPD Server in terms of execution speed and disk space consumption. ACLs keep files private to individual users and within groups.

Only disable ACLs if your computing environment requires free access of any user to any other user's files. Migrating from a non-ACL environment to an ACL-based environment is not simple, so use ACLs if you foresee needing security controls at a future time. Files created by SPD Server running ACLs only should be accessed by SPD Servers running ACLs. Likewise, areas created without ACLs should be accessed only by SPD Servers using -NOACL.

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

Enabling ACL Security

Overview of Enabling ACL Security

You enable SPD Server security with the -ACL option on the spdserv command. Numerous security features are in effect with ACLs enabled.

UNIX File-Level Protection with ACL Security

Each session of SPD Server is attached to a user with some UNIX or Windows user ID. If SPD Server runs on UNIX, all files created by the software are protected according to the UNIX file creation permissions associated with that UNIX user's ID. The SPD Server only can read or write files that have the appropriate file and directory access permissions to the SPD Server's user's ID. Use the UNIX unmask command to restrict the creation permissions.

User/Password Validation

SAS users must issue a user ID and password with the LIBNAME statement in order to connect to SPD Server. The user ID and password are verified against an SPD Server user ID table set up by the system administrator. Password expiration can be enforced by the system administrator via the psmgr administration tool for the user ID table or through the SAS Management Console, if it is installed and configured for SPD Server. In either of the two environments, the system administrator can prevent logins under the anonymous user ID by placing user 'anonymou' in the user ID table with a password unknown to the SAS users.

Control of LIBNAME Domains by the System Administrator with ACL Security

The system administrator defines the valid LIBNAME domains with 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

In the LIBNAME parameter file, the system administrator 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.)

User Ownership of Tables

Each table created is tagged with the SPD user ID (referred to as the owner) 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.)

Example Server Setup with ACL Security

The following command invokes SPD Server with ACL support enabled and configures it with the specified LIBNAME domain definitions.

```
spdssrv -ACL -aclDir
InstallDir/site -nameserver samson
-LIBNAMEfile libnames.parm
```

The libnames.parm file contains:

```
LIBNAME=public pathname=/disk1/public;
LIBNAME=qadata pathname=/disk2/qadata
owner=qamgr;
LIBNAME=marketing pathname=/disk3/marketing
```

```
owner=mktmgr;
LIBNAME=clinical pathname=/disk4/clinical
owner=drzeuss;
```

SPD Server is invoked connecting to the name server running on machine 'samson'. The password file listing all valid system users resides in directory 'InstallDir/site'.

LIBNAME domains 'public', 'qadata', 'marketing' and 'clinical' are registered with the name server. The **/disk1/public**, **/disk2/qadata**, **/disk3/marketing**, and **/disk4/clinical** directories must exist and the user ID that invokes **spdsrv** must have read and write access to them.

The following LIBNAME statements connect SAS clients to the data areas:

```
LIBNAME open sasspds 'public'
  host='samson'
  user='employee'
  prompt=yes;

LIBNAME pres sasspds 'clinical'
  host='samson'
  user='ceo'
  prompt=yes;

LIBNAME report sasspds 'marketing'
  host='samson'
  user='ceo'
  aclgrp='mrktng'
  prompt=yes;

LIBNAME efficacy sasspds 'clinical'
  host='samson'
  user='drfda'
  prompt=yes;
```

Additionally, ACLs can be created on the LIBNAME domains themselves and the resources that are created within them. The simplest way to do this is using PROC SPDO. The following example demonstrates this:

```
LIBNAME clin sasspds 'clinical'
  host='samson'
  user='drzeuss'
  prompt=yes;
PROC SPDO lib=clin;
set acluser;
add ACL /
  LIBNAME groupread;
modify ACL /
  LIBNAME drfgood=(y,y,y);
quit;
```

The owner of the LIBNAME domain 'clinical' has granted permission to other members of his or her ACL group to the LIBNAME domain to have READ access to the domain. This permits these users to perform SAS LIBNAME assignments to the domain. Users not belonging to the owner's ACL group will not even be permitted to make LIBNAME assignments to the 'clinical' domain. The owner has also granted READ, WRITE, and CONTROL access to the explicit user 'drfgood'. This enables 'drfgood' to make

LIBNAME assignments and write new files to the 'clinical' domain, and to also alter the LIBNAME ACL permissions if desired.

Disabling ACL Security

Overview of Disabling ACL Security

You disable SPD Server security by using the -NOACL option with the **spdssrv** command. When ACLs are disabled, there are almost no security restrictions in the SPD Server environment. Anyone can access SPD Server, as long as they know the LIBNAMES that are defined by the system administrator in the -LIBNAME file.

UNIX File-Level Protection with ACL Security Disabled

In UNIX, each SPD Server session runs under a UNIX user ID. All files created by SPD Server therefore are protected according to the UNIX file creation permissions of that UNIX user ID. Use the UNIX 'umask' command to restrict the desired creation permissions. File permissions are based on the permissions of the directory where the file was created.

Control of LIBNAME Domains by the System Administrator without ACL Security

The system administrator defines the valid LIBNAME domains with entries in the LIBNAME parameter file for each SPD Server. PATHNAME= defines the file system for the LIBNAME. LIBNAME= provides the access route to the file system. Restricting knowledge of the LIBNAME= labels restricts access to the corresponding file system.

Example Server Setup without ACL Security

The following command invokes SPD Server without ACL security enabled.

```
spdssrv -noacl -aclmdir
InstallDir/site -nameserver samson
-libnamefile libnames.parm
```

The libnames.parm file contains:

```
LIBNAME=open_access
    pathname=/disk1/sas_tables;
LIBNAME=mgmt_access
    pathname=/disk2/managers/data;
```

SPD Server is invoked, connecting to the name server running on the machine called 'samson'. Despite no ACLs, a password file is still required in the directory called **InstallDir/site**.

Note: InstallDir is a documentation substitute for the actual path specification for the directory where SPD Server is installed on a particular machine.

LIBNAME domains 'open_access' and 'mgmt_access' are registered with the name server. The /disk1/sas_tables and /disk2/managers/data directories must exist, and the user ID that invokes spdssrv must have read and write access to those directories. The following LIBNAME statements connect a SAS client to the data areas:

```
LIBNAME open sasspds 'open_access'
    host='samson';
```

```
LIBNAME mgmt sasspds 'mgmt_access'
host='samson';
```

Controlling SPD Server Resources with PROC SPDO and ACL Commands

Using PROC SPDO

Overview of PROC SPDO

PROC SPDO is the SAS procedure for the SPD Server operator interface.

PROC SPDO runs only on systems where the SAS is installed.

PROC SPDO Command Set

To invoke PROC SPDO, submit:

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

where libref is a LIBNAME that was previously allocated to the sasspds engine.

Currently there are two classes of PROC SPDO commands:

- ACL commands
- LIBNAME proxy commands

The ACL commands are described below with some simple examples that demonstrate their syntax and usage. More information about LIBNAME Proxy Commands is available in [Chapter 21, “SPD Server Operator Interface Procedure \(PROC SPDO\),” on page 223](#).

Using ACL

An SPD Server Access Control List (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 defined, the accessor gets these permissions.
2. If a resource is owned by the same ACL group as the accessor, the accessor gets the resource's GROUP permissions.
3. If the resource is owned by a different ACL group than the accessor, the accessor gets the resource's UNIVERSAL permissions.

To turn on LIBACLINHERIT permissions in your spdsserv.parm file, submit the statement:

```
LIBACLINHERIT ; .
```

To turn off LIBACLINHERIT permissions in your spdsserv.parm file, submit the statement:

```
NOLIBACLINHERIT ; .
```

You can also use your libnames.parm file to turn on LIBACLINHERIT permissions.

To turn on LIBACLINHERIT permissions in your libnames.parm file, submit the statement:

```
LIBACLINHERIT=YES.
```

To turn off LIBACLINHERIT permissions in your libnames.parm file, submit the statement::

```
LIBACLINHERIT=NO.
```

If the LIBACLINHERIT parameter file option is turned on, the ACL precedence of permission checks changes. Turning on LIBACLINHERIT creates a LIBNAME ACL on the specified LIBNAME domain. The LIBNAME ACL grants users rights to all resources within the LIBNAME domain. When a LIBNAME ACL is created for a specified LIBNAME domain, the ACL precedence of permission checks becomes:

1. Check user-specific permissions first. If defined, the accessor gets these permissions.
2. If a resource is owned by the same ACL group as the accessor, the accessor gets the resource's GROUP permissions.
3. LIBNAME ACL permissions are used for domains where LIBACLINHERIT is turned on.
4. If the resource is owned by a different ACL group than the accessor, the accessor gets the resource's UNIVERSAL permissions.

ACL Concepts

ACL Groups

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

The SPD Server administrator can affiliate a given SPD Server user ID with up to five ACL groups. When you connect to an SPD Server using a LIBNAME assignment, you assert a specific ACL group using the ACLGRP= option.

The ACLGRP= value in your LIBNAME assignment must match one of the five groups that the administrator defined for you. If you do not assert ACLGRP= in your LIBNAME assignment, the SPD Server affiliates you with your default ACL group. (This is the first group in the list of five.)

When defining 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 privileges to the ACL group instead of only to a specific SPD Server user.

Column Security

SPD Server allows you to control access to table contents at the column level through the use of ACLs. Column security ACLs can be applied to individual users at the user level, or applied 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 they are

explicitly granted 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 user group, the protected column automatically becomes unavailable to **all** individual users and user groups, unless they are specifically given access to the protected column.

Examine a scenario where 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 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.

First, you submit a user-level ACL statement to restrict the secure column in table TESTING from Joe. Joe is explicitly denied access, but since the column is now a protected entity, all other TESTING table users are also denied access to the column by default. Once a column is protected via ACL security, explicit permissions must be granted in order for any user (or groups of users) to be able to access the column content. 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 that explicitly grants access to the protected column.

SPD Server reads the user-level ACL permissions first, and gives Joe access to the table TESTGROUP, but restricts him from the secure column. Then 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. This results in all of TESTGROUP having full table access, except Joe. Joe's user-level ACL column security restriction prevents him from accessing the classified column.

Now consider another scenario, where John manages a group DEVGROU 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 DEVGROU 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 DEVGROU members can have full table permissions at the group level, but will not see the protected column because John's user-level column security ACLs will override any group-level ACLs for the DEVGROU table.

Generic ACL

You can use generic ACL names for a class of resources that have a common prefix. You can use the asterisk symbol "*" as a wildcard. This permits you 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 symbol to create the generic ACL name, SALES*, to cover them all. You then would define your ACL permissions on the SALES* generic ACL.

When using PROC SPDO, use the /GENERIC command option to identify a generic ACL.

Note: If you specify /GENERIC when defining a table column ACL, the /GENERIC applies to the table name, not to the column name. You cannot use wildcards with column names.

LIBNAME ACL

You can control access permissions to an entire LIBNAME domain with the SPD Server ACL facility. When 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 using PROC SPDO, use the /PERSIST command option to identify a persistent ACL.

Resource

A PROC SPDO resource is

- 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 within a table. Use the normal SAS convention of **table.column** when specifying the table and column that you want to secure.

When issuing 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 using SPDO commands that support the `_ALL_` resource name.

Giving Control to Others

You permit other SPD Server users to alter your own ACL entry by granting a specific user/group ACL entry. See [“MODIFY ACL and MODIFY ACL _ALL_” on page 162](#) for more information about user-specific ACL entries.

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 perform an ACL-related command, you must first assert an ACL user ID to define the scope of your access. In addition, you might 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 and/or member type at any point in a command sequence, and then 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 your view to only those ACL records that have the specified user name as the owner of the ACL entry. If name is omitted, the default is the user who assigns the libref.

To actually perform an ACL operation on a resource entry, you must

- be the ACL entry owner, or
- have CONTROL access over the ACL entry, or
- have ACLSPECIAL=YES enabled on your PROC SPDO LIBNAME connection.

Note: You must first issue a SET ACLUSER command before issuing any of the following ACL commands.

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

Creates new ACL entries acl1 acl2... where ACL entries acl1 acl2 ... can be one-part resource names or two-part table column names.

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

Specifies that acl1 acl2... are generic ACLs.

PERSIST

Specifies that acl1 acl2... are persistent ACLs.

LIBNAME

Identifies the special LIBNAME domain resource.

MODEL=acl-name

Specifies the name of another ACL. This option requests the software to copy all the access permissions and access list entries from this ACL.

C=cat

Identifies the specified ACL names acl1 acl2... as the names of catalog entries in the catalog **cat**. You pair this value with the T= option.

T=type

Identifies the catalog entry type to associate with the specified ACLs acl1 acl2... when you specify the C=cat option.

ADD ACL Examples**Add LIBNAME Domain ACL**

This ACL grants universal READ and group WRITE access.

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

Add Resource ACL

This ACL for the resource MINE_JAN1999 grants universal READ and WRITE access.

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

Add Generic ACL

This generic ACL for MINE* grants universal READ access.

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

Add 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 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 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 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 resources *acl1 acl2...* where ACL entries *acl1 acl2...* can be one-part resource names or two-part table column names. Specifying *_ALL_* modifies all existing ACLs for which you have control access. Specifying *_ALL_* as the table identifier in a two-part name modifies all tables for which the given column is matched. Specifying *_ALL_* as the column identifier in a two-part name modifies all columns for which the given table is matched. The characteristics modified are specified by options and/or user list.

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

Specifies that *acl1 acl2...* are generic ACLs.

LIBNAME

Identifies the special LIBNAME domain ACL.

C=cat

Identifies the selected ACLs as names of catalog entries from the catalog cat. This value must be paired with the T= option.

T=type

Identifies the catalog entry type used to qualify the selected ACLs when the C=cat option is specified.

userlist

can be user name = (Y/N,Y/N,Y/N,Y/N) for (READ,WRITE,ALTER,CONTROL).

MODIFY ACL Examples

Modify LIBNAME Domain ACL

This modifies a LIBNAME domain to set READ and WRITE access for a given user.

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

Modify ACL MINE

This modifies ACL MINE_JAN2003 to deny universal WRITE access and add user-specific permissions.

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

Modify Generic ACL

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

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

Modify All ACLs

This modifies all ACLs to grant READ access to a given user.

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

Modify Column ACL

This modifies 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 Generic Column ACL

This modifies generic column ACL, MINE*.SALARY, to add explicit READ and WRITE access for a given user.

```
modify acl mine.salary/generic
  debby=(y,y,n,n);
```

Modify ACL for a Catalog

This modifies catalog MYCAT to remove universal READ and group WRITE access.

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

Modify Generic ACL for Catalog Entries

This modifies a generic ACL for catalog entries, MYCAT.MY*.CATAMS, to remove universal READ access.

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

LIST ACL and LIST ACL _ALL_**LIST ACL acl1 acl2... [/options];****LIST ACL _ALL_ [/options];**

Lists information about specific ACLs acl1 acl2... where ACL entries acl1 acl2... can be one-part resource names or two-part (table.column) names. Specifying _ALL_ lists all existing resource ACLs for which you have control access. Specifying _ALL_ as the table identifier in a two-part name lists all tables for which the given column is matched. Specifying _ALL_ as the column identifier in a two-part name lists all columns for which the given table is matched.

List ACL Options:

GENERIC

Specifies that acl1 acl2 are generic ACLs.

LIBNAME

Identifies the special LIBNAME domain ACL.

C=cat

Identifies the selected ACLs as names of catalog entries from the catalog cat. This value must be paired with the T= option.

T=type

Identifies the catalog entry type 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 a specified table(s). This is equivalent to a LIST ACL table followed by a LIST ACL table._ALL_.

LIST ACL Examples**List All ACL Entries**

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

```
list acl _all_;
```

List a Generic ACL

This lists a generic ACL entry for MINE*.

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

List All Column ACLS for a Table

This lists all column ACLs for table MINE_JAN2003.

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

List All Column ACLs for All Tables

This lists all column ACLs for all tables.

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

List a Specific Column

This lists the column ACL for MINE_JAN2006.SALARY.

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

List All ACL Data for a Table

This provides all ACL information for table MINE_JAN2006.

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

List All ACLs for Catalogs

This lists all ACLs for the ACL type 'catalog'.

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

List All ACLs for a Catalog

This 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 resources acl1 acl2... where ACL entries acl1 acl2... can be one-part resource names or two-part table.column names. Specifying _ALL_ deletes all

existing resource ACLs for which you have control access. Specifying `_ALL_` as the table identifier in a two-part name deletes all tables for which the given column is matched. Specifying `_ALL_` as the column identifier in a two-part name deletes all columns for which the given table is matched.

Delete ACL Options:

GENERIC

Specifies that `acl1` `acl2` are generic ACLs.

LIBNAME

Identifies the special `*LIBNAM*` ACL.

C=cat

Identifies the selected ACLs as names of catalog entries from the catalog `cat`. This value must be paired with the `T=` option.

T=type

Identifies the catalog entry type used to qualify the selected ACLs when the `C=cat` option is specified.

DELETE ACL Examples

Delete a LIBNAME ACL

This deletes a LIBNAME ACL.

```
delete acl/LIBNAME;
```

Delete All ACLs for Current ACL Type

This deletes all the ACLs for the current ACL type.

```
delete acl _all_;
```

Delete a Resource ACL

This deletes ACL `MINE_JAN2003`.

```
delete acl mine_jan2003;
```

Delete a Generic ACL

This deletes a generic ACL `MINE*`.

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

Delete a Column ACL

This deletes a column ACL on `MINE_JAN2003.SALARY`.

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

Delete All Column ACLs on a Table

This deletes all column ACLs on table `KBIKE`.

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

Delete All Column ACLs on All Tables

This deletes all column ACLs on all tables.

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

Delete a Catalog ACL

This deletes an ACL on the catalog RBIKE.

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

Delete a Generic ACL on Catalog Entries

This deletes a generic ACL on the catalog entries MYCAT.MY*.CATAMS.

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

ACL Security Examples**Overview of Security Examples**

If LIBACLINHERIT=YES is added to a LIBNAME definition, the ACL precedence of permission checks changes. In this case, the LIBNAME ACL is used to first give READ or WRITE access to the domain, and then to inherit ACLs for resources that are owned by the domain owner. When a user accesses resources in an owned domain by using LIBACLINHERIT=YES, the following precedence of permissions checks on the ACL resource:

- If user-specific permissions are defined on the object for the accessor, the accessor gets these permissions.
- If group-specific permissions are defined on the object for the accessor's group, the accessor gets these permissions.
- If LIBNAME ACL permissions are defined for the accessor, the accessor gets these permissions on the object.
- If LIBNAME ACL permissions are defined for the accessor's group, the accessor gets these permissions on the object.
- Otherwise, the accessor gets UNIVERSAL ACLs on the resource.

The following are examples using LIBACLINHERIT:

Below is a listing of the libnames.parm files that are used in the code examples, along with a listing 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

When the libname.parm option OWNER= is specified, 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 code example below 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 specific 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 which 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' in order 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 ;

```

The second way that the LIBNAME ACL can be changed is by using a user ID that has ACL Special privileges. In the example below, 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 of '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

If the LIBACLINHERIT parameter file option is turned on, the ACL precedence of permission checks changes. Turning on LIBACLINHERIT creates a LIBNAME ACL on the specified LIBNAME domain. The LIBNAME ACL grants users rights to all resources within the LIBNAME domain. When a LIBNAME ACL is created for a specified LIBNAME domain, the ACL precedence of permission checks becomes:

1. Check user-specific permissions first. If defined, the accessor gets these permissions.
2. If a resource is owned by the same ACL group as the accessor, the accessor gets the resource's GROUP permissions.
3. LIBNAME ACL permissions are used for domains where LIBACLINHERIT is turned on.
4. If the resource is owned by a different ACL group than the accessor, the accessor gets the resource's UNIVERSAL permissions.

The following is an example using 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

The SPD Server uses a general ID that is called 'anonymous'. Any person that 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, the SPD Server administrator must use the psmgr utility to add a user called, "anonymou" 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 does have the ability to 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

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 allow different groups access to the domain, 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 code below enables the following actions:

- Any user in same group as admin can read, write, or alter tables and modify the LIBNAME access to the domain.
- Users in group1 and group2 receive read access to the domain.
- Users in group3 and group4 receive 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 privileges to */
/* the domain                      */

        data d1.admin1_table2 ;
            do i = 1 to 100 ;
                output ;
            end ;
        run ;

/* Generic ACLs allow all users to */
/* read tables created by admin1    */
/* unless a specific ACL is placed */
/* on a resource                    */

        PROC SPDO library=d1 ;

/* Assign who owns the ACLs */

        set acluser admin1 ;

```

The two ACL commands in the code below give read privileges to members of the ACL group 'ADMIN1' for any table that is created by admin1, who has read access to the domain.

This ACL is a good example for data marts and warehouses which DO NOT contain sensitive data. A GENERIC ACL gives broad access to tables in a domain. Generic ACLs must be used correctly (or not at all) if sensitive data needs to be restricted 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 code below performs the following:

- Gives read access of admin1_table2 to group1.
- Gives the admingrp read and control of admin1_table2
- Takes precedence over the generic read ACL, which prevents users that are not granted specific access to admin1_table2 from reading, writing, altering, or controlling the table.

```

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

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 individual users. The example below shows how to provide the different groups of users access to the domain owned by the user ID "Admin", and then extends the access to the tables. This 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 permits the following:

- Any user ID in admingrp receives read/write/alter access to the domain
- Any user ID in group1 or group2 receives read access to the domain
- Any user ID in group3 or group4 receives 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 */
/* include 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_;

```

Bringing a Table Offline to Refresh

When it is time to refresh the table, the first step is to revoke read privileges 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 ;
```

Modify the table ACL in the following ways:

- Revoke read and control by user IDs that are in the same group. This prevents locks during table refreshes.
- 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 will continue running and can finish under the lock. You can also use the special PROC SPDO operator commands to identify any users that might be running unattended jobs, and disconnect them so 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) ;
```

Now, modify table ACLs to allow the user ID prod1 to perform table refreshes. Because user ID prod1 is part of 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 ;
```

There is no need to issue an add ACL command for prod1_table. Deleting a table or replacing a table does not delete the ACLs. The ACL for that table remains until:

- The table ACL is deleted using PROC SPDO delete syntax.
- The table is deleted and another user creates a table with the same name.

At that time, 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.

After the table has been refreshed, the ACL can be modified to allow read access once 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 ;
```

Bringing a Domain Offline to Refresh Tables

When it is time to refresh the table(s), one approach to minimize contention and table locking is to revoke privileges of users and groups who will not be involved in the refreshing of tables in the domain.

This example assumes that the tables are already loaded in the domain and that the groups who use them have access.

```
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 ;
```

It is possible to revoke read access at the LIBNAME or domain level, which allows the IDs that are used to refresh the warehouse 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).

By doing this, the production IDs have full control over the tables and resources.

Note: Any user that is currently accessing the domain will continue to have access until they are disconnected. This can cause a lock to occur. The PROC SPDO special operator commands can be used to identify the user and disconnect the process so the refresh can take place.

```

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

SPD Server user IDs have two levels, 0 through 3 and 4 through 7. Level 4 through 7 user IDs can log in as an SPD Server 'super user' that can:

- access any table
- change table ACLs
- disconnect users
- perform administrative functions in a pinch

In many ways, SPD Server super users must be able to take on database administrator functions. The SPD Server super user cannot change the ownership of a table but they can assume the identity of the table owner to do required work. Often, this function happens in a pinch when a user needs access and the table owner or domain owner is out of the office.

The following should be considered when giving a user SPD Server super user status:

- The user must be trusted, because 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.
- SPD Server super users must be knowledgeable about the data and the database users' needs.

Assume the table **user1_table1** is loaded, and only read permissions have been given to users in group1. User4 is a member of group4, and group4 does not have read access to the table. User1 is the owner of **user1_table1** in domain d2. User1 is on vacation and user4 has been given an assignment which requires read access to the **user1_table1** to create a report for management.

Management has approved user4 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

The goal of column-level security is to allow only privileged users to access sensitive columns of tables that other users cannot read.

```
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';
```

```

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

Symbolic Substitution	187
Symbolic Substitution Row Level Security	187
Symbolic Substitution Example	187

Symbolic Substitution

SPD Server SQL supports symbolic substitution of the user's User ID using @SPDSUSR, group using @SPDSGRP, and whether the user is ACL Special using @SPDSSPEC in SQL queries. When the query is parsed, @SPDSUSR will be replaced by the User ID, @SPDSGRP by the group, and @SPDSSPEC will be "true" if the user has ACL Special privileges. The right hand side of symbolic substitution statements must be in all upper case text. Consider the example, "@SPDSUSR" = "SOMEUSER".

Symbolic Substitution Row Level Security

A powerful use of symbolic substitution is deploying row level security on sensitive tables that use views. Suppose there is a sensitive table that only certain users or groups can access. The administrator can use symbolic substitution to create a single view to the table that provides restricted access based on user ID or groups. The administrator could give universal access to the view, but only users or groups that meet the symbolic substitution constraints will see the rows.

For another example, imagine a table that contains sensitive information has one column that contains group names or user IDs. The administrator can use symbolic substitution to create a single view that allows users to access only the rows that contain his user ID or group. The administrator could give universal access to the view, but each user or group would be allowed to only see their user or group rows.

Note: 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 long. If the table column that contains user IDs or group names is not at least 8 characters wide, symbolic substitution will not function.

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 SPD Server Passwords, Users, and Table ACLs

Introduction	191
The Password Manager Utility psmgr	191
Overview of the psmgr Utility	191
Invoking the psmgr Utility	192
Converting to a psmgr 4.x Table	192
Adding New Users with psmgr	193
psmgr Commands	195
psmgr Command Details	195
Using a File as Input to psmgr	202
SAS Management Console	203
Overview of SAS Management Console	203
LDAP User Authentication	203
LDAP Authentication Notes	203
Overview of LDAP Authentication	203
Configuring LDAP Authentication	204

Introduction

SPD Server user data is maintained in an internal SPD Server password table. Each record in the password table describes the specific attributes and capabilities that are associated with an individual user. SPD Server uses two types of user authentication. The first type of user authentication is LDAP (Lightweight Directory Access Protocol). The second type of user authentication is traditional authentication, which SPD Server performs internally. LDAP authentication is performed by an LDAP server that runs on the SPD Server host machine. The section, “[LDAP Authentication Notes](#)” on page 203, contains more details on using LDAP with SPD Server.

The Password Manager Utility psmgr

Overview of the psmgr Utility

The **psmgr** utility manages the password table that enables access to the SPD Server host. When you start SPD Server, the command line option **-ACLDIR** specifies the

location (directory path) where the table is located. The owner of the password table, typically the SPD Server administrator, can update the table.

The password table 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 before disabling the user
- an optional user performance class

A user ID is restricted to 8 characters and does not have to correspond to any system user ID.

A password is also restricted to 8 characters.

A password for the **psmgr** table must have 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 a user's last six passwords. The password cannot contain the user ID.

If a user has three consecutive failed attempts to connect to the SPD Server host, his or her user ID is no longer enabled. Until an administrator resets the user ID, the user will not be able to connect to the SPD Server host.

If you are upgrading to SPD Server 4.5 from SPD Server 3.x, the SPD Server 4.5 **psmgr** utility must be re-populated 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 table is located. (Or you can specify a password table that has not yet been created.)

For a UNIX system, use the command:

psmgr installdir/site

The command invokes the **psmgr** utility and specifies the directory path for the password table.

For a Windows system, use the command:

psmgr

The command invokes the **psmgr** utility and uses the directory where SPD Server was installed to access the password table.

Converting to a psmgr 4.x Table

With SPD Server 4.5, you must convert your SPD Server 3.x password file to 4.5 format. Converting your password file to 4.5 format enables you to use the same set of

active SPD Server user IDs in SPD Server 4.5 that you used in SPD Server 3.x. To convert your SPD Server password file to 4.5 format from 3.x format, do the following:

1. Start the SPD Server 3.x **psmgr** utility using your SPD Server 3.x password table.
2. Export your SPD Server 3.x password table.
3. Start the SPD Server 4.5 **psmgr** utility with a new table.
4. Import the exported file from step 2 into the new password table.

Example:

```
/Installldir3_0/bin/psmgr /Installldir3_0/site
```

```
Enter Command > export /Installldir3_0/site/oldtable
```

```
Enter Command > quit
```

```
/Installldir4_0/bin/psmgr /Installldir4_0/site
```

```
Enter Command > import /Installldir3_0/site/oldtable
```

This creates a **psmgr** table from an old format **psmgr** table that exists at **/installldir3_0/site**.

Adding New Users with psmgr

Overview of Adding New Users

Whether you want to create the password for the user, or you want the user to create his own password, there are two methods to set up a new user.

Add a New User Who Creates His Own Password

This two-part method requires that the new user knows how to use one of the following LIBNAME options to change his password: CHNGPASS=, or NEWPASSWD=. The first part adds a new user. Perform the following steps:

1. Enter the PSMGR command and the password directory. For example:
psmgr /SPDS/pwdir
2. Enter the ADD command and the user ID. For example:
add debby
3. Enter a temporary password for the user. For example:
temporarypassword
4. Re-enter the temporary password for accuracy. For example:
temporarypassword
5. Enter an authorization level number from 0 to 7, depending on the authorization level that you want to assign to the user. For example:
0
6. Enter the IP address of a client machine if you want to limit the user's access to the client machine, or skip this step by pressing **Enter**. For example:

11.21.1.217

7. Enter the number of days you want the password to be valid. For example:

30

8. Enter the group name if the user is part of an ACL group, or skip this step by pressing **Enter**. For example:

groupname

The user **debby** is added. The temporary password is good for one logon.

The second part changes the user's password. Change the user's password with the NEWPASSWD= option. For example:

```
LIBNAME mylib sasspds "spdsdata"
host="bubba"
serv="5200"
user="debby"
password="temporarypassword"
NEWPASSWD="abc123"
```

Or, change the user's password with the CHNGPASS= option. For example:

```
LIBNAME mylib sasspds "spdsdata"
host="bubba"
serv="5200"
user="debby"
password="xyz123"
CHNGPASS=YES
```

The user will be prompted for a new password.

Add a New User and Set a Password for the User

The second method adds a new user to the password table using the **psmgr** utility. The password expires immediately after the user is created. The SPD Server administrator creates a new password for the user and sends it to him.

1. The SPD Server Administrator issues the CHGPASS command. For example:

chgpas debby

2. The SPD Server administrator enters Debby's old password. For example:

oldpassword

3. The SPD Server administrator enters Debby's new password. For example:

newpassword

4. The SPD Server administrator re-enters the new password for accuracy. For example:

newpassword

The SPD Server administrator can now inform Debby of her new password. The new password expires in the number of days that was specified in step 7 of the previous section.

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, structuring each command so that no input is required.

The commands and operands are positional, and they must be separated by blank spaces. If you give 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.

psmgr Command Details

ADD

adds a new user to the password table.

Syntax

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

Note: The new user's password expires during the first login to SPD Server.

Arguments

username

the user ID of an SPD Server user, which is restricted to 8 characters. The first character of the username can be either an alpha or an underscore. The remaining characters can be either an alpha, numeric, 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.

privilege

an authorization level number from 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 table and override any ACL restrictions on SPD Server tables. You might want to restrict special privileges to only the SPD Server user ID and password for yourself, the SPD Server administrator.

ip_addr

a numerical IP address; or a dash (-), which indicates that no IP address is specified. Use the IP address to restrict the user's access to SPD Server to that specific IP address.

Note: The IP address is not verified.

expiretime

a password expiration time; or a dash (-), which indicates that no password expiration time is being specified. The expiration time requires the user to change his password before the specified number of days has expired. The value, which is specified in days, represents the number of days from today (the current day) that the password is valid.

group

the default group for the user; or a dash (-), which indicates that no default group is being specified. If specified, the group definition must exist, which means that it was created by a previous GROUPDEF command. Group affiliation can be changed by a GROUPMEM command.

timeout

a maximum amount of time that is allowed between successful logins before the account is no longer enabled, or a dash (-), which indicates that no timeout is being specified.

failures

the number of password failures; or a dash (-), which 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 specified as a number from 1 to 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 table.

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

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

USER	AUTHORIZATION	IP ADDRESS
bar	7	
foo	1	192.149.173.5

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

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, which is specified using numbers 0 through 7. See the argument in the **ADD** command for an explanation of the numbers.

CHGEXPIRE

changes the expiration date for a given 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.

exptime

a password expiration time. The expiration time requires the user to change his password before the specified number of days has expired. The value, which is specified in days, represents the number of days from today (the current day) that the password is valid.

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 table.

Syntax

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

Arguments

username

the name (user ID) of an SPD Server user that also exists in the password table.

IP Address

the new IP address from which the user must connect to the SPD Server host. The IP address must be specified numerically using the xxx.xxx.xxx.xxx format. 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 timeout period. The timeout period requires the user to successfully logon before the specified number of days has expired. The value, which is specified in days, represents the number of days from the last successful logon that the password is valid.

CHGPASS

changes the password for a user.

Syntax

```
chgpass username oldpwd newpwd
```

Arguments

username

the user ID of an SPD Server user.

oldpwd

the user's old password.

newpwd

a 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 6 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, which is specified using numbers 1 through 3. See the [class argument on page 196](#) in the ADD command for an explanation of the numbers.

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 table. If you do not specify !, you will receive a Y or N prompt to verify the deletion.

EXPORT

exports the current password table 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 table.

Description

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

What you see in the flat file is a representation of what is stored in the password table. When you have changes that affect many users, it might be easier to edit the flat file than to use the **psmgr** utility. After making changes in the file, you can use the IMPORT command to construct a new, modified password table.

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. The first character of the groupname can be either an alpha or an underscore. The remaining characters can be either an alpha, numeric, or underscores. The groupname argument verifies that the groups that are specified with 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 table. If you do not specify !, you receive a Y or N prompt to verify the deletion.

GROUPMEM

updates the ACL group list for a user ID.

Syntax

```
groupmem username groupname [groupname|""]  
[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 is specified becomes the default ACL group for the user. You can specify up to five groups.

Note: If you specify fewer than five ACL groups, the utility prompts for additional ACL groups (up to five). Press **Enter** for the remaining ACL groups if no more are required.

Note: If you use the groupmem command in batch mode, the syntax requires you to submit five groupname arguments. If you want to update the user ID with less than five ACL group members, replace the empty groupname arguments with “ ”.

GROUPS

lists all the ACL groups in the password table.

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, which was created with the EXPORT command, to the password table.

Syntax

```
import textfile
```

Arguments**textfile**

the name of the flat file to import that contains the user definitions to add to the password table.

Description

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

During the import, if the **psmgr** utility encounters an identical user name (SPD Server user ID) in the password table, it skips the line. The **psmgr** utility displays a message that states that the line was skipped.

LIST

lists the contents of the password table, or a specific user.

Syntax

```
list [username]
```

Arguments**username**

the user ID of an SPD Server user. If no username is specified, the entire password table 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. The RESET command resets 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 he can connect to a server.

Syntax

```
reset username newpwd newpwd
```

Arguments

username

the name (user ID) of an SPD Server user, up to 8 characters.

newpwd

a new password for the user. The new password can be up to 8 characters maximum. The new password must contain at least 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 for accuracy. **Note:** The new password expires immediately and must be changed with the **psmgr** CHGPASS command.

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.

Example

Here is a command file named pscmds:

```
authorize bar barpwd
add newuser newpwd1 newpwd1 0 - - - - -
list
quit
```

The command file contains the password barpwd for user **bar**. Because the command file contains user IDs and user passwords, you might want to secure access to the command file. In UNIX environments, you can secure access to command files using native UNIX file permissions.

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 the SAS Management Console. The SAS Management Console is an application that the SPD Server administrator can use to manage passwords and ACLs. Administrators that need to configure passwords and ACLs can choose between using the SAS Management Console or the traditional SPD Server management tools (PROC SPDO and the **psmgr** utility).

More extensive information about using SPD Server with the SAS Management Console is available in the section, [Chapter 7, “Administering and Configuring SPD Server Using the SAS Management Console,”](#) on page 75 .

LDAP User Authentication

SPD Server users can be authenticated using LDAP or the **psmgr** utility. LDAP Authentication is performed by an LDAP server that runs on the SPD Server machine. When you use LDAP authentication, the operating system handles password maintenance. LDAP authentication adds the benefit of operating-system-level security and convenience.

LDAP Authentication Notes

Overview of LDAP Authentication

Remember the following information when you use an LDAP server to perform SPD Server user authentication:

- 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, all LDAP parameters must be removed from the SPD Server `server.parm` file. You must restart SPD Server so that the changes to the `server.parm` file are read.
- When you configure SPD Server to perform user authentication using the LDAP server, the **psmgr** utility is still needed. When using 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 authentication.
- Users that connect to an SPD Server must have corresponding logon information on 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 8 characters or less.

- 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

Instructions and examples of how to configure SPD Server to use LDAP authentication can be found in the SPD Server [Chapter 4, “SPD Server Windows Installation Guide,”](#) on page 39 and in the SPD Server [Chapter 3, “SPD Server UNIX Installation Guide,”](#) on page 13.

Chapter 17

DICTIONARY.PWDB and DICTIONARY.ACLS

DICTIONARY.PWDB and DICTIONARY.ACLS	205
Overview of Dictionaries	205
Example - Listing the Users in the Password Database Using SQL Pass-Through	205
Example - Listing ACL Objects Using SQL Pass-Through	206

DICTIONARY.PWDB and DICTIONARY.ACLS

Overview of Dictionaries

In addition to dictionary information for tables and columns, SPD Server provides information about the users in the password database and the ACL objects available. The column definitions for DICTIONARY.PWDB and DICTIONARY.ACLS are as follows:

```
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

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 only get the result set 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:

```
select *
from connection
to sasspds
(select *
from dictionary.pwdb)
```

To select only the user name and last log in date, submit:

```
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 making 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 only get the result set for the users who are making a pass-through connection and not for all users.

Then, submit the following:

```
select *
from connection
to sasspds
(select *
from dictionary.acls);
```

To find any ACL objects where "Jones" is the owner, submit the following:

```
select *
from connection
to sasspds
```

```
(select *  
  from dictionary.acls  
 where owner = "Jones");
```


Chapter 18

Using SPD Server with an Internet Firewall

Using SPD Server with an Internet Firewall	209
Overview of Using SPD Server with a Firewall	209
Assigning SPD Server Ports That Require Firewall Access	210

Using 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 will allow 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

The SPD Server Name Server listen port can be specified using well-known port definitions that are declared in the operating system's services file, or by using the SPD Server command line interface to specify the listen port. In the services file, the **spdsname** specification corresponds to the listen port. The SPD Name Server listen port can also be defined for UNIX installations 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 will use the **spdsname** service entry.

SPD Server Listen Port and SPD Server Operator Port

The SPD Server listen and operator ports can be specified using well-known port definitions that are declared in the operating system's services file, or they can be specified using 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. The SPD Server listen and operator ports can also be defined in the **rc.spds** start-up script for UNIX installations. In a **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 a zero value, the SPD Server will by default use **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 will choose any available ports for listen and operator port functions. This is the normal mode of operation when SPD Server clients and servers run in environments that have no 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 will use 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 have no firewalls.

How many port numbers need to be reserved for SPD Server Base User proxy processes? Each SPD Server Base User Proxy process produces its own command port. The command port can be accessed via command-line specifications issued by an SPD Server client. The operator port for a command port can be accessed 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

SPD Server Auditing

SPD Server Auditing	213
Overview of SPD Server Auditing	213
Proxy Auditing	213
WHERE Clause Auditing	214
SQL Query Auditing	215

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 or SQL audit logging, or both. For more information about start-up options, see the Help section in the SPD Server Administrator's Guide on [“SPD Server Host Commands” on page 53](#).

SPD Server auditing provides a way to log access to SPD Server resources, or to log 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. These programs enable you to input the audit logs into SAS tables. Then, you can query the SAS tables to determine access to SPD Server tables and resources.

Proxy Auditing

Proxy auditing provides a means to determine access to SPD Server resources. The audit record contains the following information:

- the activity timestamp
- the primary path of the domain that contains the resource
- the LIBNAME of the domain
- the user ID of the SPD Server user that is accessing the resource
- the resource name
- the resource type

- the SPD User ID of the resource
- the SPD Group ID of the resource
- the resource operation type for librefs:
 - ASSIGN
- the resource operation type for tables:
 - DELETE
 - RENAME
 - OPEN
 - REOPEN
 - REPAIR
 - TRUNC
- the resource operation type for clusters:
 - CREATE
 - UNDOCL
 - ADDCL
- the resource operation type for a WHERE clause:
 - WHERE
- the resource operation mode for librefs:
 - ACCESS
- the resource operation mode for tables and clusters:
 - OUTPUT
 - INPUT
 - UPDATE
 - UTILITY
- access requested to a SPD Server resource by an SPD user
- access granted to the 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

WHERE clause auditing is enabled using the WHEREAUDIT option. The maximum size that can be allocated WHERE clauses is controlled by the WHAUDLEN option. For more information, see [“SPD Server Parameter File Configurations for Auditing” on page 122](#).

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 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 length of the query in characters
- the text of the submitted SQL query

The maximum size that can be allocated in the SQL log for an SQL statement is controlled by the SQLAUDLEN option. For more information, see [“SPD Server Parameter File Configurations for Auditing”](#) on page 122.

Chapter 20

SPD Server Table WHERE Constraints

SPD Server Table WHERE Constraints	217
Introduction to Table WHERE Constraints	217
Creating and Controlling Table WHERE Constraint	217
Table Constraint WHERE Clauses	218
Table Constraint Restrictions and Limitations	218
Example Table Constraint WHERE Clauses	218

SPD Server Table WHERE Constraints

Introduction to Table WHERE Constraints

SPD Server table WHERE constraints enable SPD Server table owners to associate a WHERE clause with a table so 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 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 affect 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 only grant update access 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 [“Symbolic Substitution” on page 187](#). The table constraint WHERE clause is applied any time a user accesses the table (including the table owner if the WHERE clause specifies it). Table owners should be careful to construct only WHERE clause constraints that allow 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 will not work 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 will only contain rows that were filtered by the WHERE constraints on the source table. The destination table does not inherit the WHERE constraints of the source table.

Example Table Constraint WHERE Clauses

Assume you have the following SPD Server users:

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 certain 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 and uses table WHERE constraints to determine which table rows a user or user group can see:

```
PROC SPDO lib=foo;
SET ACLUSER;
ADD ACL employees;
MODIFY ACL employees / read;
quit;
```

The following code creates a table WHERE constraint that allows **Bossman** to read any row of the table, but 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;
```

The following code creates a table WHERE constraint that allows **Bossman** or an ACLSPECIAL user to read any row of the table, but 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;
```

The following code creates a table WHERE constraint that allows **Bossman** to 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**, who 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;
```

The following code creates a table WHERE constraint that allows **Bossman** to 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;

```

The following code creates a table WHERE constraint that allows **Bossman** to read all rows. **William** can read rows for employees that belong to the West region. **Ed** can read rows for employees that belong to the East region. **Frank** can read rows for employees that 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

System Management

<i>Chapter 21</i>	
SPD Server Operator Interface Procedure (PROC SPDO)	223
<i>Chapter 22</i>	
SPD Server Index Utility Ixutil	233
<i>Chapter 23</i>	
SPD Server Table List Utility Spdsls	241
<i>Chapter 24</i>	
SPD Server Backup and Restore Utilities	245
<i>Chapter 25</i>	
SPD Server Directory Cleanup Utility	265
<i>Chapter 26</i>	
SPD Server Debugging Tools	273

Chapter 21

SPD Server Operator Interface Procedure (PROC SPDO)

Special SPDO Commands	223
Overview of SPDO Commands	223
SPDO Command Examples	224
LIBNAME Proxy Commands	224
Overview of Proxy Commands	224
LIBNAME Proxy Command Examples	225
Privileged OPER Commands	227
TRUNCATE Command and Example	228
Refreshing SPD Server Parameter and LIBNAME Files	229
The REFRESH Command	229
REFRESH Command Examples	229
Commands to Nonexistent Users	230

Special SPDO Commands

Overview of SPDO Commands

Note: For additional information on using PROC SPDO, see the chapter, "Controlling SAS Scalable Performance Data Server Resources with PROC SPDO and ACL Commands."

The following SPDO commands require that you have ACLSPECIAL= enabled for your SPDO LIBNAME connection.

To enable ACLSPECIAL=, you must first grant the SPD Server user ID ACLSPECIAL= access rights. Second, 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 the following SPDO commands:

```
SPDSCMD 'command'
```

This example runs the specified command in the context of the user ID for the spdsserv process that 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.

SPDO Command Examples

1. List the WORKPATH directory:

```
spdscmd 'ls /spdswork/*.spds';
spdscmd 'dir d:\spdswork\*.spds';
```

2. 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 will be listed.

SET USER userID [portnumber];

allows 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 will be 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 [userID threadID=#];

Once 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];

where 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

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

```
-----
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/qabigl_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 performs 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. With privileged OPER commands, you must first set yourself as the proxy operator by submitting the following command:

```
SET MODE OPER;
```

The SET MODE OPER command sets you as the operator of the user proxy that you are currently set to. There can be only one operator for a user proxy 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, the following commands can be submitted:

OPER CANCEL [/DUMP];

The OPER CANCEL command cancels and exits the user proxy. If the /DUMP option is specified for a non-locking 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;

The OPER DISCONNECT command 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, when the control socket disconnect is detected by the user proxy varies, with different results.

OPER INTERRUPT;

The OPER INTERRUPT command 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 an interrupt flag is detected, the user proxy 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 will be intact, and the user proxy will still be viable. Whether a job will be interrupted cannot be determined; it depends on the job that is currently running. To determine if 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 the OPER INTERRUPT command has been submitted, you can wait and check again later. If the tables need to be closed immediately, you can use the OPER CANCEL command to cancel the user proxy.

TRUNCATE Command and Example

The TRUNCATE command is a PROC SPDO command that allows 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;

```

Refreshing SPD Server Parameter and LIBNAME Files

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.

The syntax for the REFRESH command to refresh the libnames.parm file is:

```
REFRESH DOMAINS
```

The syntax for the REFRESH command to refresh the spdsserv.parm file is:

```
REFRESH PARMS
```

Each REFRESH operation completely refreshes and replaces the contents of the previous libnames.parm file or the 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 spds44 sasspds 'spds44'
      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 example:

```

/* 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 selecting 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 non-locking user proxy, and you submit an OPER command, you get the following message:

```
ERROR: Specified user <userID> with pid <Process-ID> no longer exists.
```

Either of these messages indicates 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

SPD Server Index Utility Ixutil

The Index Utility Ixutil	233
Ixutil Usage	233
Ixutil Options	234
Ixutil Examples	235
Create a Hybrid Index	235
Retrieve Disk Usage and Fragmentation Statistics	236
Retrieve Index Distribution Statistics	237
Reorganize the Index	238
Review Disk Usage Statistics	238
Create a Join Index	239
Generate Join Index Statistics	239
Delete the Join Index	240

The Index Utility Ixutil

The ixutil utility supports reorganizing an SPD Server hybrid index to improve query performance and minimize disk space. The utility also prints the disk usage statistics or the contents of indexes.

Ixutil Usage

```
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 in the same domain that can be used by the SPD Server Parallel Range Join optimization. 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 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. Pay attention to the average join row percentage, which indicates the average number of rows that are read by a parallel

join work unit. For example, a percentage of 75 indicates the parallel join work unit will use 75 percent of the rows it must read. The closer the percentage is to 100, the better the performance. The percentage will increase as the distribution of the data for the join column becomes more sorted.

ixutil -stats<indx1,indx2,...>**-dsn**<data set> **-libpath**<physical path> [**-dist**] ;

For a specified set of indexes that belong to a given table, print the disk usage statistics and segment list fragmentation statistics. 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 **-reorg** option of ixutil.

ixutil -runstats<indx1,indx2,...>**-dsn**< data set name> **-libpath**<physical path> [**-maxruns**<number>] ;

For a specified set of indexes that belong to a given table, print the run statistics for each index. Run statistics give an indication of how the values of a particular index are sorted in relation to their observation numbers. By default, ixutil run stats 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. The optional [**-maxruns**<number>] argument can be used to change from the default setting of 10 runs to any integer between 1 and 100. Ixutil run stats can be useful in constructing 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 -fixclustmem -dsn <data set name> **-libpath** < physical path>

Make cluster member tables accessible, if the CDF metadata is corrupted or deleted. Run this command for each member table in the cluster, and then remove the .cdf file from the directory path.

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 SPD Parallel Join 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, prints 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>
Specifies the number of parallel join work units for a join index. Parallel join threads will 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 containing the table.
 - help**
Prints the ixutil help menu.
- Note:* Index names are case sensitive and **MUST** be specified exactly as listed in the PROC CONTENTS output.

Ixutil Examples

Create a Hybrid Index

Assume there is an SPD Server domain named **my domain** 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;
```

The SAS program above creates a hybrid 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 4.5(TS M0)
Build(Feb 26 2009, 11:50:08)
Index File Utility
Copyright (c) 1996-2009 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:

- 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. This is 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 value segment lists. As this value increases, it can affect the query performance for 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 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
  4.5(TS M0) Build(Feb 26 2009, 11:50:08)
Index File Utility
Copyright (c) 1996-2008 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:

- the number of unique values in the index
- the number of non-unique values in the index
- the minimum number of segments that any value is in
- the maximum number of segments that any value is in

- the average number of segments that all values are in
- the average percentage of segments that all values are in

The distribution statistics can be used to determine the effectiveness of the index. The index will perform 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 in, the better the index will perform.

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
    4.5(TS M0) Build(Feb 26 2009, 11:50:08)
Index File Utility
Copyright (c) 1996-2009 by SAS Institute Inc, Cary NC 27513 USA
```

```
Reorg for Index x:
Reorg successfully completed
Ixutil completed successfully
```

Running the index utility program again to get the statistics shows that the segment lists for all of the values have been aggregated (the `avg_chunks_per_list` is 1.0) and the unused disk space has been freed (the `idx_garbage_bytes` is 0), resulting 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. It will also increase the locality of segment data for an index key. The combination of these will give the best query performance for the index.

Review Disk Usage Statistics

Use the **-stats** option once more 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
    4.5(TS M0) Build(Feb 26 2009, 11:50:08)
Index File Utility
Copyright (c) 1996-2009 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 there are SPD Server tables in a domain in directory `/tmp`. A user has created two tables, **Table1** and **Table2** that can be joined on 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
    4.5(TS M0) Build(Feb 26 2009, 11:50:08)
Index File Utility
Copyright (c) 1996-2009 by SAS Institute Inc, Cary NC 27513 USA

```

Ixutil completed successfully.

Generate Join Index Statistics

Now, get statistics on the join index that you created, using the `-statjidx` option of the **ixutil** command. The statistics are printed for each join range of the index, as well as 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 will be joined in the range for each table.

```

> ixutil -statjidx Table1,ID Table2,ID
    -libpath /tmp

```

```

SAS Scalable Performance Data Server
    4.5(TS M0) Build(Feb 26 2009, 11:50:08)
Index File Utility
Copyright (c) 1996-2009 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, range_pct= 0.00
+-Range 2
+----<Table1,ID>: sobs=50001 eobs=75000 (Sorted)
+-----unique_keys=25000, max_occurance=1
+-----obs=25000, joinobs=25000, range_pct=100.00
+----<Table2,ID>: sobs=-1 eobs=0
+-----unique_keys=0, max_occurance=0
+-----obs=2, joinobs=0, range_pct= 0.00
+-Range 3
+----<Table1,ID>: sobs=75001 eobs=100000 (Sorted)
+-----unique_keys=25000, max_occurance=1
+-----obs=25000, joinobs=25000, range_pct=100.00
+----<Table2,ID>: sobs=-1 eobs=0
+-----unique_keys=0, max_occurance=0
+-----obs=2, joinobs=0, range_pct= 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
  4.5(TS M0) Build(Feb 26 2009, 11:50:08)
Index File Utility
Copyright (c) 1996-2009 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 will join 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. The more sorted the table is by the join key, the fewer rows a range work unit will need 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 will be discarded by a parallel work thread.

Chapter 23

SPD Server Table List Utility Spdsls

SPD Server Table List Utility Spdsls	241
Description	241
Usage	241
Options	242
Return Values	243
Spdsls Examples	243

SPD Server Table List Utility Spdsls

Description

spdsls lists the contents of an SPD Server domain directory, or given an SPD Server table component file, lists all other component files for the table. There are three purposes of the list utility:

- to furnish a complete list of tables for each SPD Server domain that you want to include in a backup
- to provide, for a specified damaged or questionable component file, a list of all other component files for the table that might be affected
- to 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

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.

spdsls -c

For a specified component file (identified with a complete path), lists all other component files for the table. If you have a corrupted or deleted SPD Server table file, use this option to find all related component files that might be affected.

spdsls -info

For a specified SPD Server table in the LIBNAME domain, list information about the table. 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**

Will include the domain ACL files in the listing. The files contain the ACLs (Access Control Lists) for any SPD Server table in the domain.

-aonly

Will include 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 a corrupted or deleted SPD Server table file, 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 used with **spdsls -info**, lists the size of the accumulated component file in bytes.

-v

Includes the version number in the listing.

-verbose

When used with **spdsls -info**, the **-verbose** option includes detailed information about an SPD Server table, including the number of observations in the table, observation length, the size of the index segment, the partition size, and whether the table is compressed, encrypted, or is a cluster member.

-v6

Only lists SAS 6.x data sets.

-v8

Only lists SAS 8.x or SAS 9.x data sets.

<LibPath>

The complete path of an SPD Server LIBNAME directory.

<ComponentPath>

The complete path of a specified table component file.

[Table...]

The table(s) 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 **spdsls** return value is 0, the utility was successful. If the **spdsls** return value is 1, the utility was unable to complete 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 domain **/bigdisk/sas/data/public** as follows:

```
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.idx._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

SPD Server Backup and Restore Utilities

Introduction	246
Overview of the SAS Scalable Performance Data Server Backup and Restore Utilities	246
Using Utilities with SPD Server	247
SPD Server Utilities Path Requirements	247
Compatibility with Previous Versions	247
Privileged Access Protection	247
Spdsbkup - The SPD Server Table Backup Utility	248
Description	248
Important Details about Full Backup	248
Important Details about Incremental Backup	248
Return Values	248
Backup Requirements	249
Client Access to an SPD Server Domain	249
LIBNAME Created with BACKUP= Option	249
Example Libnames.parm Statements	249
Backup Usage	250
Backup Options	250
Backup Return Values	252
Backup Data File	253
Backup Data File Nomenclature	253
Backup Data File Extensions	253
Backup Table of Contents File	253
Backup User Messages	254
Confirm Your Backup Job Status	254
Successful Backup	254
Warning: Table Cannot Be Backed Up	254
Failed Backup	254
Spdsrstr - The SPD Server Table Restore Utility	255
Restore Description	255
Restore Requirements	255
Restore Syntax	255
Restore Options	256
Restore Return Values	257
Restore User Messages	257

Restore Usage Examples	258
Using PROC SPDO to Back Up and Restore SPD Server Tables	261
Back Up and Restore Table Indexes Using SPD Server Full Backups	262
Back Up and Restore SPD Server Table Indexes Using System Full Backups ...	262
Back Up and Restore SPD Server Table Indexes Using System Full Backups ...	262
Method 1 – Restore the Index Dynamically	262
Method 2 – Recreate the Index after the Table Is Restored	263

Introduction

The SPD Server Backup and Restore utilities have the ability to:

- perform a backup of an SPD Server table that is open for query access
- provide a detailed Help menu for each utility
- offer an enhanced user interface and user messages

Overview of the SAS Scalable Performance Data Server Backup and Restore Utilities

The standard file system backup and restore facilities that native operating systems provide are generally inadequate for backing up and restoring SPD Server tables. 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 an add, delete, or change of a record. A change in a table is normally 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 machine resources. For this reason, administrators frequently struggle with a dilemma: are incremental backups of large tables worth the resources they consume?

The Scalable Performance Data Server backup and restore utilities alleviate these problems because they 'sense' the table data. The backup utility is capable of a real incremental backup. That is, instead of backing up the entire table, the utility backs up only the records that changed after the previous table backup date. Further, if a later 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. The increased frequency realizes the ultimate goal of the utilities: to minimize possible loss or corruption of an SPD Server table for whatever reason. Moreover, the software gives you a choice for 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.

In summary, the SPD Server incremental backup and restore facilities, used with the full backup and restore capabilities of either SPD Server or your system, can furnish comprehensive backup and restore capabilities.

The SPD Server backup and restore utilities include:

- **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 created by the utility.

Using Utilities with SPD Server

SPD Server Utilities Path Requirements

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, and 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 4.5 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 4.5 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 4.5 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 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 using the SAS PROC SPDO **spds cmd**. The PROC SPDO **spds cmd** command requires similar access: either identical user IDs or ACLSPECIAL=YES user privileges. See [Usage Scenarios](#) for more information about running the SPD Server backup and restore utilities with PROC SPDO.

Access to the backup and restore utilities will be given to the special user "SPDSBKUP". Optional user and password options are available for the utilities that can be used to give access to a specific privileged user.

Spdsbkup - The SPD Server Table Backup Utility

Description

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

- 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 returned by the query
- compresses the record data
- stores the data in a flat data file so that the restore utility can use it later when restoring the tables

Important Details about 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 a full backup is restored, the table is created with those attributes, 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 will not be backed up, and a warning message will be sent to the log. The **spdsbkup** utility will continue to back up all specified tables.

Important Details about 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 an SPD Server incremental backup is restored, 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 **spdsbkup** return value is 0, the utility was successful. If the **spdsbkup** return value is 1, one or more data sets could not be backed up. In that case, examine your SAS log for warning messages. If the **spdsbkup** return value is 2, a critical error caused an early process termination. 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. Access by the client to the physical path of the domain can be direct or through a network connection.

LIBNAME Created with BACKUP= Option

Before a table is eligible for backup, you must create the SPD Server LIBNAME domain using the **BACKUP=YES** option in the parameter file. Here are two example LIBNAME entries from a data's server's LIBNAME parameter file, libnames.parm, to explain how domains are processed with and without the **BACKUP=** option:

Example Libnames.parm Statements

No Backup Statement and Can Backup Statement Examples

```
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 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**, has the **BACKUP=YES** option. As a consequence, tables created through this domain are eligible for backup.

When **spdsbkup** performs a backup, it checks every table in **/usr/foo/test**. However, based on our example parameter file entries, **spdsbkup** backs up only the eligible tables in **canbackup**. On the client connections created using pass-through or LIBNAME, the **BACKUP=NO** LIBNAME option can be used to override. Tables that are created when either of these is in effect will be able to be backed up.

Incremental Backups Requires 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 have two alternatives for the full backup:

- Use the system's full backup utility, and then inform **spdsbkup** of the system's last full backup date.
- Use **spdsbkup** to perform a full backup, if none has been done before.

After a full backup has been performed for an SPD Server table, you can then proceed with an incremental backup strategy. The first incremental backup saves all table changes that were performed after the last full backup date. Each successive incremental backup saves the changes that were made after the previous incremental backup.

Backup 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 or full backup of SPD Server tables.

- If a table DOES NOT have a pre-existing full backup, it performs a full backup of the table and sets the last full backup date.
- If the table DOES have a pre-existing full backup, it 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, it prints a warning message and the table is not backed up.
- If a full backup is available, it performs an incremental backup of the table using the later of the two dates: the last full backup or 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 will be 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, it 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.

Backup Options

-a

Include the domain ACL files in the backup.

-aonly

Only include the domain ACL files in the backup. No tables will be backed up.

- d
The SPD Server LIBNAME domain.
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
The prefix filename for the backup data file. This filename is concatenated with **_BK_ddmmmyyy_hhmmss.0.0.0.spds**. The complete name identifies it as an SPD Server backup file. If the backup file exceeds the system's file size limit, **spdsbkup** automatically extends the file, separating it into multiple backup files with a unique SPD Server filename extension (the "0.0.0" portion of the filenames will be different).
- fibfact <n>
Increase 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 file information block metadata space (fibspace). File information block 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, it 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
Does not save index information for SPD Server tables when performing full backups. When the table is restored, the restore utility does not create indexes. Note that the index itself is not backed up, only the definition of the index is.
- nv6warn
Suppresses the 'Cannot back up v6 data set' warning. **Spdsbkup** 4.5 can only back up SPD Server and SPD Server data sets. Attempting to back up earlier versions of SPD Server data sets results in a warning message unless the **-nv6warn** option is invoked.
- p
The user password.
- proj <dir>
The domain project directory.
- q
Runs **spdsbkup** in quiet mode, which outputs only error messages and warning messages during a backup operation.

- r
The number of times **spdsbkup** should retry accessing a table that is not available to **spdsbkup** because it is being updated. A table that is being updated cannot be backed up. **Spdsbkup** 4.5 pauses five seconds, then retries the table if it was unavailable during the previous access attempt. The default retry count is one.
- s
The port number of the name server. If this is not specified, the default value is *spdsname*.
- t
The date/time of the last full system backup for the table that are to be backed up.

When the **-t** option is used with **spdsbkup**, the utility performs a full backup only if a table was created after the specified date/time. Otherwise, it 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 the **-t** option is used 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, it 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.

The **-t** option cannot be used 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 ...]
The list of tables in the domain that you want to include in the backup. If no tables are specified, 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 Return Values

When the **spdsbkup** utility exits, it generates a return value. If the **spdsbkup** return value is 0, then the utility was successful. If the **spdsbkup** return value is 1, one or more data sets could not be backed up. In that case, examine your SAS log for WARNING information. If the **spdsbkup** return value is 2, a critical error caused an early process termination. Examine your SAS log for WARNING and ERROR information.

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, added to the filename, generates a unique backup file that indicates when the backup was performed. Because the suffix is unique, the same filename can be used 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, storing the excess data in additional files. The software identifies these files with a file extension after the date/time. (The SPD Server file extension is the "0.0.0" portion of the filename.) While the extensions for the files will vary, the date/time will be 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 normal 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 backed up table:

- Columns 1 - 32 contain the table name or ACL name if it is a domain ACL file.
- Columns 33 - 232 contain the backup filename.
- Columns 233 - 250 contain the last full backup date, using SAS datetime18. format.
- Columns 251 - 258 contain the incremental backup sequence number since the last full backup. For example, a value of 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 a 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, and 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 SAS format for the table of contents file is

```
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 acfs $1.;
```

After performing each SPD Server backup, you should append the resulting table of contents file to the SAS table of contents backup file. This is an important step because it saves the backup history, and assists later when restoring tables.

For example, if you want to determine which backup files you need to restore a specific table, using a last full backup with a known date, you could create the following SQL query:

```
select bk_file from foo.bkup_toc
where table = "dset"
and datepart(lastfull) >= 'ddmmmyyyy'd;
```

Backup User Messages

Confirm Your Backup Job Status

Three basic types of SPD Server **spdsbkup** backup user messages can appear in your SAS log:

- Successful Backup
- Warning: Table Cannot Be Backed Up
- Failed Backup: Error and Program Aborts

Successful Backup

If **spdsbkup** successfully backs up a table, it writes some notes to **stdout**, unless the quiet **-q** option is specified. The notes include useful summary information, such as the name of the table that was backed up, the number of observations that were backed up, and whether the backup that was performed was a full backup or an incremental backup.

Warning: Table Cannot Be Backed Up

If **spdsbkup** cannot back up a table, it will print a warning message that states why the table could not be backed up.

Failed Backup

If the **spdsbkup** utility detects a serious failure condition, it will halt and print an error message that states the reason for the failure.

Spdsrstr - The SPD Server Table Restore Utility

Restore Description

Spdsrstr is a restore utility that 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.

- 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.

Restore Requirements

Before you can use the **spdsrstr** utility to restore a table, the following must be true:

- The table to be restored must be identical to the table that was backed up. That is, the name and create date must match the name and create date of the backed up table.
- Incremental table restores must be performed in the same order as the incremental backups that were performed.
- The table must not have been modified between the incremental restore dates, assuring that the table is returned to the exact state at time of backup.
- The backup file (with any file extension) 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 will restore only the tables that meet the restore criteria.

Restore Syntax

```
spdsrstr -d <dom> -h <host> {-f <fullfile> | -e <extfile>} [-hash]
      [-r <count>] [-a | -aforce] [-aonly] [-n] [-q] [-s <serv>]
      [-u <user>] [-p <passwd>] [-proj <dir>] [table ...]
```

```
spdsrstr -v -d <dom> -h <host> {-f <fullfile> | -e <extfile>}
      [-s <serv>] [-u <user>] [-p <passwd>] [-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 indicating when the backup file was created, and whether it is a full backup, and if so, the number of indexes. Further, it lists for each backed up table, the name, backup sequence, and the number of columns and records that are contained in the table.

spdsrstr -v

Verifies all or selected tables from a backup file can be restored, but does not do the actual restore.

Restore Options

-a

restore the backed up domain ACL (Access Control List) files if they do not already exist.

-aforce

restore the backed up domain ACL files if they do not exist or overwrite the current files if they do exist.

Note: This option should be used when restoring multiple files with the **-e** option to ensure that the domain ACL files are consistent with the last file restored.

-aonly

Only restore 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 can be used 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 the order from oldest to newest as determined by the **ddmmmyyyy_hhmmss** component of the filename.

-f

<fullfile> The backup filename that contains the tables to restore.

Note: The filename must be the full filename (including its extension) 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

Does not create any indexes 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 outputs only error and warning messages during a backup operation.

- r
Specifies the number of times **spdsrstr** will retry accessing tables that are not available during a restore operation because they were in query/update mode. **Spdsrstr** cannot restore a table if that table is in query/update mode when **spdsrstr** accesses it. **Spdsrstr** pauses five seconds and retries the table if it was in query/update mode. **Spdsrstr** will retry the file **-r** times before the restore fails. The default retry count for **-r** is one.
- s
The port number of the name server. If the port number is not specified, the default value is **spdsname**.
- u
The user name.
- v
Verify 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 no tables are specified, all tables in the file are restored.

Note: The list of tables **must** be the last option specified in your **spdsrstr** command.

Restore Return Values

When **spdsrstr** exits, it generates a return value. If the **spdsrstr** return value is 0, the utility was successful. If the **spdsrstr** 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 **spdsrstr** return value is 2, a critical error caused an early process termination. Examine your SAS log for WARNING and ERROR information.

Restore User Messages

Successful Restore

If **spdsrstr** successfully restores a table, it writes some notes to **stdout**, unless the **-q** option is specified. The notes include useful information, such as the name of the table that was restored, the number of rows that were restored, and whether the restore that was performed was a full restore or an incremental restore.

Warning: Table Cannot Be Restored

If **spdsrstr** cannot restore a table, it will print an error message stating the reason for the failure. No tables are restored after the failure.

Failed Restore

If the **spdsrstr** utility detects a serious failure condition, it will halt and print an error message that states the reason for the failure.

Restore Usage Examples

Backup and Restore Utility Example Scenario

Here are some common SPD Server backup and restore utility examples. In our example scenario, the starting date for the backup cycle is Sunday, February 3, 2008. The weekly schedule includes

- a full backup of the SPD Server domain "test" every Sunday at 23:30
- an incremental backup of the domain at 23:30 the rest of the week.
- **Exclusive SPD Server Full and Incremental Backups**
- **SPD Server Incremental/Full Backups and System Full Backups**
- **Restoring a Single SPD Server Table**
- **Restoring an SPD Server Domain**

Example 1: Exclusive SPD Server Full and Incremental Backups

You can use SPD Server backup and restore utilities exclusively to perform full and incremental table backups and restores.

This example outlines the steps 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 will 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:

```
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 any ACL files in the domain. The table of contents file contains information for each table that was backed up.

2. Archive the SPD Server backup file and source 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 also backs up any ACL files that are in the domain.

Example 2: SPD Server Incremental/Full Backups and System Full Backups

You can use SPD Server utilities to perform incremental backups on data sets that you have previously archived, as well as to perform full backups on new data sets that have never been backed up. You can also back up your SPD Server data sets using a system utility from your native operating environment. Which one should you use? The advantage in using system full backups is that a system utility does not parse the data set, and therefore usually runs faster than the SPD Server utility when performing 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 you use to perform a full system backup of the domain "test" using operating system utilities once a week, then 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 **spds1s -l** to produce a listing of the tables that belong to the domain "test" in preparation for a full backup.

```
spds1s -l -a <physical_path_of_domain>
```

Run the operating system backup utility to perform a full backup of the domain tables and ACL files, and then archive the backup.

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. **Spdsbkup** then performs an incremental backup of tables that have changed since the last full system backup, and 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 for 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
```

There is no last full backup date 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, appending the date/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: Restoring 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, 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 dtime >= '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. Our 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 Thursday, February 7. Thus, the order of the statements required to restore the table are

```
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
```

Alternatively, you could use the **-e** option of **spdsrstr** and restore all of the files with one single 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: Restoring an SPD Server Domain

Use the following steps to restore an SPD Server domain named "test" that 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 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 dtime >= '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 domain "test":

```
spdsrstr -aforce -d test -h host -s serv -e backup
```

The **-aforce** option will cause the domain ACLs to be updated for each restore file, resulting in the latest backup of the ACLs being restored.

Using PROC SPDO to Back Up and Restore SPD Server Tables

You can use the SAS **PROC SPDO** **spds**cmd to run the SPD Server backup and restore utilities. There is one constraint: 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 **spds**cmd, the current working directory is the root directory of SPD Server. Output messages from 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**.

Currently, there is a limitation when using the **-aforce** option with PROC SPDO to restore on Windows. The **-aforce** option fails if ACLs exist and there are active connections to the domain that were specified using the **-d** option during the restore. ACLSPECIAL= connections to a libref must specify a domain that is separate from the domain where you are attempting to restore the ACLs (if the ACLs currently exist). Making ACLSPECIAL= libref connections that specify the domain where you are attempting to restore the ACLs will cause the ACL restore operation to fail.

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;
```

Our example creates the LIBNAME backup for 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.

```
spds
```

cmd '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, specify a run of the SPD Server restore utility to restore the domain to its last full backup state.

```
spds
```

cmd '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 recreate the indexes. This information is subsequently used if the table is fully restored, to be able to recreate the indexes.

The SPD Server full backup utility does not save the index data, only the information necessary to recreate the indexes when the table is restored. Therefore, when backing up table indexes, the saved information requires no additional overhead and little additional space.

If you must fully restore a table later, there are two methods available for restoring the indexes:

1. You can allow the SPD Server restore utility to recreate the indexes when the table is created. In this method, as each observation is added to the table, the index will be dynamically updated.
2. Alternatively, you can use the `-n` option of the SPD Server restore utility. The `-n` option suppresses index creation. After the table is fully restored, you can use PROC DATASETS or PROC SQL to recreate the indexes.

Back Up and Restore SPD Server Table Indexes Using System Full Backups

Back Up and Restore 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 available methods for restoring indexes from a system full backup:

- **restore the index dynamically as the table is restored**
- **recreate the index after the table is restored.**

What decides which method to use? You must balance the time and resources needed to back up the index against the time needed to re-create the index when the table is restored.

Method 1 – Restore 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 `spds1s` with the `-i` index option. The output lists component files for each table in the domain that is intended for full backup.

When restoring 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.

In summary, the first method trades the additional resources 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.

Method 2 – Recreate the Index after the Table Is Restored

If you use this method, you do not need to include the index files in the table's full backup. Thus, when running `spdsls` to list the component files for each table in the domain that you intend to back up, leave off the `-i` index option. The `spdsls` utility then outputs a list that excludes index files.

A cautionary note about method 2: If you do not save index information, you can experience problems when you attempt to fully restore the table. The reason: the table's metadata will have information about the index files that can be missing or out of date. As a result, the metadata no longer mirrors the table.

Before you can perform an incremental restore of the table, you must first repair the table metadata. To repair the metadata, use `PROC DATASETS` to modify the table and delete all of the indexes, then run `spdsrstr` to restore the table. After performing the incremental restores of the table, use `PROC DATASETS` again to modify the table and create the indexes.

The second 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 recreate the indexes fully, if the table must be restored.

Chapter 25

SPD Server Directory Cleanup Utility

Introduction	265
Using the Directory Cleanup Utility Spdsclean	266
Spdsclean Wildcards and Pattern Matching	266
Spdsclean Options	266
Spdsclean Options That Define Actions	266
Spdsclean Options That Modify Behavior	267
Spdsclean Examples	268
Cleaning WORKPATH Files on Your Server	269
Cleaning Residual Temporary LIBNAME Domain Files	269
Cleaning Specific LIBNAME Domains	269
Cleaning Other LIBNAME Domain File Classes	269
Cleaning WORKPATH and LIBNAME Combinations	270
Cleaning Log Files	270
Cleaning WORKPATH, LIBNAME Domain, and Log Files	270
Spdsclean Glossary	270

Introduction

You use the SPD Server cleanup utility **spdsclean** to perform routine maintenance functions on

- 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 so users can control the level of cleanup performed and control the behavior of elements used in the utility.

CAUTION:

The spdsclean command line utility should be used 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, permitting 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 you use to specify the location and names of tables to convert, and behaviors that you want to control during the conversion process.

The command line is as follows:

spdsclean <--options>

The order of options on the command line does not matter. 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.
- Using **.** or **?** characters in the search pattern will find a wildcard match to any single character in a LIBNAME domain name in the LIBNAME file.
- The ***** 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*'** will match the domains **'ATEST1'**, **'ATEST123'**, **'ATESTXYZ'**, **'CTEST1'**, and so on from a LIBNAME file. The **-domains** pattern **'test*'** will match 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 would need to enclose the wildcard pattern in quotation marks. The question marks ensure that the wildcard pattern matching occurs relative to the **spdsclean** command.

```
spdsclean -domains "?test*"
```

You can also disable command shell globbing for the execution of the **spdsclean** command.

Spdsclean Options

Spdsclean options fall into two classes. The first class of **spdsclean** options are options that define actions. The second class of **spdsclean** options are options that modify behavior. Examples for both classes of **spdsclean** options follow.

Spdsclean Options That Define Actions

The **spdsclean** utility uses the following option settings to define specific actions:

-parmfile parmFile

The **-parmfile** option to the **spdsclean** command runs cleanup on the specified SPD Server environment that is defined in the named SPD Server parameter file. The cleanup action empties all directory resources that are defined in the SPD Server parameter file. All files contained in the **WORKPATH=** path list are deleted. Options which modify **-parmfile** cleanup actions are described in [“Spdsclean Options That Modify Behavior” on page 267](#)

-libnamefile libnameFile

The **-libnamefile** option runs a cleanup on the SPD Server environment specified in **libnameFile**, the **LIBNAME** parameter file. The cleanup 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 this action. **Spdsclean** always deletes residual lock files left behind in the domain directory. The cleanup 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 **libnameFile**. For more information, see the description for the **-domains** option.

-logdir logPath

The **-logdir** option specifies the path for SPD Server to use when cleaning server log files. SPD Server searches the specified log path directories for **.spd slog** files. When **.spd slog** 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 aged less than or equal to the specified threshold are retained. For more information, see the **-LOGAGE** option under [“Spdsclean Options That Modify Behavior” on page 267](#).

Spdsclean Options That Modify Behavior

The **spdsclean** utility uses the following option settings to modify specific behaviors:

-all

The **-all** setting is equivalent to specifying options using:

```
-tmp -acl -lib11.
```

-tmp

The **-tmp** setting enables deletion of residual temporary files in the **LIBNAME** domain path list. Deletion enabled is the default setting for the **-tmp** variable.

+tmp

The **+tmp** setting disables deletion of residual temporary files in the **LIBNAME** domain path list.

-acl

The **-acl** setting enables deletion of ACL files in the **LIBNAME** domain path list. Enabling the **-acl** setting deletes **.spres11*** and **.sppro11*** files from the selected **LIBNAME** domains in the specified **-libnamefile**. Use the **-acl** setting with care, since it applies to all selected **LIBNAME** domains. Deleting the ACL files does not give broader access to a given resource. Deleting the ACL files restricts the access to the resource owner. The default setting for **-acl** is enabled, or **+acl**.

+acl

The **+acl** setting disables deletion of ACL files in the LIBNAME domain path list.

-lib11

The **-lib11** setting enables deletion of the domain state file, `.spdslib11`. The default setting is **+lib11**.

+lib11

The **+lib11** setting disables deletion of the domain state file. This is the default setting for the `lib111` variable.

-verbose

The **-verbose** setting is equivalent to specifying **-vwork** and **-vdomain**, enabling logging for resource cleanup from WORKPATH, system workspace directories, and LIBNAME domain directories.

+verbose

The **+verbose** setting is equivalent to specifying **+vwork** and **+vdomain**, disabling logging for resource cleanup from WORKPATH, system workspace directories, and LIBNAME domain directories.

-vwork

The **-work** setting enables logging for resource cleanup from WORKPATH and system workspace directories.

+vwork

The **+vwork** setting disables logging of resource cleanup for WORKPATH and system workspace directories. This is the default setting for this variable.

-vdomain

The **-vdomain** setting enables logging for resource cleanup from LIBNAME domain directories.

+vdomain

The **+vdomain** setting disables logging for resource cleanup from LIBNAME domain directories. This is the default setting for this variable.

-domains dompat1, [dompat2,]

Use the **-domains** option to specify a domain list. The domain list is a comma-separated list of domain names and wildcard matching patterns that builds the LIBNAME domains from the LIBNAME file when it is processed. Standard pattern matching rules and wildcards apply.

-logage ageDays

Use the **-logage** option to set the age threshold, in days, for keeping `.spdslog` files in the SPD Server log directory when the **-logdir** option is specified. When the **-logdir** option is specified and the `.spdslog` files in the SPD Server log directory are older than the threshold `ageDays` value, they will be deleted. The default value for `ageDays` is seven days.

Spdsclean Examples

For the following examples, assume that the `InstallDir/` for your SPD Server is the directory `/opt/spds45`.

Cleaning WORKPATH Files on Your Server

This **spds**clean command cleans all of the files in the WORKPATH directory list designated by **/opt/spds45/site/spdsserv.parm**.

```
spds45clean -parmfile /opt/spds45/site/spdsserv.parm
```

If you want **spds**clean to log the files it deletes, add the **-verbose** option to the command.

```
spds45clean -parmfile /opt/spds45/site/spdsserv.parm -verbose
```

Cleaning Residual Temporary LIBNAME Domain Files

This **spds**clean command cleans all of the residual temporary files from all of the LIBNAME domains that are defined in the **-libnamefile** specified.

```
spds45clean -libnamefile /opt/spds45/site/libnames.parm
```

If you want **spds**clean to log the files it deletes, just add the **-verbose** option to the command.

```
spds45clean -libnamefile /opt/spds45/site/libnames.parm -verbose
```

Cleaning Specific LIBNAME Domains

This **spds**clean command cleans all residual temporary files from the LIBNAME domain TRIAL99.

```
spds45clean -libnamefile /opt/spds45/site/libnames.parm -domains trial99
```

Suppose that you want to add domain UJOE04 to be cleaned also. The following command will do this:

```
spds45clean -libnamefile /opt/spds45/site/libnames.parm -domains trial99, ujo04
```

Suppose you want to clean all TRIAL9x domains and all domains that begin with UJOE from the specified **-libnamefile**. The following command will do this:

```
spds45clean -libnamefile /opt/spds45/site/libnames.parm -domains trial9?, ujo*
```

To log the domains processed and the files deleted from each, just add the **-verbose** option to any of these **spds**clean commands.

Cleaning Other LIBNAME Domain File Classes

This **spds**clean command only cleans the ACL files from LIBNAME domains that begin with 'UJOE' that are defined in the specified **-libnamefile**. Because of the **+tmp** option, deleting residual temporary files is suppressed. To log the LIBNAME domains cleaned and the ACL files deleted, add the **-verbose** option.

```
spds45clean -libnamefile /opt/spds45/site/libnames.parm +tmp -acl -domains ujo*
```

To clean domain state files from domains TRIAL9x for the specified **-libnamefile**, submit the following **spds**clean command:

```
spds45clean -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

This **spds-clean** command cleans all of the WORKPATH files from the directory list specified in **-parmfile** and cleans residual temporary files from domain directories specified in **-libnamefile**.

```
spds-clean -parmfile /opt/spds45/site/spdsserv.parm
          -libnamefile /opt/spds45/site/libnames.parm -verbose
```

Logging occurs for the WORKPATH and LIBNAME domain directories and for the files that were deleted from each.

Cleaning Log Files

This **spds-clean** command cleans the .spdslog files from the specified **-logdir** directory that are more than seven days old.

```
spds-clean -logdir /opt/spds45/log
```

Suppose you want to keep log files older than 10 days from the date of execution. The following **spds-clean** command will do this:

```
spds-clean -logdir /opt/spds45/log -logage 10
```

If you want to see the files that were deleted, add the **-verbose** option to the **spds-clean** command.

Cleaning WORKPATH, LIBNAME Domain, and Log Files

This **spds-clean** command cleans WORKPATH files from the directory list in **-parmfile**, residual temporary files from domain directories in **-libnamefile**, and cleans .spdslog files that are older than seven days from the **-logdir** directory.

```
spds-clean -parmfile /opt/spds45/site/spdsserv.parm
          -libnamefile /opt/spds45/site/libnames.parm
          -logdir /opt/spds45/log -verbose
```

Spds-clean Glossary

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. Normally, 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 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.param File

The libnames.param file defines the SPD Server LIBNAME domains for the SPD Server environment. The libnames.param file is a collection of LIBNAME statements. Each LIBNAME statement defines a storage domain that SPD Server uses with clients. You modify the libnames.param file using the **-libnamefile** option with the **spdsserv** command.

Residual Lock File

When SPD Server accesses a data resource or table 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 cause no problem upon subsequent accesses 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 are named with a leading '\$' character, which prevent the residual temporary files from appearing in a PROC DATASETS directory listing. You should periodically remove old or abandoned residual temporary files that were created by unexpected proxy process terminations.

Spdsserv.param File

The spdsserv.param file defines the SPD Server operating parameters. The WORKPATH= statement in this file lists the directories that SPD Server will use for transient or working disk storage. To specify the spdsserv.param file, use the spdsserv command with the **-paramfile** 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 running. SPD Server normally cleans up these temporary files when exiting. If SPD Server terminates abnormally, these temporary files might be left in the temporary directory. In UNIX operating environments, the temporary files would usually appear in the 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

SPD Server Debugging Tools

Introduction	273
SPD Server 4.5 LIBNAME Statement Debug Options	273
DEBUG=	273
ALTPATH=	274
SPD Server 4.5 Server Parameter File Debug Options	274
ALTBINPATH=	275
RECORDFLAGS=	275

Introduction

SPD Server includes debugging tools that system administrators will find useful. The debugging tools allow SPD Server system administrators to create debug images and to evaluate test images that will not interfere with a pre-existing production SPD Server environment. The debugging tools are for use with SPD Server 4.5 running on SAS 9.2. The debugging tools are organized into LIBNAME statement options for debugging, and server parameter file options for debugging.

SPD Server 4.5 LIBNAME Statement Debug Options

When you issue a LIBNAME statement in SPD Server 4.5, the following debug options are available:

DEBUG=

Description

Enables or disables launching of a debug image for base SPD Server (spdsbased) and SPD Engine (spdsengd).

Syntax

Use the following arguments:

DEBUG= YES | CORE | NO

YES	launches a debug image (spdsbased and spdsengd) that will enable a good traceback.
CORE	launches debug images and sets the SPD Server parameter file option for that proxy. This ensures a good core file.
NO	disables the debug image for the specified proxy, if one is present.

DEBUG= Examples

Launch a debug image and ensure that the LIBNAME proxy creates a core file in the event of an unexpected process trap:

```
LIBNAME mylib sasspds 'spdsdata'
      user='denettee' debug=core;
```

Disable the debug image:

```
LIBNAME mylib sasspds 'spdsdata'
      user='denettee' debug=no;
```

ALTPATH=**Description**

Enables the use of an alternate binary path that is defined using the ALTBINPATH= option in the file. The ALTPATH= option will not search through 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 for SPD Server administrators who want to load a non-production copy of SPD Server (for example, testing a fix) without having to replace the production copy of SPD Server on a user basis.

Syntax

Use the following arguments:

```
ALTPATH= YES | NO
```

YES	enables use of the alternate binary path that is defined in as ALTBINPATH=.
NO	disables the alternate binary path for the specified proxy, if one is present.

ALTPATH= Example

Issue a LIBNAME proxy that uses the alternate binary path that is defined in ALTBINPATH=:

```
LIBNAME mylib sasspds 'spdsdata'
      user='denettee' altpath=y;
```

SPD Server 4.5 Server Parameter File Debug Options

SPD Server provides the following server parameter file options that system administrators can use as troubleshooting and debugging tools

ALTBINPATH=**Description**

The ALTBINPATH= option specifies the path to an alternate executable binary file directory. An alternate binary file path allows system administrators to load a non-production copy of SPD Server without having to replace 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'
```

RECORDFLAGS=**Description**

The RECORDFLAGS= option is used to control various startup and debug options for the SPD Server record level locking proxy. The default setting launches the optimized image.

Syntax

The RECORDFLAGS= option uses bit flag arguments for numeric values of 1, 2, and 4. To submit more than one argument, just submit the sum of the respective bit flags. For example, to choose option 1 and option 2, submit an argument of 3 (1+2=3). To choose options 1, 2, and 4, submit an argument of 7 (1+2+4=7).

```
RECORDFLAGS= 1 | 2 | 4
```

- 1 launches a debug image for the SPD Server record locking process. If RECORDFLAGS=1 is not specified, the default setting launches the optimized image.
- 2 loads the image that is specified in ALTBINPATH= for the SPD Server record locking process. If RECORDFLAGS=2 is not specified, the default setting uses the normal PATH setting.
- 4 produces a core file if the SPD Server record locking process encounters an unexpected exception while running. If RECORDFLAGS=4 is not specified, the default setting terminates the client connection if an unexpected exception is encountered while running.

RECORDFLAGS= Examples

Suppose you want to use the RECORDFLAGS= option to launch a debug image for the SPD Server record locking process (bit flag 1), and to produce a core file if an unexpected exception is encountered while running (bit flag 4). To perform both actions, submit the sum of the bit flags (1+4) = 5 as the argument for your RECORDFLAGS= option statement:

```
RECORDFLAGS= 5
```

Use the RECORDFLAGS= option to launch a debug image for the SPD Server record locking process (bit flag 1), to load the image specified in ALTBINPATH= (bit flag 2),

and to produce a core file if an unexpected exception is encountered while running (bit flag 4). To perform all three actions, submit the sum of the bit flags $(1+2+4) = 7$ as the argument for your `RECORDFLAGS=` option statement:

```
RECORDFLAGS= 7
```