SAS® Scalable Performance Data Server® 4.42

Administrator's Guide

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2007. *SAS® Scalable Performance Data Server® 4.42: Administrator's Guide*. Cary, NC: SAS Institute Inc.

SAS® Scalable Performance Data Server® 4.42: Administrator's Guide

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

1st printing, June 2007

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/pubs** or call 1-800-727-3228.

Table of Contents

- ACL Security Overview
- SPD Server ACL Security Model
  - Enabling ACL Security
  - Disabling ACL Security
  - ACL Security Examples
- Controlling SPD Server Resources with PROC SPDO and ACL Commands
- Symbolic Substitution
- DICTIONARY.PWDB and DICTIONARY.ACLS
- Managing SPD Server Passwords, Users, and Table ACLs

System Management

- SPD Server Operator Interface Procedure (PROC SPDO)
- SPD Server and SAS Management Console
- SPD Server Hybrid Index Utility Ixutil
- SPD Server Backup and Restore Utilities
- SPD Server Directory Cleanup Utility
- SPD Server Debugging Tools

SAS Scalable Performance Data (SPD) Server 4.4 Product Notes

Overview

This document summarizes enhancements and changes in SPD Server 4.4, including the SPD Server 4.42 and SPD Server 4.41 maintenance releases. The enhancements and changes that were in SPD Server 4.3 are included in this document in order to provide users with a chronology of changes and enhancements.

Pay special attention to the following notes for SAS System compatibility related to your SPD Server 4.4 media:

- The SPD Server 4.4 distribution CD-ROM comes packaged with client modules that are compatible with SAS 9.
- SPD Server 4.4 is not compatible with SAS versions earlier than SAS 9. Consult the appropriate UNIX or Windows installation guide for more information about SAS software requirements for use with SPD Server 4.4.
- For SAS 9.1.3 Service Pack 3 and earlier releases, you must **rename** the `sassqlu_for_sas913_sp3_and_earlier` modules from the SPD Server client installation to `sassqlu`. If you do not rename this module for SAS 9.1.3 Service Pack 3 and earlier releases, problems will occur with SPD Server implicit pass-through SQL that uses three part names. You will get an SQL parse error from SPD Server that causes the implicit pass-through SQL to fail.

What's New in SPD Server 4.42?

SPD Server 4.42, or SPD Server 4.4 TSM2, is an interim release. SPD Server 4.42 contains maintenance fixes and feature enhancements not found in SPD Server 4.41 and previous releases.

The following feature enhancements are provided in the SPD Server 4.42 release:

- The SQL reset option, PRINTLOG, logs SQL queries to the SPD Server log. For

more information, see the "Important SPD Server SQL Planner Options" section of the documentation chapter "Scalable Performance Data Server SQL Features," in the online SPD Server 4.4 User's Guide.

.

- SQL LIBNAMEs and record-level locking LIBNAMEs are supported. For more information, see the "LIBNAME Proxy Commands" section of the documentation chapter "Scalable Performance Data Server Operator Interface Procedure (PROC SPDO)," in the online SPD Server 4.4 Administrator's Guide.

- An SPD Server proxy manager utility is part of the SAS Management Console. The SPD Server Manager utility monitors SPD Server LIBNAME activity. For more information, see the "Proxy Manager" section of the documentation chapter "Administering and Configuring SPD Server Using the SAS Management Console," in the online SPD Server 4.4 Administrator's Guide.

- The SPD Server STARJOIN facility offers an IN-SET transformation. The IN-SET transformation allows you to use star schema processing when the star schema's fact and dimension tables have simple indexes on join columns. For more information, see the "SPD Server STARJOIN Optimization" section in the documentation chapter "Scalable Performance Data Server STARJOIN Facility," in the online SPD Server 4.4 User's Guide.

- A BY clause sort optimization is available for cluster tables if the member tables in the star schema are sorted by the BY variable. For more information, see the "Dynamic Cluster BY Clause Optimization" section in the documentation chapter "Scalable Performance Data Server Dynamic Cluster Tables," in the online SPD Server 4.4 User's Guide.

- Secure LDAP authentication is available for Solaris, AIX, and Windows. For more information, see the "SPD Server Parameter File Configurations for LDAP" section in the documentation chapter "Setting Up Scalable Performance Data Server Parameter files," in the online SPD Server 4.4 Administrator's Guide.

The SPD Server User's Guide and Administrator's Guide documents in PDF format can be viewed online at the following URL:

http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html.

SPD Server 4.42 Enhancements

- Cluster Create Option for Unique Indexes

Cluster Create Option for Unique Indexes

The CLUSTER CREATE command in PROC SPDO has a new optional argument that allows the user to specify whether unique indexes that are defined in the member tables should be validated and marked as unique in the cluster. If the UNIQUEINDEX option is set to NO, then unique indexes are not validated and the cluster metadata will not show the indexes as unique within the cluster. If the UNIQUEINDEX option is not specified, then it defaults to YES and the indexes are validated and marked unique within the cluster.

The usage syntax for the Cluster Create option is:

```
CLUSTER CREATE clustername
  MEM=member_table1
  MEM=member_table2
     ...
  MEM=member_table_n
  MAXSLOT=n
  UNIQUEINDEX=<yes|no>;
```

For more detailed information on PROC SPDO commands, see the chapter "SPD Server Operator Interface Procedure (PROC SPDO)," in the online SPD Server 4.4 Administrator's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html.

Additional Operator Interface Proxy Commands

The existing PROC SPDO (SPD Server Operator Interface Procedure) command set has new commands that gather proxy information about pass-through SQL librefs. The new commands, LIST USERS/LOCKING and SET USER/LOCKING, return information to the user about record-level locking proxies that are associated with pass-through SQL librefs.

The new privileged operator command OPER INTERRUPT provides the ability for certain users to interrupt long-running jobs. The new OPER DISCONNECT privileged operator command disconnects the proxy from its client. The OPER HALT and OPER RESUME privileged operator commands are no longer supported.

For more detailed information on Operator Interface Proxy Commands, see the chapter "SPD Server Operator Interface Procedure (PROC SPDO)," in the online SPD Server 4.4 Administrator's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html.

What's New in SPD Server 4.41?

SPD Server 4.41, or SPD Server 4.4 TSM1, is an interim release. SPD Server 4.41 contains maintenance fixes and feature enhancements not found in SPD Server 4.4 and previous releases.

The following feature enhancements are provided in the SPD Server 4.41 release:

- Indexes can be created on materialized views. For more information, see the "Materialized Views" section in the chapter "Optimizing SAS Scalable Performance Data Server Performance," in the online SPD Server 4.4 User's Guide.

- The std, avg, stderr, uss, css, and var GROUP BY functions are supported for use with fast index scans. All functions that can utilize index scans can also accept the DISTINCT term as well. For more information, see the "Index Scans" section in the chapter "Optimizing SAS Scalable Performance Data Server Performance," in the online SPD Server 4.4 User's Guide.

- The SPDSBKUP utility backs up MINMAXVARLIST information in addition to table column metadata such as FORMAT and LABEL. The SPDSRSTR utility restores the MINMAXVARLIST metadata with the table column metadata. For more information about SPDSBKUP and SPDSRSTR, see the chapter "SAS Scalable Performance Data Server Backup and Restore Utilities," in the SPD Server 4.4 Administrator's Guide.

- When you create a sorted table by using the ORDER BY clause with the CREATE TABLE SQL statement, the ORDER BY column in the newly created table is marked as sorted. Subsequent queries to the table that include an ORDER BY on the column will not cause the table to be re-sorted.

The SPD Server User's Guide and Administrator's Guide documents in PDF format can be viewed online at the following URL:

http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html.

What's New in SPD Server 4.4?

- SPD Server 4.4 User's Guide and Administrator's Guide
- SPD Server 4.4 Platform Support Changes
- SPD Server 4.4 New Features
- SPD Server 4.4 Enhancements

SPD Server 4.4 User's Guide and Administrator's Guide

The SPD Server 4.4 User's Guide and Administrator's Guide documents have been removed from the installation media. The documentation is available at the following URL:

http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html.

Placing the SPD Server 4.4 documentation on the SAS Documentation Page enables simple access via a bookmarked location on your preferred Web browser. Placing the SPD Server 4.4 documentation on the SAS Documentation Page also facilitates rapid distribution of periodic SAS documentation updates between successive SPD Server releases.

SPD Server 4.4 Platform Support Changes

New Platforms:

SPD Server 4.4 has added support for the UNIX Solaris x64 platform.

Platforms No Longer Supported:

SPD Server 4.4 no longer supports the Linux ia64 platform or the UNIX HP TRU64 platform.

SPD Server 4.4 and SAS Data Integration Studio

You can integrate the processing power of SPD Server 4.4 with SAS Data Integration Studio. The plug-in file that SPD Server uses to integrate with the SAS Management

Console can also integrate SPD Server resources into the SAS Data Integration Studio user interface.

To integrate SPD Server 4.4 functionality into the SAS Data Integration Studio user interface, copy the SPD Server 4.4 Java plug-in file into the SAS Data Integration Studio **plugins** subdirectory.

The SPD Server 4.4 Java plug-in file is located at:

SASROOT **/spdssmc/sas.smc.SpdsMgr.jar**

Note: *SASROOT* represents the path to the base directory of the SAS software installation on your client machine. *Spds44/* represents the installed SPD Server software directory. The name of the installed SPD Server software directory varies according to the specific version and release of your SPD Server software. For example, the path to your SPD Server Java plug-in file might begin with *SASROOT/spds44*, *SASROOT/spds44tsm1*, or *SASROOT/spds44tsm2,* depending on if you have the original SPD Server 4.4 software, or the first or second maintenance release of the SPD Server 4.4 software.

Copy the SPD Server 4.4 Java plug-in file to the SAS Data Integration Studio **plugins** directory:

SASROOT **/SASETLStudio/9.1/plugins/sas.smc.SpdsMgr.jar**

SPD Server 4.4 New Features

- Materialized Views
- SPD Server Profiling
- LDAP Password Authentication
- Dynamic Locking
- Surfacing Ports through an Internet Firewall

Materialized Views

A Materialized View will save the results of an SQL view in a temporary table. When the view is queried, rather than executing the view to determine the results, the temporary table is used. If any of the input tables to the view are modified, the Materialized View will dynamically update the temporary results. Materialized View is supported only via the SPD Server

SQL pass-through interface. A materialized view can result in significant performance gains for SQL queries that reference the view.

For more detailed information on materialized views, see the section "Materialized Views" in the chapter "Optimizing SAS Scalable Performance Data Server Performance," in the online SPD Server 4.4 User Documentation located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html.

SPD Server Profiling

A Server Profiling and Logger process is available to monitor and log the activity of the SPD Server processes. Once logged, the output can be formatted to be read into a SAS table for post analysis.

The SPD Server Manager utility found in the SAS Management Console connects to the Server Profiling Logger to provide real-time feedback of SPD Server process activity. The SPD Server 4.4 process profile panel will dynamically refresh SPD Server process activity such as memory and CPU usage. SPD Server processes are identified by their process ID, and for any proxy process the SPD Server user name that is associated with the proxy.

This feature is available only for SPD Server 4.4 (and later) installed on UNIX.

For more detailed information on SPD Server Profiling, see the section "SPD Process Profile Manager" in the chapter "Administering and Configuring SPD Server Using the SAS Management Console" in the online SPD Server 4.4 Administrator's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html.

LDAP Password Authentication

LDAP Authentication will cause SPD Server to authenticate a user password via LDAP rather than by the password in the PSMGR database. LDAP authentication will allow an SPDS User to have the same user/password as the users' UNIX/Windows logon, provided that the UNIX/Windows logon meets the SPD Server user/password character restrictions.

The administrator can select the mode of password authentication with server parameters; either via the PSMGR database or LDAP. Once selected all authentication will be done in that mode. With LDAP Authentication, a SPD Server user must still be entered in the SPD Server PSMGR database to maintain other information necessary for SPD Server, such as the user's groups and access level.

This feature is available only for SPD Server 4.4 installed on Windows or Solaris.

For more detailed information on SPD Server LDAP Authentication, see the chapter "SPD Server Password Manager Utility," in the online SPD Server 4.4 Administrator's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html.

Dynamic Locking

Dynamic Locking provides more flexible locking semantics on a domain which allows multiple clients to share both read and write access to tables in the domain without getting member lock failures. Dynamic Locking differs from SPD record level locking in that clients using dynamic locking connect to a separate SPD User Proxy process for each LIBNAME connection in the domain, versus record level locking where all users share the same Record Level Locking Proxy process. Having separate proxy processes versus the single record level proxy lessens the chance of hitting resource limits in the single process, and removes a single Record Level Locking point of failure for the Record Level Proxy.

Dynamic Locking can provide better performance than record level locking for some cases where concurrent reads and updates to a table are required, however
the performance benefit needs to be measured on a case by case basis.

For more detailed information on SPD Server Dynamic Locking, see the chapter "Accessing and Creating SAS Scalable Performance Data Server Tables," in the online SPD Server 4.4 Administrator's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html.

Surfacing Ports through an Internet Firewall

SPD Scalable Performance Data Server uses a client-server relationship, where the client cannot exist on the same host as the server. If the site has an Internet firewall, it is necessary to control the ports that the SPD Server and client use for communication such that those ports can be surfaced through the Internet firewall. Certain ports that the SPD Server uses are defined at startup time, and can therefore be easily controlled. However, ports are dynamically allocated to support each connection to the SPD Server and the subsequent user proxy process or processes that were created as a result of the connection. These ports are usually allocated as any available port. The MINPORTNO and MAXPORTNO server parameters are fully supported features in SPD Server 4.4. You can use the MINPORTNO and MAXPORTNO server parameters to control the dynamic ports that SPD Server uses.

For more detailed information on surfacing ports through an Internet firewall, see the chapter "Setting Up SPD Server Host Parameter Files," in the online SPD Server 4.4 Administrator's Guide, and the questions "How do SPD Server host and client processes communicate?" and "How do I know which ports must be surfaced through an Internet firewall?" in the chapter "SPD Server Frequently Asked Questions," in the SPD Server 4.4 User's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html.

SPD Server 4.4 Enhancements

Minmax Table Indexing for Character Columns

The SPD Server table option for MINMAXVARLIST= has been enhanced to also support character columns. The SPD Server WHERE planner will use the minmaxvarlist for a table to quickly determine whether a WHERE clause on the column can be quickly evaluated as trivially TRUE or FALSE.

For more detailed information on Minmax table indexing, see the chapter

"Optimizing SAS Scalable Performance Data Server Performance on Minmax Indexes," in the SPD Server 4.4 User's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html.

Expression Support for STARJOIN

The SPD Server STARJOIN optimization has been enhanced to be able to accept queries that previously could not use the optimization because they met the STARJOIN requirements except that a selected column was an expression rather than a simple column.

For more detailed information on expression support for STARJOIN, see SQL section in the chapter "STARJOIN Optimization," in the SPD Server 4.4 User's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html.

Dynamic Support for Larger Index Keys

The SPD Server Indexes can now dynamically support an index key up to 32,608 bytes long. An index key is the sum of the length of all of the columns that comprise the index. Previously, it was necessary to reconfigure the server BTREE_PAGESIZE option to support larger index keys. With dynamic sizing of the index metadata to support large index keys, this is no longer necessary and the BTREE_PAGESIZE server option is now obsolete.

SORTEDBY Specification for Dynamic Clusters

SPD Server now supports the SORTEDBY specification for columns that are defined on a dynamic cluster. In order to use the SORTEDBY specification, each member table in the dynamic cluster must have SORTEDBY specification set for the column. You specify the SORTEDBY setting on a dynamic cluster in the same way you would for a simple table.

```
PROC DATASETS library=libdomain;
modify clustername(sortedby=<var>);
quit;
```

The SORTEDBY specification assumes that the dynamic cluster was created using member tables that were added in the correct SORTEDBY order.

Additional Backup, Restore, and List Options

The SPD Server backup utility has added a -v option to provide verbose output. The -v option will log the full name of the backup file and table of contents file.

The SPD Server backup utility has added a -proj <dir> option to support backing up files in a domain project directory.

The SPD Server restore utility has added a -proj <dir> option to support restoring files to a domain project directory.

The SPD Server list utility has added an -s option to include the size (in bytes) of the component files listed.

The SPD Server list utility has added an -info option to get table information for a domain, including the number of component metadata, data, and index files for a table, and the accumulated size of the component files for a table.

For more detailed information, see the chapter "SPD Server Backup and Restore Utilities," in the online SPD Server 4.4 Administrator's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html.

Additional IXUTIL Options

The SPD Server IXUTIL utility has added the -crejidx option to create a join index, the -deljidx option to delete a join index, the -statjidx option to print join index statistics, and the -lstjidx option to list the join indexes in a domain.

For more detailed information, see the chapter "SPD Server Index Utility," in the online SPD Server 4.4 Administrator's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html.

What's New in SPD Server 4.3?

The What's New features for SPD Server 4.3 are included here to provide users with a chronological map of recent developments to the SPD Server feature set over the most recent releases.

SAS 9.1.3 Compatibility and Large Table Support

SPD Server 4.3 is compatible with the improved I/O infrastructure of SAS 9.1.3.

SPD Server 4.3 provides on-disk structures that are compatible with SAS 9 and the large table capacities that it supports. Enterprise-wide data mining often creates immense tables. In order to generate business intelligence quickly, the ability to update tables that contain billions of rows is more important than ever. Previous versions of SPD Server were based on 32-bit architecture that supported just over 2 billion rows and 32,768 columns. SPD Server 4.3 is based on a 64-bit architecture which supports tables with over *9 quintillion* rows and over *2 billion columns*.

The architectural differences between SAS 9 and earlier versions mean that SPD Server 4.3 cannot access SPD Server 3.x stores, and vice versa. For more information on sharing SPD Server 3.x and SPD Server 4.3 data stores, see the chapter, "SPD Server 3.x and SPD Server 4.x Compatibility" in the online SPD Server 4.4 Administrator's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html.

SPD Server 4.3 and SAS 9.1.3 Password Encoding

SPD Server 4.3 supports the integration of the SAS 9.1.3 PROC PWENCODE. This permits scripts to be generated that do not explicitly contain secure passwords that could easily be used without authorization. You must run PROC PWENCODE in Base SAS to enable the usage of script password encoding within SPD Server 4.3. See the Base SAS documentation for detailed instruction on running PROC PWENCODE for use with SPD Server 4.3.

The example below shows an SPD Server 4.3 LIBNAME statement that uses the password encoding option:

```
libname mylib sasspds 'spdsdata'
  server=kaboom.5200
  user='spdsuser'
  password='{sas001}c3BkczEyMw==';
```

SPD Server 4.3 and the SAS Management Console Utility

The 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

After installing the SPD Server 4.3 Java plug-in file, SPD Server administrators can use the SPD Server Manager utility in SAS Management Console to configure SPD Server 4.3 user/ group

passwords and ACLs instead of using the traditional SPD Server **psmgr** utility and PROC SPDO commands.

By default, SAS Management Console looks for plug-ins in the **plugins** subdirectory of each installed SAS product. The Java plug-in file that makes the SPD Server Manager utility available in SAS Management Console is located at

SASROOT /**spds43/plugins/sas.smc.SpdsMgr.jar**

Note: *SASROOT* represents the path to the base directory of the SAS software installation on your client machine. The Java plug-in file path above is specifically for SPD Server 4.3. The Java plug-in file for SPD Server 4.4 resides in a different location.

SPD Server 4.3 and SAS Data Integration Studio

You can integrate the processing power of SPD Server 4.3 with other SAS software tools, such as SAS Data Integration Studio. The same plug-in file that SPD Server uses to integrate with the SAS Management Console can be used to integrate SPD Server resources into the SAS Data Integration Studio user interface.

SAS Data Integration Studio is software that enables data warehouse specialists to create and manage metadata objects that define sources, targets, and the sequence of steps for the extraction, transformation, and loading of data into data marts or warehouses. SPD Server can be an excellent tool for managing the large tables of data associated with large data marts and warehouses.

By default, SAS Data Integration Studio looks for plug-ins in the **plugins** subdirectory of the SAS Data Integration Studios installation. To integrate SPD Server 4.3 functionality into the SAS Data Integration Studio user interface, copy the SPD Server 4.3 Java plug-in file into the SAS Data Integration Studio **plugins** subdirectory.

The SPD Server 4.3 Java plug-in file is located at:

SASROOT/**spds43/plugins/sas.smc.SpdsMgr.jar**

Note: *SASROOT* represents the path to the base directory of the SAS software installation on your client machine. *Spds43/* represents the installed SPD Server software directory. The name of the installed SPD Server software directory varies according to the specific version and release of your SPD Server software. For example, the path to your SPD Server Java plug-in file might begin with *SASROOT/spds43*, *SASROOT/spds43tsm1*, or *SASROOT/spds43tsm2,* depending on if you have the original SPD Server 4.3 software, or the first or second maintenance release of the SPD Server 4.3 software.

Copy the SPD Server 4.3 Java plug-in file to the SAS Data Integration Studio **plugins** directory:

SASROOT/**SASETLStudio/9.1/plugins/sas.smc.SpdsMgr.jar**

SPD Server 4.3 Utility Requirements

SPD Server 4.3 provides NLS functionality, or National Language Support for multiple languages and character sets in database operations. As a result, all SPD Server 4.3 utilities require access to the *InstallDir*/**bin64** directory, and you must ensure that the *InstallDir*/**bin64** directory is included in your SPD Server 4.3 PATH.

Here is an example of statement that specifies the necessary path:

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:InstallDir/bin64

export LD_LIBRARY_PATH
```

SPD Server 4.3 SQL Planner Enhancements

SPD Server 4.3 includes SQL Planner optimizations. SQL Planner optimizations improve the performance of the more frequent query types that used in data mining solutions such as Enterprise Marketing Automation. A key enhancement to the SPD Server 4.3 SQL Planner is optimizing correlated queries through the use of query rewrite techniques. Correlated queries are common in business and analytic intelligence data mining. Another significant enhancement is the tighter integration of the

parallel GROUP BY technology in the planner. The tighter integration adds performance benefits to nested GROUP BY syntax.

SPD Server 4.3 STARJOIN Facility

The SPD Server 4.3 enhanced SQL planner includes the new STARJOIN facility. The SPD Server 4.3 STARJOIN facility validates, optimizes, and executes SQL queries on data that is configured in a star schema. Star schemas are composed of two or more normalized dimension tables that surround a centralized fact table. The centralized fact table contains data elements of interest derived from the dimension tables.

For more detailed information on expression support for STARJOIN, see the SQL section in the chapter "STARJOIN Optimization," in the SPD Server 4.4 User's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html.

SPD Server 4.3 Index Scans

SPD Server 4.3 SQL provides users with the capability to use lightning-fast index scans on large tables. Rather than scanning entire tables that might have millions or billions of rows, in specific cases, SPD Server 4.3 SQL can use index data to resolve the query. Index data is compact, small, and faster to scan than an entire table. SPD Server 4.3 SQL provides enhanced index scan support for the following functions:

> **min, max, count, count distinct, nmiss**, and **range** functions.

For more detailed information on server index scans, see the section on Index Scans in the chapter "Optimizing SAS

Scalable Performance Data Server Performance," in the SPD Server 4.4 User's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html.

SPD Server 4.3 Optimized Correlated Queries

Intelligent storage must have the ability to interpret and process complex requests such as correlated queries. A correlated query is a select expression where a predicate within the query has a relationship to a column that is defined in another scope. Today's business and analytic intelligence tools often generate SQL queries that are nested 3 or 4 layers deep. Queries with cross nested relationships consume significant processor resources and require more time to complete processing. New algorithms in the SQL Planner of SPD Server 4.3 implement techniques that significantly improve the performance of correlated queries for patterns that permit query rewrites or query de-correlation.

SPD Server 4.3 Parallel GROUP BY

Parallel GROUP BY is a high performance parallel summarization of data executed using SQL. Parallel GROUP BY works against single tables used to aggregate data. Summarization tasks are common in data warehousing applications. Parallel GROUP BY was developed to speed up processor performance summarization tasks. Parallel GROUP BY is often used in SQL queries (through the use of sub-queries) to apply selection lists for inclusion or exclusion.

The parallel GROUP BY support in SPD Server 4.3 has been expanded in many areas. Parallel GROUP BY is integrated in the WHERE clause planner code so that it will boost the capabilities of the SPD Server SQL engine. Any section of code that matches the parallel GROUP BY trigger pattern will use it.

For more detailed information on server index scans, see the section on parallel GROUP BY in the chapter "Optimizing SAS Scalable Performance Data Server Performance," in the SPD

Server 4.4 User's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html.

SPD Server 4.3 Parallel Join

Parallel join is a high performance pairwise join of two SPD Server tables. The parallel join feature enhances join performance in two ways. First, SPD Server parallel joins are performed using parallel threading. Second, SPD Server parallel joins use enhanced data summarization methods after rows in a table are joined.

For more detailed information on SPD Server parallel joins, see the section on parallel joins in the chapter "Optimizing SAS Scalable Performance Data Server Performance," in the SPD Server 4.4 User's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html.

SPD Server 4.3 MINMAX Table Indexing

SPD Server 4.3 contains a new table option called MINMAXVARLIST=. The primary purpose of the MIINMAXVARLIST= table option is for use with SPD Server 4.3 dynamic cluster tables, where specific members in the dynamic cluster contain a set or range of values, such as sales data for a given month. When a SPD Server SQL subsetting WHERE clause specifies specific months from a range of sales data, the WHERE planner checks the min/max indexes. Based on the min/max index information, the SPD Server WHERE planner includes or eliminates member tables in the dynamic cluster for evaluation.

Use the MIINMAXVARLIST= table option with numeric columns. MINMAXVARLIST= uses the list of columns you submit to build an index. The MINMAXVARLIST= index contains only the minimum and maximum values for each column. The WHERE clause planner uses the index to filter SQL predicates quickly, and to include or eliminate member tables belonging to the cluster table from the evaluation.

Although the MINMAXVARLIST= table option is primarily intended for use with dynamic clusters, it also works on standard SPD Server tables. MINMAXVARLIST= can help reduce the need to create many indexes on a

table, which can save valuable resources and space.

For more detailed information on SPD Server parallel joins, see the section on Minmax Indexes in the chapter "Optimizing SAS Scalable Performance Data Server Performance," in the SPD Server 4.4 User's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html.

SPD Server 4.3 WHERE Costing Improvements

The WHERE clause Planner implemented in SPD Server 4.3 avoids computation-intensive operations and uses simple computations where possible. WHERE clauses in large database operations can be very resource-intensive operations. In SPD Server 3.x and earlier releases, query authors often needed to manually "tune" queries for performance. The tuning was accomplished using macro variables and index settings. The WHERE clause planner integrated into SPD Server 4.3 does the tuning work for the user by costing the different approaches to index evaluation.

For more detailed information on SPD Server WHERE costing improvements, see the section on the WHERE clause planner in the chapter "Optimizing SAS Scalable Performance Data Server Performance," in the SPD Server 4.4 User's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html.

SPD Server 4.3 Cluster Tables

SPD Server 4.3 uses a virtual table structure called a ***cluster table***. Why are cluster tables important? Clustered data tables provide a storage architecture that SPD Server 4.3 can take advantage of by using its parallel processing and data management capabilities.

A cluster table is a structure that can store multiple SPD Server tables. A cluster table is composed of members (or partitions). Each member can store a single SPD Server table. The cluster table uses a layer of metadata to manage the members. cluster tables can be used in WHERE clause costing as well. Each member in a cluster table is analyzed and assigned an EVAL strategy that best fits the data patterns in the member or slot. Using multiple EVAL strategies while performing WHERE clause costing on a cluster table provides better process granularity, which can improve

overall data throughput and performance.

Dynamic Cluster Tables

SPD Server Cluster Tables are virtual SPD Server table structures. SPD Server 4.3 cluster tables are a bound collection of multiple members. Each member is a standard SPD Server table. All member tables that belong to a given dynamic cluster table must share the same metadata formats and organization. SPD Server 4.3 cluster data tables use the metadata layer to manage the data that is contained in the member tables.

The SPD Server 4.3 dynamic cluster table structure provides an architecture that enables flexible loading, rapid storage, and parallel processing features for very large data tables. Using dynamic cluster tables, loading, removing data from, and refreshing very large data marts becomes easier and more timely. Dynamic cluster tables also provide organizational features and performance benefits that traditional SAS and SPD Server tables do not have.

For example, users can add new data or remove historical data from very large tables by accessing only the member tables that are affected by the change. It is possible to access the individual member tables in parallel. This strategy reduces the time needed for the job to complete and is accomplished using very simple commands. Furthermore, a complete refresh of a dynamic cluster table can take place using a fraction of the disk space that would be needed to refresh a large standard table that contained the same data.

For more detailed information on SPD Server dynamic cluster tables, see the chapter "Dynamic Cluster Tables," in the SPD Server User's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html.

Unsupported Features in Cluster Tables

Because of differences in the load and read structures for cluster tables, some standard features normally associated with Base SAS and SPD Server tables are currently not supported in SPD Server 4.3 cluster tables.

The non-supported features in Dynamic Cluster tables are:

- You cannot append data to a dynamic cluster table. To append data to a dynamic cluster table, the table must be unclustered, the data is appended to the individual unclustered files, then the unclustered files must be reclustered.
- Record-Level Locking
- SPD Server Backup/Restore Utility
- Copying data with PROC COPY or PROC SQL

If you have a task for a Dynamic Cluster table that requires one of the above features, you should undo the cluster and create standard SPD Server tables before continuing.

SPD Server 4.3 Random Placement of Initial Data Partitions in DATAPATH= List

In SPD Server 3.x, the first data partition files for all tables in the same domain started in the first DATAPATH= setting that was defined in the `libnames.parm` LIBNAME configuration file. Subsequent data partition files for that table would be placed in subsequent data paths. When all SPD Server datapath contain a data partition file, the process returns to the first datapath and continues in the same fashion. However, many SPD Server installations have many small-to-medium-sized tables that would not have data partition files in all of the available datapath. This strategy could unevenly balance the distribution of data on the disk, resulting with the first few datapath in a domain containing significantly more data than the last few datapath in the domain. The skewed data distribution results in unbalanced I/O.

In SPD Server 4.3, the first partition for tables in a domain is no longer assigned to the first DATAPATH= setting that was defined in the `libnames.parm` domain definition. Instead, SPD Server chooses randomly from the available data paths when placing the first partition for a large data table. As a result, the data is physically distributed more evenly and permits more balanced I/O within SPD Server processing.

By default, SPD Server 4.3 is configured to use random placement of initial data partitions among SPD Server data paths. The RANDOMPLACEDPF option is specified in the **spdsserv.parm** file. To disable random placement of initial data partitions in the DATAPATH= list, remove the RANDOMPLACEDPF option from your **spdsserv.parm** file.

Additional information on the RANDOMPLACEDPF option can be found in the SPD Server Administrator's Guide, Second Edition: Setting Up SPD Server Parameter Files, under spdsserv.parm file options.

SPD Server 4.3 Debugging Tools

SPD Server 4.3 includes debugging tools that system administrators will find useful. The debugging tools will allow SPD Server system administrators to create debug images and to evaluate test images without interfering with a pre-existing production SPD Server environment. The debugging tools are for use with SPD Server 4.3 running on SAS 9.1.3. The debugging tools are organized into LIBNAME statement options for debugging, and server parameter file options for debugging. enable SPD Server 4.3 users to create a debug image that will not interfere with the SPD Server environment.

For more detailed information on SPD Server debugging tools, see the chapter "SPD Server Debugging Tools," in the SPD Server 4.4 Administrator's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html.

SAS Scalable Performance Data Server 4.4
Pre-Installation and System Requirements Guide

AIX Requirements and Tuning for 64-bit SPD Server

System Requirements

- Required OS level: **5.1**
  (**Note**: *Critical* threading problems which affect data integrity were found in release 5.1.)
- Minimum System Configuration: RS/6000 system with minimum 1 Gb memory

Kernel Tuning Requirements

The following kernel parameters need to be adjusted on the RS/6000 system where you will run SPD Server.

- You must have root access to make these changes.
- In order to persist these changes across a system reboot, you will need to follow IBM recommendations for running the tuning commands at system startup time.
- For AIX 5.1 and earlier:
  `/usr/samples/kernel/vmtune -R 64`
  (-R 64 sets the read-ahead page threshold to 64 pages.)
- For AIX 5.2 and later:
  `ioo -o maxpgahead=64`
  (sets the read-ahead page threshold to 64 pages.)

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 2Gb memory.

Kernel Tuning Requirements

The following kernel parameters are for HP-UX 11.11 and HP-UX 11.23. They need to be adjusted on the HP server system where you will run SPD Server.

- You must have root access to make these changes via the sam utility (/usr/sbin/sam).

- After you make these kernel parameter changes, be sure to reboot the system before you attempt to use the SPD Server. In the following, MAX(a,b) means to take the maximum of the values a or b.

- ```
  dcb_max_pct = 10%
      dcb_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
  ```

- **Note:** SORTSIZE and INDEX_SORTSIZE are SPD Server parameters from the spdsserv. parm file. Increasing these SPD Server parameters may require adjusting the HP-UX kernel parameters accordingly. For more information on SPD Server parameters consult the SPD Server UNIX Installation Guide.

- Other HP-UX kernel parameters that may need to be increased depending on the way you use the SPD Server include:

    - **ninode** = Maximum open inodes in memory. Adjust for the maximum number of concurrently open SPD tables multiplied by the maximum number of partitions in an SPD Server table.

    - **nfile** = System-wide open file limit. Adjust for the maximum number of concurrently open SPD Server tables multiplied by the maximum number of partitions in an SPD Server table.

    - **nflocks** = System-wide file lock limit. Adjust for the maximum number of concurrently open SPD Server tables.

    - **maxfiles_lim** = Process hard limit for open files. Adjust for the maximum number of concurrently open SPD Server tables multiplied by the maximum number of partitions in an SPD Server table. The minimum recommended setting is 8192

Required Patches

The following HP-UX 11.23 for Itanium (IA-64) patches should be applied 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 Requirements and Tuning for 64-bit SPD Server

System Requirements

The following kernel parameter needs to be adjusted on Solaris server systems where you will run SPD Server.

- **rlim_fd_max** = Process limit for open files. Adjust the parameter to accommodate the maximum number of the number of concurrently open SPD tables multiplied by the maximum number of partitions in an SPD Server table. The minimum recommended setting is 8192

Windows Requirements and Tuning for 32-bit SPD Server

System Requirements

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

SPD Server 4.4 Client Requirements

System Requirements

- Required SAS level: **SPD Server 4.4 requires SAS 9.1.3**

SAS Scalable Performance Data Server UNIX Installation Guide

Note: Before installation, you should consult the SPD Server product release notes for important feature information about this release.

Before You Install: Precautions and Required Permissions

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

- Read the SPD Server Pre-Installation and System Requirements Guide document.

- SPD Server 4.4 is distributed only as a 64-bit environment application for Solaris by Sun, AIX by IBM, and HP/UX by Hewlett-Packard. The SPD Server media contains empty directory structures where 32-bit binaries resided in earlier releases.

- The SPD Server distribution comes packaged with SAS client modules that are compatible with SAS 9.1.3. SPD Server 4.4 is not compatible with versions of SAS earlier than SAS 9. You must make configuration changes to existing SAS 9.1.3 installations before the SAS clients can access the SPD Server environment. The section Configuring SPD Server Clients contains step-by-step information on the changes that must be made to existing SAS clients.

  Note: For SAS 9.1.3 Service Pack 3 and earlier releases, you must **rename** the `sassqlu_ for_ sas913_sp3_ and_ earlier` modules from the SPD Server client installation to `sassqlu`. If you do not rename this module for SAS 9.1.3 Service Pack 3 and earlier releases, problems will occur with SPD Server implicit pass-through SQL that utilizes three part names. You will get an SQL parse error from SPD Server that causes the implicit pass-through SQL to fail.

- SAS recommends that you use a UNIX user ID *other* than **root** to run your production SPD Server environment. While there are no known security or integrity problems with SPD Server 4.4, **root** access is not required to run the SPD Server environment. After you properly configure the UNIX directory ownership and permissions on your LIBNAME domains, there is no real need or benefit for **root** access to SPD Server. Notes for SPD Server Administrators contains more details and provides a list of options you can use to configure SPD Server resources under UNIX user IDs *other than* **root**.

- SAS recommends that SPD Server is installed in a location with that is adequately mirrored and /or backed up to assure reliability. The SPD Server installation location should use system space where the SPD Server Administrator for your organization has full rights. It's a good idea to keep progressive versions of SPD Server in a common area, but each successive installation should have its own dedicated directory. In the example UNIX directory listing below, the /**spdsmgr** folder contains six subfolders. Each subfolder contains the SPD Server software from a different SPD Server version and release.

```
tangoserver.unx.xyz.com> pwd
/users/spdsmgr

drwxrwxr-x 20 spdsmgr mis 512 Nov 12 2004 spds41tsm0
drwxrwxr-x 19 spdsmgr mis 512 Jan 15 2005 spds41tsm1
drwxrwxr-x 20 spdsmgr mis 512 May 23 2005 spds43tsm0
drwxrwxr-x 20 spdsmgr mis 512 Jul 21 2006 spds43tsm1
drwxrwxr-x 22 spdsmgr mis 512 Sep 27 2006 spds44tsm0
drwxrwxr-x 19 spdsmgr mis 512 Nov 11 2006 spds44tsm1
```

- SPD Server requires Base SAS software to run. You will need write access to the directories where SAS is installed on client computers. You must create directories on each SAS client and you will copy SPD Server client software to those directories. You will also need to modify SAS configuration files that are located in the root directory of the SAS installation on client machines. Because installation paths vary from machine to machine, this document uses the term *SASROOT* to represent the path to the base directory of the SAS software installation on your client machine.

- During SPD Server installation, you must create a directory named **/spds44** under *SASROOT*. You will be use the *SASROOT***/spds44** directory to hold SPD Server client files. You should not use the *SASROOT***/spds44** directory as the location for you SPD Server host installation, the base directory for the SPD Server UNIX installation. This document uses the term *InstallDir/* to represent the path to the *SASROOT***/spds44** directory on your machine.

- General familiarity with the UNIX language is required to install SPD Server 4.4. At a minimum, installers 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.

- You will need suitable access permissions to create the install directory for SPD Server on the file system where you install the server software. The owner of the SPD Server install directory should be the UNIX user ID of the SPD Server administrator. Notes for SPD Server Administrators contains more details.

- You will need write access to your server machine's

```
/etc/inet/services
```

  or

```
/etc/services
```

  file if you want SPD Server clients to connect to the SPD Server host using name services instead of specifying port numbers at invocation. Named services require you to define one or more registered ports which will use the services file appropriate to your machine.

- If your SPD Server clients will access the SPD Server host using named services instead of specifying port numbers, you will also need write access to the services files on the client computers, under the path

```
/etc/services
```

  or

```
/etc/inet/services
```

  For Windows, the path would be

```
C:\winnt\system32\drivers\etc\services
```

- You should 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 will be a RAID-structured volume with sufficient disk space to accommodate the transient storage needs for the SPD Server. The spdsserv.parm file is located in the root directory of your SPD Server host installation. The SPD Server User's Guide contains more detail on the WORKPATH= option and configuring servers for performance.

Unpacking SPD Server Distribution Files

Unpacking your SPD Server distribution files is another way to describe expanding an archived .tar file. A .tar file is one file that usually holds many compressed files. The SPD Server distribution file is a .tar file called SPDS.TAR. When you expand SPDS.TAR, you create your executable installation files and directory structures.

Restore the .tar file into an install directory that you have rights to on the computer that will run the SPD Server host application. For the purposes of this documentation, the directory on the UNIX file system where you plan to install the SPD Server software will be referred to as *InstallDir/*. Because computer topologies and paths vary, *InstallDir/* is a placeholder for the full path specification on the machine you install SPD Server on. When document instructions refer to *InstallDir/*, you should supply your computer's full installation path in its place.

Use the following commands to decompress the SPDS.TAR file on your SAS distribution CD into your *InstallDir/* directory. (Of course, you will need to mount the CD-ROM first.)

```
cd InstallDir
tar -xvf /cdrom_mount_point/SPDS.TAR
```

where */cdrom_mount_point* is a system-specific path that points to the CD-ROM drive containing the distribution files..

After expanding the SPDS.TAR file, you can perform a directory listing to see the files and directories that the .tar file created when it was expanded. The install process will make use of several subdirectories where SPD Server is installed.

Packing List for SPD Server Distribution

SPD Server 4.4 distribution media contains only contains 64-bit software on platforms that had both 32-bit and 64-bit software for SPD Server 3.x. SPD Server 4.4 for Solaris by Sun, AIX by IBM, and HP/UX by Hewlett-Packard have empty **bin32/** directories; the **bin64/** directories contains the 64-bit executable SPD Server files and appendages. SPD Server 4.4 for Solaris x64 and HP Itanium is only available in 64-bit software.

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

Note: *InstallDir/* represents the root directory where SPD Server is installed.

Note: The SPD Server 4.4 executable files for HP Itanium and Solaris x64 installations are stored in *InstallDir/***bin.** There are no bin64, bin32, lib64, or lib32 directories for Itanium or Solaris x64 installations.

In the **bin64/** directory you should expect to see the following binary files:

- **spdsnsrv** is the name server
- **spdsserv** is the SPD Server host
- **spdsbase** is the LIBNAME proxy
- **spdslog** is the message logger
- **spdsaud** is the audit logger
- **spdseng** is the SQL pass-through engine
- **ixutil** is the data set index utility
- **psmgr** is the password file utility
- **spdssnet** is the ODBC/JDBC/htmSQL gateway

- **spdsperf** is the performance and profiling server
- **spdsls** gives physical file listings for a LIBNAME domain
- **spdsbkup** performs full or incremental table backups
- **spdsrstr** restores full or incremental table backups
- **spqldrive** is a stand-alone SQL pass-through driver
- **spdsclean** is the SPD server disk cleanup utility.
- **spdsconv** is the SPD Server 3.x to SPD Server 4.4 table conversion utility
- **dulibv3** is the SPD Server 3.x 64-bit version of the shared library used by spdsconv, and is only included if your system previously supported SPD Server 3.x tables
- **spdsbased** is the debug version of spdsbase
- **spdsengd** is the debug version of spdseng
- **spdsnlslib** is the NLS support library
- **spdsnlslibd** is the debug version of spdsnlslib
- **spdsiotest** is the stand alone SPD Server I/O scalability test

The **lib32/** subdirectory is intentionally empty.

The **lib64/** subdirectory contains 64-bit library files.

The **lib/** subdirectory contains the appropriate SPD Server library files.

- **spdslib** is the runtime library that performs SQL pass-through from C/C++ applications to SPD Server.

The **bin32/** directory contains:

- **dulibv3** is the SPD Server 3.x 32-bit version of the shared library used by spdsconv.

The **samples/** directory contains various files of interest:

- **libnames.parm** is a sample SPD Server Host LIBNAME configuration file. Use with the -libnamefile option for the spdsserv command.
- **libsamp.parm** is a more sophisticated example of a LIBNAME configuration file.
- **pwdb** is a script to start up the password manager executable.
- **spdsserv.parm** is a sample SPD Server Host parameter file. It sets the recommended defaults for SPD Server options. Use with the -parmfile option for the spdsserv command.
- **rc.spds** is a Bourne shell script to startup a "standard" SPD Server environment.
- **rc.perf** is a Bourne shell script to start up a "standard" SPD Performance and Profiling Server.
- **killspds** is a shell script that kills all processes for a given UNIX user prefaced with the letters 'spds'. 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 also begin with the letters 'spds'.
- **killrc** is a shell script that kills all processes related to a run of rc.spds. Killrc is selective in that it will not kill spds processes not related to the core processes resulting from a run of rc.spds. The core processes are those which are initially started when rc.spds is run. They are usually spdsnsrv, spdsserv, spdsbase, spdslog, and spdssnet based on the rc.spds given in the samples directory.
- **doc_examples.sas** contains sample SAS code used in the SPD Server user's guide documentation. This is a good online reference which demonstrates SPD Server LIBNAME and data set usage and syntax options.
- **verify.sas** is a SAS installation verification job. You should run it after you install SPD Server.
- **spdsinst.sas** demonstrates simple use of WHERE clauses and WHERE planner output.
- **passthru.sas** demonstrates SQL pass-through usage. It gives examples of simple single level pass-through as well as secondary LIBREF and connection scenarios.
- **tempwork.sas** demonstrates temporary LIBNAME domain support. Files 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. The technique exploits a parallel load capability in the LIBNAME proxy.

- **aclcolrw.sas** shows the use of ACL row/column security features.
- **symbsub.sas** shows how symbolic substitution in pass-through SQL statements can provide row-level security in tables.
- **fmtgrpby.sas** shows how to use formatted parallel group-by statements in pass-through SQL.
- **scale.sas** can be used to benchmark the scalability of your SPD Server.
- **dynamic_cluster*.sas** shows how to use dynamic clusters with a minmax variable list.
- **minmax*.sas** shows how to use a minmax variable list on an SPD Server table.
- **paralleljoin*.sas** shows the use of the SQL Parallel Join performance enhancement.
- **starjoin*.sas** shows the use of the SQL star join performance enhancement.
- **index_scan*.sas** shows the use of the SQL index scan performance enhancement.
- **materialize_view*.sas** shows the use of the SQL materialized view performance enhancement.
- **process_perf_log** is a Perl script that will process a performance server log and server log into data that can be read into a SAS data set for post-performance analysis. The parameters are detailed in the script.
- **PerfDataSample.sas** is used to read in a processed performance server log into a SAS data set.

The **doc/** directory contains online SPD Server documentation for use by the SPD Server Administrator and users as desired. Documents are available in HTML (.html) format.

The **lic/** directory contains the SPD Server license file for your installation.

The **spds.lic** file is used by the name server validate SPD Server Hosts for their target hardware configurations. Once you obtain a valid SPD Server SETINIT for a particular machine, you will need to append it into this file.

The **msg/** directory contains SPD Server message files. The collection of ***.m** files are used by various SPD Server components to generate message text.

The **site/** directory is a storage directory for users' site-specific customization of their sample SPDS startup and configuration files. No SPD Server files are shipped in this directory. It is for customer use only.

The **spdssmc/** directory contains the SAS Management Console (SMC) files that support SPD Server.

SPD Server Component Directories for SAS Clients

When you expand the SPDS.TAR file into your *InstallDir/* location, it creates directories for each type of supported SAS client platform. The directory for each supported SAS client platform contains the components that SAS clients need to access SPD Server 4.4. The installation instructions will prompt you when it is time to copy files from the subdirectory for your SPD Server operating environment to a specified location on SAS computers that are clients to SPD Server 4.4.

For the comprehension of the installer, the following table lists the set of SAS client directories that are included on the SPD Server distribution media. The SAS clients are listed by platform :

SOLx64/ sasexe	contains Solaris x64 versions of the SAS System appendages for SPD Server installation.
SOLx64/ sasmsg	contains Solaris x64 versions of the SAS System message files for SPD Server installation.
SOL2/sasexe	contains Solaris2 versions of the SAS System appendages for SPD Server installation.

SOL2/sasmsg	contains the Solaris2 versions of the SAS System message files for SPD Server installation.
HPIA64/sasexe	contains HP Itanium 64-bit versions of the SAS System appendages for SPD Server installation.
HPIA64/ sasmsg	contains HP Itanium 64-bit versions of the SAS System message files for SPD Server 4.4 installation
HPUX/sasexe	contains HP/UX versions of the SAS System appendages for SPD Server installation.
HPUX/sasmsg	contains the HP/UX versions of the contains HP Itanium 64-bit versions of the SPD Server 4.4 installation.
AIX/sasexe	contains AIX versions of the SAS System appendages for SPD Server installation.
AIX/sasmsg	contains the AIX versions of the SAS System message files for SPD Server installation.
WIN/core/ sasexe	contains WIN32 versions of the SAS System appendages for SPD Server installation. These files have a .dll extension.
WIN/core/ sasmsg	contains the WIN32 versions of the SAS System message files for the SPD Server installation.

The **/sasexe** subdirectories for each platform contain operating system-specific versions of files needed by the SAS System to access SPD Server.

- **sasspds** is LIBNAME engine required to access the SPD Server environment from SAS 9.1.3.
- **sasspdo** is the SPD Server operator procedure required to access the SPD Server 4.4 environment from SAS 9.1.3.

The **/sasmsg** subdirectory for each platform contains an operating system-specific version of the following file:

- **spds.msg** is the SAS compatible message file for the SPD Server LIBNAME engine and SPDO operator procedure.
- For SAS 9.1.3 Service Pack 3 and earlier releases, you must **rename** the `sassqlu_ for_ sas913_sp3_ and_ earlier` modules from the SPD Server client installation to `sassqlu`. If you do not rename this module for SAS 9.1.3 Service Pack 3 and earlier releases, problems will occur with SPD Server implicit pass-through SQL that utilizes three part names. You will get an SQL parse error from SPD Server that causes the implicit pass-through SQL to fail.

Upgrading SPD Server 3.x with SPD Server 4.4

If you are **NOT** upgrading an existing SPD Server 3.x installation to SPD Server 4.4, skip this section and begin Configuring SPD Server 4.4 Server Software for Your Site. If you have SPD Server 3.x running on the same machine where you are

installing SPD Server 4.4, **complete this section before proceeding with your SPD Server 4.4 installation.**

If you are currently running SPD Server 3.x and you plan to install SPD Server on the same system, you should not have to shut down a running SPD Server to perform this installation. SPD Server can coexist with a running SPD Server 3.x system.

To configure SPD Server 4.4 with a concurrent SPD Server 3.x environment:

1. SAS clients that will access SPD Server 4.4 must be running SAS 9.1.3. See the Before You Install: Precautions and Required Permissions section for details.

   SAS clients that will access SPD Server will require some client components to be installed and some SAS configuration files will need to be modified on client machines. The instructions to modifying and upgrade existing SAS client installations are described in the section on Installing and Configuring SPD Server Clients. You can perform the modifications when the instructions direct you to.

2. Copy the following start-up and parameter files from your SPD Server 3.x *Installdir*/**samples/** subdirectory to your SPD Server *Installdir*/**site/** subdirectory. The files you must copy are **spdsserv.parm**, **libnames.parm** and **rc.spds**.

3. In order to use your existing SPD Server Version 3.x password file, use the **psmgr** EXPORT/IMPORT utility to create a new SPD Server password file. You will need to use the SPD Server 3.x **psmgr** to perform the EXPORT, and then IMPORT the old data base with your **new** SPD Server **psmgr:**

   - SPD Server 3.x **psmgr**:

     ```
     psmgr V3.x_old_site_dir
     export /tmp/spdspwd.dat
     quit
     ```

   - SPD Server **psmgr**

     ```
     psmgr InstallDir/site
     import /tmp/spdspwd.dat
     quit
     ```

4. Continue to Configuring SPD Server Host Software for Your Site.

Running SPD Server 3.x Concurrently with SPD Server 4.4

You may simply run SPD Server as a completely independent environment from SPD Server 3.x. This means two name servers and SAS clients must know whether they need to connect to the SPD Server Release 3.x name server to get their data, or whether they need to connect to the SPD Server name server to get their data. For more information on compatibility issues consult SPD Server 4.4 Compatibility Issues.

Converting SPD Server 3.x Tables to SPD Server 4.4 Tables

If you are converting an SPD Server 3.x environment to SPD Server 4.4 and do not plan on retaining the 3.x server, you may use the SPDSCONV utility to convert all existing SPD Server 3.x tables to SPD Server 4.4 tables. As a prerequisite to running the SPDSCONV utility, you must configure the LD_LIBRARY_PATH environment variable to contain the path of the **dulibv3** shared library. Here is an example that uses .ksh to configure the **dulibv3** shared library path, where *InstallDir* is a placeholder for the path to the SPD Server installation directory:

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:InstallDir/bin64
    export LD_LIBRARY_PATH
```

Copying SPD Server 3.x Data to SPD Server 4.4

With the latest version of SAS, it is not necessary to use the SPDSCONV utility to migrate SPD Server 3.x data for

use with SPD Server 4.4.

If you are using SAS 9.1.3 and SPD Server 4.4, you can access SPD Server 3.x tables using the sasspds3 client. (The sasspds3 client is shipped with SPD Server in the same location as the SPD Server 4.4 client.)

To migrate the data for use in the SPD Server 4.4 environment, copy the SPD Server 3.x tables to a location on the SPD Server 4.4 server. Before you copy the tables to the new server, you will need to issue LIBNAME statements for both new and old sasspds engines.

The code example below issues the LIBNAME statements, one for the SPD Server 4.4 engine and server and one for the SPD Server 3.x engine and server. After issuing the two libname statements, use standard SAS syntax to copy the tables to the location that you specify on your SPD 4.4 Server..

```
LIBNAME newspds sasspds 'newspds'
   server=host.new-port
   user='anonymous' ;

LIBNAME oldspds sasspds3 'oldspds'
   server=host.old-port
   user='anonymous' ;

/* SAS statements to copy data */
```

Configuring SPD Server Host Software for Your Site

After you unpack the .tar file from your SPD Server distribution CD, you must finish configuring SPD Server software to run on your server machine. It is important to note that this release of SPD Server 4.4 distribution includes only 64 bit components.

Itanium installations use only a **bin** directory; there are no **bin64** or **bin32** directories. Use **bin** instead of **bin64** or **bin32** when configuring an Itanium installation.

Complete the following steps to install the SPD Server system:

1. For 64-bit installations, add the *InstallDir*/**bin64** directory to your current PATH.
   For 32-bit installations, add the *InstallDir*/**bin32** directory to your current PATH.
   For Itanium and Solaris x64 installations, add the *InstallDir*/**bin** directory to your current PATH.

   For ksh users:

   ```
   export PATH=$PATH:InstallDir/bin64
   or
   export PATH=$PATH:InstallDir/bin32
   or
   export PATH=$PATH:InstallDir/bin
   ```

   For csh users:

   ```
   set path = ($path InstallDir/bin64)
   or
   set path = ($path InstallDir/bin32)
   or
   set path = ($path InstallDir/bin)
   ```

2. Seven files need to be copied from the *InstallDir*/**samples** directory to the *InstallDir*/**site** directory. The seven files are:

   ```
   rc.spds
   pwdb
   spdsserv.parm
   ```

```
killrc
libnames.parm
verify.sas
rc.perf
```

3. In the *InstallDir/***site** directory, edit the **pwdb** script file. Note: depending on the shell that you are running, you may have to make some minor changes to the script file. Itanium installations will need to change text instances of **bin32** or **bin64** in the **pwdb** script to just read **bin**.

   The **pwdb** file configuration must be modified to point to the INSTDIR= path, the path where your copy of SPD Server is installed.

   The INSTDIR= path in your **pwdb** file should be changed to

   ```
   INSTDIR=<explicit UNIX path to your SPD Server Installation Directory>
   ```

4. Invoke the **pwdb** script from the ***/*site** subdirectory to make an initial SPD Server password file. The password file is created in *InstallDir/* by executing the following command:

   ```
   pwdb
   ```

   First, use the psmgr **groupdef** command to define a group called **admingrp**. Next, use the psmgr **add** command to add an SPD Server user ID for yourself. (This is assuming that you are the SPD Server administrator). Both the **groupdef** and **add** commands will prompt you for values to enter. Use the following transcript file from a typical command sequence for a reference. You should notice that the password prompt does not echo any characters as you type. If you want to verify your work, you can use the psmgr **list** command to print the contents of the SPD Server password file following the **add** command.

   You should see something like this below:

   ```
   SAS Scalable Performance Data Server Host 4.40
   Password Manager Utility
   Copyright (c) 1996-2006 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>
   >
   User admin added
   Enter command
   > quit
   ```

These commands initialize the user password database.

You should add other user IDs before opening up the SPD Server system for broad use. The authorization level 7 shown above is privileged. Authorization level 7 allows users to circumvent desirable SPD Server ACL security measures. *Unlike the above example*, most or all users should be given authorization level 0 (non-privileged), so that SPD Server security cannot be bypassed. See Notes for SPD Server Administrators for more information.

Note: The admin password expires during the first logon to the SPD Server Host. Be sure to read the **psmgr** utility reference for more information about passwords.

5. In the *InstallDir/***site** directory, edit the **libnames.parm** file to add the site-specific LIBNAME domains your SPD Server will support. This step requires some thought and planning. You should decide how to organize your existing disk storage to best exploit the capabilities of the SPD Server. Refer to the SPD Server Host Reference below and the **libsamp.parm** file for more information. Also see the Help sections for more information on, "Using the SPD Server Name Server to Manage Resources," and, "Configuring Disk Storage for SPD Server," in the online SPD Server Administrator's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html.

6. Next you must 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:

   - Confirm settings for INSTDIR= pointers to your installation directory
   - Confirm settings for INSTDIR= pointers to your /bin directory
   - Specify whether or not to start up the SPDSSNET server
   - Check SNET port assignments if you use SNET
   - Reassign SNET ports if there are conflicts
   - Specify whether or not to create a log using LOGDIR=
   - Specify log cycle time and file prefix with LOGTIME= and LOGFILE=
   - Specify whether or not to create an audit file facility using AUDDIR=
   - Specify audit file cycle time and file prefix with AUDTIME= and AUDFILE=
   - Specify the location of your server user password database and parameter files

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

```
#!/bin/sh -x
# Sample startup script for SPD Server.
# This script starts the SPD Server Name Server
# SPD Server Host 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 SPD Server site directory and modify that
# copy just to make sure that a subsequent
# SPD Server software upgrade doesn't wipe out
# your site modifications to the script.
#----------------------------------------
#
# Define some primary variables. Hopefully
# you'll only need to customize only INSTDIR.
# NSPORT is the SPD Name Server listen port;
#        if omitted uses "spdsname" service entry.
# SNSPORT is the SPDS ODBC server listen port;
#        if omitted uses "spdssnet" service entry.
#
#
NSPORT=5190
SNSPORT=5191
INSTDIR=/usr/local/spds
```

```
PARMDIR=$INSTDIR/site
ACLDIR=$INSTDIR/site
LICDIR=$INSTDIR/lic
if [ -x /usr/bin/hostname ]; then ......
```

The **rc.spds** file configurations you need to examine are:

- **INSTDIR=:** You should change the line for INSTDIR= that is shown in highlighted text above. The location for the INSTDIR= setting should reflect the path of your installation. To follow the documentation example, the path should be changed to

  ```
  INSTDIR=/SomeDirectory/InstallDir
  ```

  You should substitute your own path specification.

- **INSTDIR/bin:** The PATH=, LD_LIBRARY_PATH, and LIBPATH= statements in the default **rc.spds** file refer to the INSTDIR/**bin** directory. Depending on your installation, you may need to specify INSTDIR/**bin32** or INSTDIR/**bin64** instead. The PATH=, LD_LIBRARY_PATH, and LIBPATH= statements are found in the following section of **rc.spds**:

  ```
  #
  # Define some secondary variables for server
  # parameter files
  #
     SPARM=$PARMDIR/spdsserv.parm
     LICFILE=$LICDIR/spds.lic
     PATH=$INSTDIR/bin
     export PATH
     MSGPATH=$INSTDIR/msg/
     export MSGPATH
     LPARM=$INSTDIR/site/libnames.parm
     LD_LIBRARY_PATH=$INSTDIR/bin
     export LD_LIBRARY_PATH
     LIBPATH=$INSTDIR/bin
     export LIBPATH
  ```

  For 64-bit installations, edit the INSTDIR/**bin** specification to read INSTDIR/**bin64**.
  For 32-bit installations, edit the INSTDIR/**bin** specification to read INSTDIR/**bin32**.
  For Itanium installations, the INSTDIR/**bin** specification does not need to be edited.

- **SNET:** **rc.spds** assumes you want to startup the SNET server (spdssnet) to support OBDC, JDBC or htmSQL access to SPD Server data stores. If not desired, you may delete or comment out the following lines that are located near the bottom of the 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:** **rc.spds** assumes you will be running SPD Server concurrently with an SPD Server 3.x

environment. It 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 if resources dictate otherwise, choose new port numbers. If you omit these assignments entirely, **rc.spds**uses the named services entries SPDSNAME and/or SPDSSNET. If you do not run the spdssnet server, you do not need to worry about the SNSPORT definition.

- **Logging:** **rc.spds** assumes you want to keep the logs resulting from messages written to stdout/stderr of the **spdsnsrv** (name server) and **spdsserv** (SPD Server Host) processes. The shell variable LOGDIR= defines the directory where these are kept. If you do not want to keep these, change to LOGDIR= and **rc.spds** will use **/dev/null**. If you want to keep the logs some place other than ***InstallDir*/log**, change LOGDIR=.

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

  ```
  LOGFILE=spdsserv
  ```

  Specifies log file prefix.

  ```
  LOGTIME=00:00
  ```

  Specifies time of day to cycle the log file.

  These settings enable automatic log file name 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 on these options, see SPD Server Host Reference. When this facility is enabled, the only messages that will go to the *InstallDir*/**log/spdsserv.log** file are those written to **stderr**. If you want to disable this capability, change the settings to empty pointers such as LOGFILE= and LOGTIME=.

- **Audit File Facility:** **rc.spds** is designed to allow you to use the audit file facility but this is not enabled by default. The following shell variables control this:

  AUDDIR=
  Use the AUDDIR= shell variable to specify the directory for the audit log files.

  AUDFILE=
  Use the AUDFILE= shell variable to specify the prefix for audit log files.

  AUDTIME=
  Use the AUDTIME= shell variable to specify the time of day (HH:MM) to cycle the audit log file.

  When AUDDIR= and AUDFILE= are set, you enable audit file creation. If AUDTIME= is set, automatic audit file cycling will occur at the specified time of day. For more information on the audit logging facility, see SPD Server Host Reference.

- **User Password and Parameter Files:** **rc.spds** assumes you will keep your **spdsserv.parm** parameter file and your SPD Server user password file in the *InstallDir*/**site** directory. If not, you will need to change ACLDIR= and/or PARMDIR= assignments. You can hook this script into your system startup file to be executed as part of booting the system. Otherwise, the SPD Server Administrator must manually start SPD Server after system startup.

- After you have finished making your changes, save and close your **rc.spds** file.

Note: The example **rc.spds** script provided in this step is a generic UNIX script. Some additional path fixes may be required for other operating environments. For example, Linux operating systems do not keep the **ps** and **grep** commands in **/usr/bin**, so corresponding changes will be required.

7. Assuming you wish to use registered ports for your SPD Server host, and you choose to use the default name server port of 5190 and SNET server port of 5191, add the following services to your **/etc/services** or **/etc/inet/services** file on the SPD Server host machine.

```
spdsname 5190/tcp # SPDS name server
```

Service declaration for the SPD Server name server

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

Service declaration for the SNET server.

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

If you choose to use different port numbers, replace the ???? strings below with unused 4-digit port addresses, and remember to update your **rc.spds** script accordingly. Determine unused port addresses by scanning the existing addresses and choosing a number that does not appear. Choosing a number greater than 5000 avoids conflicts with reserved and system-defined port addresses.

```
spdsname ????/tcp # SPDS name server
```

Service declaration for the SPD Server name server

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

Service declaration for the SNET server.

Note: If you have 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.

8. You are now ready to start up the SPD Server environment. Execute the *InstallDir/***site/rc.spds** script that you customized in the previous step. This starts up the SPD Server environment in the context of your current UNIX user ID.

   The **site/rc.spds** script customization is important because it defines the UNIX ownership and file access permissions on SPD Server resources. The ownership and file permissions are set in the context of the SPD Server runtime environment. If you plan to execute **rc.spds** from your system startup, the **rc.spds** script should be executed in the context of the proper UNIX user ID. Using the right UNIX user ID ensures that the resources created in the startup configuration meet the necessary file ownership and permission requirements for SPD Server.

9. The **rc.perf** script is an example script to start the Performance and Monitoring Server.

Verify SPD Server 4.4 Is Running

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

```
$ ps
PID TTY TIME COMD
834 pts/5    0:00 spdslog
847 pts/5    0:00 ps
820 pts/5    0:01 ksh
831 pts/5    0:00 spdsnsrv
832 pts/5    0:01 spdsserv
835 pts/5    0:01 spdssnet
836 pts/5    0:01 spdsbase
```

If the spds* processes are not running, check the log for errors. Unless you change the log file defaults in **rc.spds**, the log paths will be:

- *InstallDir*/**log/spdsnsrv.log**
- *InstallDir***log/spdsserv.log**
- *InstallDir*/**log/spdsserv_*mmddyyyy_hh:mm:ss*.spdslog**

If there were problems on start-up and any processes failed to initialize, terminate the remaining SPD Server processes before reinvoking the **rc.spds** script. Use the supplied **killspds** shell script located 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 upgrade from SPD Server 3.x to SPD Server 4.4, when you are satisfied with your SPD Server install, you should copy the **libnames.parm** file from your SPD Server 3.x location into your SPD Server location. The new **libnames.parm** file overwrites the temporary version which was created when you verified your SPD Server 4.4 installation. The new file also provides you with access to all of the SPD Server 3.x LIBNAME domains from your previous environment.

Configuring SPD Server Client Software

Note: SPD Server media contains SAS client software modules for 64-bit SAS 9.1.3 installations for these platforms: Solaris by Sun, AIX by IBM, Solaris x64, and HP/UX by Hewlett-Packard.

Once the SPD Server environment is configured and running, you need to complete some other installation functions on the SAS client platforms that will use SPD Server. This may include the system that is actually running the SPD Server, so some of these steps outlined here may have already been covered during installation of the SPD Server host software. If so, skip over the duplicated steps.

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

1. If you want to access SPD Server through a registered port (named service), add the following services to your **/etc/inet/services** or **/etc/services** file if not already done:

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

   The service defines the port number for the **spdsnsrv** (name server) process. Make sure the added port number matches the port number used during the SPD Server installation. If you are running SAS with 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, **sp44name**) or you can directly include the SPD Server port number in your SAS statements.

2. SPD Server is designed to be accessed from multiple platforms. For SPD Server clients running SAS on Solaris, Solaris x64, Windows, AIX, HPIA64, and HP/UX operating systems, you must create your SPD Server client file directory. Locate the base directory of your SAS software installation, and create a subdirectory called **spds44.** Find the directory on your SPD Server installation media that matches your operating system, and copy the contents into the */SASROOT* **/ spds44** directory that you just created. This document refers to the */SASROOT***/spds44** directory of the SPD Server 4.4 installation as *SPDClientDir*.

   Use the table below to match each platform with the corresponding component directory:

Platform	Subdirectory
Solaris2	SOL2
AIX/RS-6000	AIX
Windows/Intel	WIN/core
Solaris x64	SOLX64
HP/UX Itanium	HPIA64
HP/UX	HPUX

Enter the directory that corresponds to your client's operating system. (If you are using Windows, be sure to navigate to the WIN\core directory). All of the platform-specific directories contain two subdirectories called **sasexe** and **sasmsg**. Copy all of the files that you find the **sasexe** subdirectory and the **sasmsg** subdirectory into the *!SASROOT/***spds44** directory that you created on your SPD Server client computer. Remember that *!SASROOT/* is a place holder that this documentation uses for the full path specification to the directory that your copy of SAS is installed in.

3. Now that SPD Server client software is on your SAS computer, you need to tell SAS where it is by modifying your SAS configuration file, **sasv9.cfg**. Open the **sasv9.cfg** file with a text editor and insert both of the following lines into the **sasv9.cfg** file. Make sure that the new SPD Server directory is searched before any other -path() directories in your **sasv9.cfg** file. The -ins_msg() directory statement should immediately follow the -msg() statement in the **sasv9.cfg** file. Otherwise, SAS will not pick up the upgraded SAS System appendages and message files.

```
-path !SASROOT/spds44

-ins_msg !SASROOT/spds44
```

   If you want to launch the SPD Server client through a command line invocation instead of through your SAS configuration file, on UNIX systems you can use the following SAS invocation command.

```
-path !SASROOT/spds44 -ins_msg !SASROOT/spds44
```

4. The SPD Server can be accessed via the SAS/ODBC Driver version 9.0, SAS JDBC Driver, and SAS/htmSQL. Each of these drivers can be downloaded from the support link on the SAS Web at

   http://support.sas.com

   See, "Using SPD Server with Other Clients," in the online SPD Server 4.4 User's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html for more information on connecting and configuring these applications.

   ODBC client applications require installation of the spds.dll application extension. Install the ODBC client application extension as follows

   - Install the SAS/ODBC Driver Version 9
   - Copy

     ```
     InstallDir/WIN/core/sasexe/spds.dll
     ```

     into

     ```
     <drive letter>:\Program Files\sas\shared files\general
     ```

   - Configure an ODBC Data Source for direct SPD Server access. (For more details about configuring and using these clients, see, "Using SPD Server with Other Clients," in the online SPD Server User's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html.

5. SPD Server software includes support for the SAS Management Console (SMC). You must copy the necessary SMC files to the client system:

   - Ensure that the SAS Management Console support is installed. You can verify this by checking the **InstallDir/spdssmc** directory.
   - Copy

     ```
     InstallDir/spdssmc/sas.smc.SpdsMgr.jar
     ```

into

```
SMCInstallDir/plugins
```

where *InstallDir* is a placeholder for the directory your SPD Server software is installed, and where *SMCInstallDir* is a placeholder for the directory your SAS Management Console is installed.

Testing Your SPD Server Installation Using SAS

Testing your SPD Server installation is relatively simple. To verify, you only need to make two SAS LIBNAME assignments using the SPD Server LIBNAME engine. The examples in this section refer to the SASSPDS engine, the engine for SAS 9.1.3.

1. Startup 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. See the Notes for SPD Server Administrators section for more information on the **rc.spds** script and SPD Server LIBNAME files.

2. On a properly configured client system, invoke SAS and make the following LIBNAME assignments:

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

ServerNode is the server's node name and *port* is either the numeric value assigned to NSPORT from your **rc.spds** file, or the service name you use to access the SPD Server name server. If you used the sample **rc.spds**, your LIBNAME assignment would look like this:

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

If you use the spdsname service, your LIBNAME assignment would look like this:

```
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;
```

Verifying these simple statements confirms connectivity between the SAS client system and the SPD Server environment. Successfully performing these LIBNAME assignments means that the network configuration is correct and that most of the SPD Server configuration is correctly in place.

Substitute *serverNode* with the node name that runs the SPD Server environment you want to test. This will be the node which invokes **rc.spds**. The test assumes the 'tmp' LIBNAME definition included in the sample **libnames.parm** file was not changed during installation.

Watch the SAS log for error messages. You may see failures to properly locate one or more required SPD Server components. Examples of common error messages are displayed below. If you find one of the following errors, check your -path option to confirm that the directory where you loaded the SAS System components is properly set:

```
ERROR:    Protocol version mismatch. Proxy version
          is 4.4 while engine version is 3.x.

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

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

If you see the following (or similar) error describing failures to access messages, check your -sasmsg option and confirm that the directory where you loaded the SPD Server components is properly set:

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

If the attempted connection to the server hangs for several minutes, check the -path option to make sure the directory where you loaded the SAS System components is properly set. The spds44 client component directory needs to be at the beginning of the path option.

3. Once the LIBNAMES are assigned, you can verify further by running the sample SAS program, *InstallDir***/samples/verify.sas.** Just submit the SAS command to execute the test stream:

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

This SAS stream exercises many of the features of the SPD Server LIBNAME engine and proxy to provide further confidence in your installation configuration. The test performs a sequence of DATA and PROC steps using a generated data set and self-checks the results expected from various DATA step queries. If any query fails to produce the expected result, the SAS job is aborted. The job **verify.sas** requires that the SAS librefs TEST and TESTRL are assigned as shown in Step 2 above.

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

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

where ***serverNode*** and ***port*** are the same as in the previous LIBNAME assignment step.

SPD Server Command Reference

SPD Server operation revolves around the executables described in the packing list section of this document. The executables are in the /**bin64** directories. The /**bin32** directories are intentionally left **empty**. Each executable supports a set of command line options which override default features, or supplies site-dependent configuration information. The command options for each executable are given below:

Name Server Commands

The SPD Server LIBNAME engine connects to the name server. The name server resolves LIBNAME domain names into physical file system paths for LIBNAMEs. The name server also resolves host node and end-point (TCP port) addresses for each LIBNAME. Each SPD Server Host process (**spdsserv** process) registers LIBNAME domain information from its configuration file with its appointed name server process (**spdsnsrv** process). Multiple SPD Server Hosts may 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 connecting to the name server.

The SPD Server name server is invoked using the following command line syntax:

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

The spdsnsrv command supports the following options:

-listenport port#

Used to specify 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, the system chooses a dynamic port number for the name server to use.

-licensefile lic-file

License file keys are generated by SAS Institute and supplied to you. With this production release of SPD Server, you receive an SPD Server license key for each machine where you license the SPD server and that key must be entered into this license file by the SPD Server administrator. The SPD Server will not run on a given machine without first entering a valid license key in this file. License keys are plain text strings that product, site, and machine information along with the password required to allow you to use 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.

The SPD Server Host is invoked using the following command line syntax:

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

Part of the function of the SPD Server Host process is to startup SPD Server logging processes. The spdslog process performs message logging functions. The spdsaud process performs audit logging functions. Message and audit logging functions are controlled using spdsserv command line options.

Both message and audit logging facilities include automatic logfile naming and periodic logfile cycling support. **Spdsserv** command line options control automatic logfile naming and cycling properties. Server availability improves because you can periodically switch to a new server log and/or audit log file without halting and restarting the SPD Server environment. The default **rc.spds** script shipped in the **samples/** directory provides examples of the command line options.

Audit log records are kept for all resource accesses 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 which processes the audit log and produces a report is provided. Check **samples/audit.sas** for information on processing the audit log and generating the report. To enable the audit trail log, use the **spdsserv** command with the -auditfile option.

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

The spdsserv command supports the following options:

-parmfile file-spec

Allows you to specify 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 spdsserv.parm in the process's current working directory. Option declarations in this file are of the form:

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

The recognized -parmfile option names are listed here but are described in the online documentation. Most sites need not modify the defaults in ***InstallDir*/site/spdsserv.parm**. For information on server parameters, see the Help section on, "Setting Up SPD Server Parameter Files," in the online SPD Server Administrator's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html.

-acldir pwd-dir-path

Specifies the directory path to the SPD Server Host SPD Server password file. This option may be omitted if the PASSPATH option is declared in the SPD Server Host's -parmfile. A valid SPD Server password file is required even when running with the -noacl option. You must use the SPD Server **psmgr** utility to create the password file and 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 does not need to be the same node that is hosting the SPD Server Host processes. This option is required.

-nameserverport port#

Allows you to specify an explicit TCP port number for the SPD Server Host to use to connect to its name server. If no port is specified, the name server queries the system for a registered port address 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 may span multiple lines and must begin with the LIBNAME=name keyword. Each LIBNAME definition must be terminated with a ";" character. A brief summary of the valid LIBNAME definition keywords follows:

LIBNAME=domain-name

Declares the logical LIBNAME domain name for this definition. Must be first keyword in each definition.

PATHNAME=lib-dir-path

Primary directory path specification for the LIBNAME domain. This corresponds to the SAS LIBNAME statement physical path specification.

OWNER=SPDS-owner

Specifies the SPD Server user ID of the owner of the LIBNAME domain itself. This allows ACL controls to be placed on the SPD Server users that are allowed to create members in the LIBNAME domain. This keyword is optional. If omitted, no specific user ID ownership is asserted for the LIBNAME domain.

ROPTIONS=SPDS-opts

Specifies restrictions to SPDS-specific LIBNAME statement options that are carried along with the LIBNAME assignment when you refer to the logical LIBNAME domain name. LIBNAME options specified by ROPTIONS= always take precedence over those specified by OPTIONS=, or on the LIBNAME statement itself. This enables the SPDS administrator to ensure that certain LIBNAME options cannot be overridden by the SAS user. You specify this using the same option names and syntax as on the SAS LIBNAME statement. The entire string must be included in double quotes as shown in the following example:

```
ROPTIONS="DATAPATH=(' /data/trials' ' /data2/trials')
INDEXPATH=('/index1/trials' '/index2/trials')"
```

OPTIONS=SPDS-opts

Similar to ROPTIONS, but supplements rather than overrides LIBNAME statement options.

-logfile *fileSpec*

Selects automatic server log file creation by the logger process. *fileSpec* specifies a partial path/file name spec that is used to generate the complete log file path. For example if you specified *fileSpec* as **/logs/spds**, the generated name would appear as follows:

```
/logs/spds_mmddyyyy_hh:mm:ss.spdslog
```

where *mmddyyyy* and *hh:mm:ss* are taken from the system time when the log file is created.

-logtime *hh:mm*

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

-auditfile *fileSpec*

Enables audit logging for the server and automatic audit log file creation by the audit process. *fileSpec* specifies a partial path/file name spec that is used to generate the complete audit file path. For example if you specified *fileSpec* as /audit/spds, the generated name would appear as follows:

```
/audit/spds_mmddyyyy_yyyy.spdsaudit
```

where *mmddyyyy* are taken from the system date when the log file is created.

-audittime *hh:mm*

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

SNET Server Commands

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

The SNET server is invoked with the following command line syntax:

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

The spdssnet command supports the following options:

-listenport *listen_port*

Specifies the listen port number spdssnet will use to accept connections from ODBC, JDBC, or htmSQL clients. If not specified, spdssnet will use the named service *spdssnet* from the /etc/services file to determine its listen port.

Password Utility Reference

The SPD Server **psmgr** utility allows the SPD Server administrator to create and maintain the data set containing the authorized SPD Server user IDs. This is the SPD Server analogue of the normal UNIX user ID facility. If you choose to run SPD Server ACL support, you will need to create and populate the SPD Server password file using this utility before starting the SPD Server environment. Refer to the Help section on, "SPD Server Password Manager Utility," in the online SPD Server Administrator's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html for more information.

Performance and Profiling Server Reference

The SPD Performance and Monitoring Server is available to minitor and log the activity of the SPD Server processes. The SAS Management Console SPD Server Manager utility can connect to the Performance and Monitoring Server to provide real-time feedback about the SPD Server process activity.

For more detailed information on SPD Server Profiling, see the chapter on, " SMC SPD Server Manager" in the online SPD Server 4.4 Administrator's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html.

SPD Server 4.4 and the SAS Management Console Utility

The SAS Management Console (SMC) 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 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 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, application modules designed to create metadata for a specific type of resource.

For example, administrators can use the SAS Management Console to configure SPD Server user and group passwords and

ACLs instead of using the traditional SPD Server **psmgr** utility and PROC SPDO statements.

The SAS Management Console plug-in file for SPD Server is located at

SASROOT/**spds44/spdssmc/sas.smc.SpdsMgr.jar**

Note: *SASROOT* represents the path to the base directory of the SAS software installation on your client machine. *Spds44/* represents the installed SPD Server software directory. The name of the installed SPD Server software directory varies according to the specific version and release of your SPD Server software. For example, the path to your SPD Server Java plug-in file might begin with *SASROOT/spds44*, *SASROOT/spds44tsm1*, or *SASROOT/spds44tsm2,* depending on if you have the original SPD Server 4.4 software, or the first or second maintenance release of the SPD Server 4.4 software.

Copy the SPD Server Java plug-in file to the SAS SMC **plugins** directory:

SASROOT/**SASManagementConsole/9.1/plugins**

Unzip the SPD Server Java documentation file to the SAS SMC documentation directory. If you are using Winzip, select the check box option to include the folder names when you unzip the files. The spdsmgrdoc.zip file is located at

SASROOT/**spds44/spdssmc/spdsmgrdoc.zip**

and should be unzipped to:

SASROOT/**SASManagementConsole/9.1/doc**

SPD Server and SAS Data Integration Studio

You can integrate the processing power of SPD Server with other SAS software tools, such as SAS Data Integration Studio.

SAS Data Integration Studio is software that enables data warehouse specialists to create and manage metadata objects that define sources, targets, and the sequence of steps for the extraction, transformation, and loading of data into data marts or warehouses.

To integrate SPD Server functionality into the SAS Data Integration Studio graphical user interface, copy the SPD Server Java plug-in file into the SAS Data Integration Studio **plugins** subdirectory.

The SPD Server Java plug-in file is located at:

SASROOT/ **spds44/spdssmc/sas.smc.SpdsMgr.jar**

Note: *SASROOT* represents the path to the base directory of the SAS software installation on your client machine. *Spds44/* represents the installed SPD Server software directory. The name of the installed SPD Server software directory varies according to the specific version and release of your SPD Server software. For example, the path to your SPD Server Java plug-in file might begin with *SASROOT/spds44*, *SASROOT/spds44tsm1*, or *SASROOT/spds44tsm2,* depending on if you have the original SPD Server 4.4 software, or the first or second maintenance release of the SPD Server 4.4 software.

Copy the SPD Server Java plug-in file to the SAS Data Integration Studio **plugins** directory:

SASROOT/ **SASETLStudio/9.1/plugins/sas.smc.SpdsMgr.jar**

SPD Server Lightweight Directory Access Protocol (LDAP) Authentication

In SPD Server for Solaris and AIX, clients can be authenticated by either the PSMGR password facility, or by a Lightweight Directory Access Protocol (LDAP) Server such as Microsoft Active Directory, Sun Microsystems Directory Server, or OpenLDAP from www.openldap.org. LDAP authentication integrates with the SPD Server password facility and offers a centralized approach to UserID and password management. SPD Server clients that use LDAP authentication should have user

accounts managed by the authenticating LDAP Server. In addition the UserID and password information must be stored on an LDAP server that the SPD Server can access. The UserID must also be entered into the SPD Server's password database through PSMGR or the SPD Server SAS Management Console Utility to record all other SPD Server User information.

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. AAfter the client is verified, SPD Server uses the client's password database record for all other SPD Server operations.

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

Parameter Description	Values	Default Setting
(NO)LDAP: directs user authentication to LDAP Server	LDAP/NOLDAP	NOLDAP
LDAPSERVER: LDAP Server IP address	a valid IP address	LOCAL_HOST
LDAPPORT: LDAP Server port number	0-65536	LDAP_PORT
LDAPBINDMETH: LDAP bind method	"LDAP_AUTH_SIMPLE" "LDAP_AUTH_SASL"	Null
LDAPBINDDN: LDAP bind distinguished name	char string	Null

The LDAP parameter turns on LDAP Authentication. If the LDAP parameter is present during start up, the SPD Server creates a context for LDAP authentication.

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

The LDAPPORT parameter specifies the TCP/IP port that is used to communicate with the LDAP server. This is usually the default "LOCAL_HOST" or port 389.

The LDAPBINDMETH parameter dictates the way SPD Server clients are authenticated by the LDAP Server. When present in the SPD Server parameter file, LDAPBINDMETH is a character string whose value is either LDAP_AUTH_SIMPLE or LDAP_AUTH_SASL.

The default authentication method, LDAP_AUTH_SIMPLE, sends the SPD Server Client's user name and password to the LDAP server in clear text. LDAP_AUTH_SIMPLE should not be used in a secure environment.

When LDAPBINDMETH="LDAP_AUTH_SASL", the LDAP server will authenticate SPD Server clients via the Simple Authentication and Security Layer or SASL method. SASL is the preferred authentication method for secure environments. When authenticating via SASL, the SPD Server specifies that the SASL DIGEST-MD5 mechanism is used. SASL/DIGEST-MD5 is the most commonly accepted means of LDAP authentication and it is a requirement for all V3 LDAP Server Products.

The LDAPBINDDN parameter is the "Distinguished Name" (DN), or the location in the LDAP Server's database where the client's information is stored. The form of this string is

```
"ID= , rdn1=RDN1, rdn2=RDN2, ...".
```

"ID" is the identifier for the Relative Distinguished Name of a UserID that exists in the LDAP Sever database. The default value of the DN is

```
"uid= , dc=DOM1, dc=DOM2, dc=DOM3".
```

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's UserID becomes the value for "uid". The result becomes the default user location for LDAP database members.

For example, let the LDAP host machine be **sunhost.unx.sun.com** and the UserID be "sunjws." The resulting default Distinguished Name would be

```
"uid=sunjws, dc=unx, dc=sun, dc=com".
```

The Distinguished Name is used to locate the user "sunjws," then compares the sunjws user password to the one that is stored in the LDAP database. If there is a specific location for SPD Server users in your LDAP database, be sure to specify it using LDAPBINDDN utility.

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

Note: Entering the LDAP_HOST value for the 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 the SPD Server system. The following are some 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 their access to SPD Server data resources via the SPD Server password facility without giving them specific login accounts on the system. This adds a measure of security and control whereby SPD Server users are permitted physical access to the SPD Server machine.

You should add the *InstallDirl***bin64** (64 bit) directory to your PATH via your shell's login script. Ksh users should modify .profile or .kshrc files. Csh users should modify .login or .cshrc files, depending on where they currently set the PATH environment variable. This makes invoking the various SPD Server utility programs much easier.

SAS recommends that you run your SPD Server environment under the same UNIX user ID that was used to install the SPD Server software 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 created with the SPD Server environment 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 runtime environment is configured, SPD Server processes always run under some UNIX user ID. That UNIX user ID owns of all the files that the SPD Server process creates. Of course, the UNIX user ID is also governed by UNIX file access permissions. Be mindful of this when starting SPD Server processes and running SPD Server administrator utilities! Otherwise, it is possible to create files with ownership and permissions which deny required access to the SPD Server processes. Performing all SPD Server installation and administration tasks from the same UNIX user ID makes subsequent SPD Server use much easier.

Following are some options for establishing the proper UNIX user ID for your SPD Server processes:

Establish a dedicated UNIX account for the SPD Server administrator and always execute the **rc.spds** script from that account.

The **rc.spds** script that starts the SPD Server processes should use the setuid bit. No 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 under the proper UNIX user ID.

At system startup, use the UNIX su command to establish the proper 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 root when you execute the su command.

SPD Server User IDs

The SPD Server system uses its own layer of access controls which 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 finer access control to SPD Server's data resources than a native UNIX user ID. Many sites will not want to give UNIX accounts to SPD Server system users, but still want protection and ownership of the data resources created in the SPD Server environment. In this case, SPD Server user IDs provide the extra layer of access control.

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

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 will not function properly. To disable the use of SPD Server user IDs at your site, supply the `-noacl` option when you startup the SPD Server host process.

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 its **stdin** so you can pipe commands to it from another command, script, or from an input file.

Lightweight Directory Access Protocol (LDAP) Password Authentication

LDAP Authentication will cause SPD Server to authenticate a SPD Server user password via LDAP rather than by the password in the PSMGR database. LDAP authentication allows an SPD Server user to have the same UserID and password as their UNIX logon, as long as the UNIX logon meets the SPD Server character restrictions for UserIDs and passwords.

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

For more detailed information on SPD Server LDAP Authentication, see the chapter on, "SPD Server Password Manager," in the online SPD Server 4.4 Administrator's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html.

Troubleshooting

Troubleshooting networked applications is often difficult. Key ingredients to SPD Server troubleshooting are the name server and SPD Server Host process log files. With those two log files, you can reconstruct some of the 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 Startup Failed

Check the name server log file. The log should contain good clues about the problem. Some common things to watch for include:

1. Invalid *-licensefile* file specification.
2. *-licensefile* specifies a file with invalid contents.
3. Name Server port is in use by another process. Check for another name server process running already on the same node.

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

SPD Server Host Startup Failed

Check the SPD Server Host log file for clues. Some common things to watch for include:

1. *-nameserver* node name is incorrect.
2. *-nameserverport* specifies wrong port number if SPD Server Name Server is running with a non-standard port assignment.
3. *-parmfile* file specification is invalid or specified file does not exist.
4. *-libnamefile* file specification is invalid or specified file does not exist.
5. Contents of specified -libnamefile does not conform to expected syntax. Check SPD Server Host's log for messages about which entries are invalid.
6. *-acldir* option was omitted from the command line.
7. *-acldir* option specifies an invalid directory path for locating the SPD Server password file or the specified directory path does not contain a valid SPD Server password file.

SAS LIBNAME Assignment Failed

On the SAS side, first attempts to diagnose a failure depend on the error messages from the SPD Server LIBNAME engine via the SAS LOG output. In most circumstances you should be able to diagnose the reason for the failure from this message. Some common problems include:

1. Invalid specification of the LIBNAME engine selector in the LIBNAME statement. The SPD Server engine name is sasspds and is misspelled in the following LIBNAME statement.

```
1? 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.
```

2. Invalid specification of the logical LIBNAME domain name in the LIBNAME statement. The domain name 'test' is not defined in the SPDS name server sunspot.spdsname.

```
2? 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.
```

3. No name server is running on the specified node name or no name server is available at the specified port address. In this case, no name server is running on the specified node `stelling`. This same message would be generated if the port address is incorrect.

```
3? 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.
```

4. An invalid or unknown node name is specified in the LIBNAME statement. In this case, node `xxx` is not accessible in the network.

```
4? 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.
```

5. Invalid SPD Server user password specified in the LIBNAME statement. In this case, the SPD Server user ID is derived from the UNIX user ID running the SAS session. The SPD Server password file has an entry for this SPD Server user ID, but the password is not xxx.

```
8? 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.
```

6. Invalid SPD Server user ID specified in the LIBNAME statement. In this case, the SPD Server user ID xxx does not exist in the SPD Server Host's password file. The resulting failure message is the same as for the invalid password case.

```
10? 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.
```

Using SETINIT to Extend SPD Server Software

When you receive SPD Server software, the licensing information is pre-initialized. When you contract to renew the license, a paper SETINIT provides you with the licensing information.

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

Create the SETINIT.LIC File

1. Create or modify a file named **spds.lic**. This is the file that the SPD Server name server points to with the license file parameter.

   The following is an example of the spds.lic file:

```
PRODUCT     = SPD Server
VERSION     = 4.4
OSNAME      = AIX
SITENAME    = TEST SPDS SITE
SITENUM     = 00999999
NUSERS      = 0
GRACE       = 062
WARN        = 031
SERIALCHECK = 0
EXPIRE      = 03OCT2006
MODEL       =
CPUSERIAL   =
PASSWORD    = 12345678.1
```

2. After you modify the spds.lic file, you can start your SPD Server environment.

SAS Scalable Performance Data Server 4.4
Windows Installation Guide

Before You Install: Precautions and Required Permissions

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

- SPD Server is compatible with Microsoft's WIN32 API. SPD Server 4.4 runs on computers using Windows NT, Windows 2000, and Windows XP operating environments.

- The SPD Server distribution comes packaged with SAS client modules that are compatible with SAS 9.1.3. SPD Server 4.4 is not compatible with earlier versions of SAS. You must make configuration changes to existing SAS 9.1.3 installations before the SAS clients can access the SPD Server 4.4 environment. The section Installing and Configuring SPD Server Clients contains step-by-step information on the changes that must be made to existing SAS clients.

- You will need write access to your server machine's

  ```
  \etc\services
  ```

  file if you want to allow SPD Server clients to connect to the SPD Server host using named services instead of specifying port numbers. Named services require you to define one or more registered ports which will use the services file appropriate to your machine.

- If your SPD Server clients will access the SPD Server host using named services instead of specifying port numbers, you will also need write access to the services files on the client computers. For Windows, the path would be

  ```
  C:\winnt\system32\drivers\etc\services
  ```

- You should insert the WORKPATH= server option in your **spdsserv.parm** file, which resides on the *InstallDir*\ *DOWN* directory of your SPD Server host computer. You should use the WORKPATH= option to configure your server to use a high-performance file system. Ideally this will be a RAID-structured volume with sufficient disk space to accommodate the transient storage needs for the SPD Server.

Installing SPD Server from CD

After you have reviewed the precautions and obtained the required permissions on the computer(s) where you plan to install SPD Server 4.4, insert the SPD Server 4.4 distribution CD in your CD-ROM drive. Your Windows computer should recognize the CD and automatically begin the install process. If the Autorun feature on your CD drive is disabled, you can use a file utility to locate and run the SETUP.EXE file located in the root of the SPD Server distribution CD.

The install program will copy and create necessary files to your SPD Server host machine, as well as initialize the SPD Server user password database.

The **spds.lic** file is contained on the CD. This file must be copied to the **\lic** directory located under the SPD Server root directory where the software has been installed.

When you complete the CD-based portion of your SPD Server host installation, you must check some of the default installation settings before proceeding to verify your SPD Server installation.

Validating Default Port and Library Assignments

After you install SPD Server from the distribution CD to your Windows server, you are almost ready to verify your installation. Before you verify your installation, you must validate your Name Server port assignment and validate the path for the temporary LIBNAME domain called TMP.

SPD Server documentation uses *InstallDir*\ as a placeholder which represents the path specification to the directory that you installed the SPD Server host to. Any SPD Server configuration files that require editing are located in the *InstallDir***site** directory. Check the following items to make sure your Windows server environment matches requirements from the installation procedure.

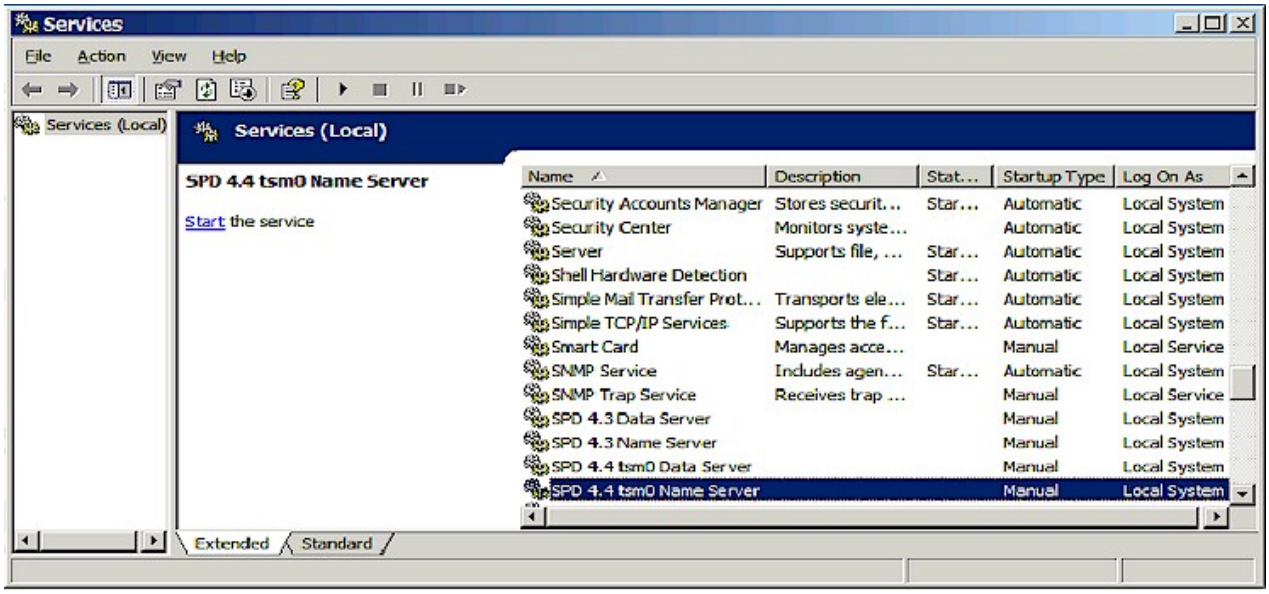
1. The default SPD Server installation configures port 5200 as the Name Server port. If you want to change the Name Server port, edit the **spdsserv.bat** and **spdsnsrv.bat** files located in *InstallDir***site**. Use the ***-listenport*** command option in **spdsnsrv.bat** to specify the port number that you want to use**.** Use the ***-nameserverport*** option in **spdsserv.bat** to specify the port number that you want to use**.**

2. During installation, SPD Server creates a library parameter file called **libnames.parm.** 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 will either need to create the directory or specify an existing directory path to replace C:\TEMP. If you want to use a different path for your TMP domain, you must modify the SPD Server **libnames.parm** configuration file in *InstallDir***site** to specify the new TMP domain path.

3. After you verify your Name Server port and TMP domain path, you are ready to start the SPD Server environment. You start the SPD Server environment using Windows Services. To open the Windows Services window, select

   `Start --> Settings --> Control Panel --> Administrative Tools --> Services`

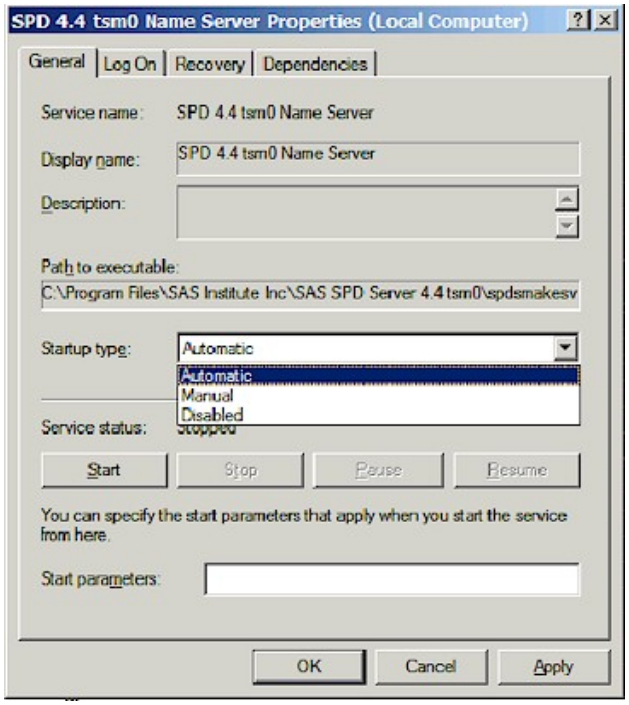
   The main panel of the Services window contains a list of Windows services, sorted alphabetically by default. Scroll down to find entries for the SPD 4.4 Name Server and the SPD 4.4 Data Server.



   The first time you configure SPD Server in the Services window, the Status will be blank and the Startup Type is set to Manual.

4. Most users will want to configure SPD Server to automatically start and stop the SPD Name and Data servers. The automatic setting loads the SPD Name and Data servers without prompting on boot-up, and similarly stops the services without intervention on Windows close.

   To change the Startup Type setting for the SPD Name Server, select the service in the list (as shown above). Right-click the highlighted entry and select Properties from the pop-up menu. This opens the Properties Window for the SPD Name Server.



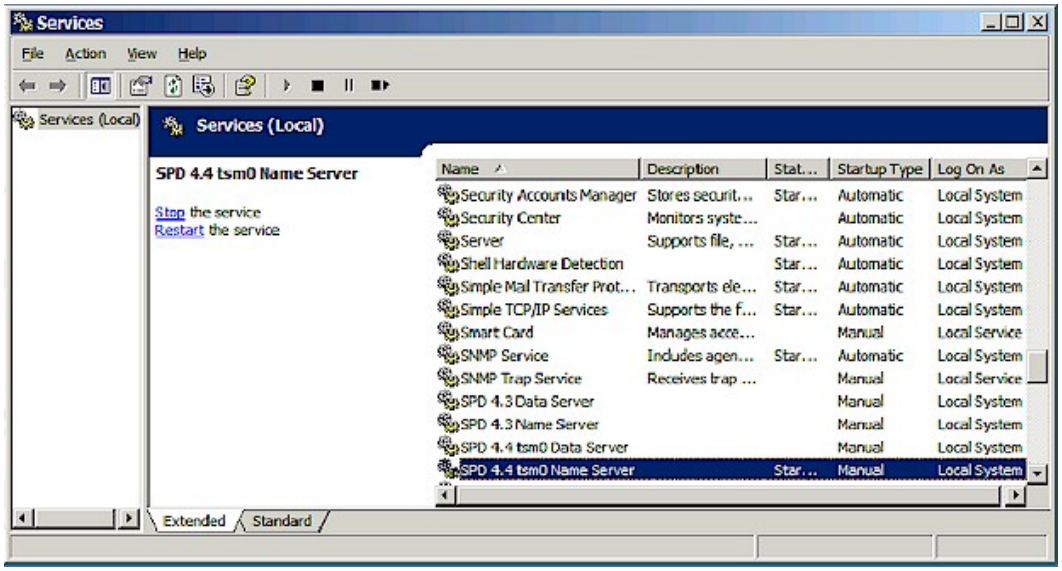
Select Automatic from the Startup Type list. This configures the SPD 4.4 Name Server service to automatically start and end with Windows.

Select OK to close the window and apply the changes.

5. Repeat the process outlined in the previous step to change the Startup Type setting for the SPD 4.4 Data Server from Manual to Automatic. At this point, SPD Name Server and Data Server services should now be configured for automatic start and stop.

   The first time you set the services to Automatic, you will need to manually start them. Afterwards, the SPD Name and Data Servers will automatically start at boot-up. Steps 6 and 7 show how to manually start the two services.

6. The SPD Name Server should always be manually started before the SPD Data Server. Highlight the SPD Name or Data Server row in the Services window, then click the *Start the Service* hypertext in the left side of the panel. An alternative method is to right-click the highlighted row and choose Start from the pop-up menu. Either approach will work.



After you start the SPD Name Server, use the same method to start the SPD Data Server. The main panel in the Services window should indicate that both services are configured for Automatic Startup Type and are both started.

7. If you need to stop the SPD Server Name or Data Server services after they are started, you can use the hypertext commands in the Services window. Once the services are started, the Extended tab of the main Services window panel displays a link to stop the service. Simply click on the appropriate service to highlight it, then click the corresponding command in the Extended tab of the Services window.

You can use the hypertext controls in the Extended tab of the Services window to start, stop, and restart the SPD Server Name and Data Server services. The hypertext controls appear regardless of whether the services are in Manual or Automatic configuration.

Configuring SPD Server Software on Your Windows Host

After you validate your port and library assignments and start the Name and Data servers, you can begin configuring the LIBNAME domains and user password files.

1. Configure the **libnames.parm** file with all of the LIBNAME domains that you will use to store SAS tables and catalogs. Declaring all your LIBNAME domains requires thought and planning in order to best exploit the parallel processing capabilities of the SPD Server. Refer to the Command Reference and the **libsamp.parm** file found in *InstallDir*\ for more information on the format of the **libnames.parm** file. Additional Help can be found on, "Configuring Disk Storage for SPD Server," and, "Using the SPD Server Name Server to Manage Resources," in the online SPD Server 4.4 Administrator's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html. Any changes that you make to the **libnames.parm** file while the SPD Server is running will not take effect until SPD Server is restarted.

2. Add your SPD Server user IDs to the SPD Server password file. You should run the SPD Server Account Manager utility to perform this function. You can add or modify the SPD Server password file at any time, even while SPD Server is running.

3. Use Account Manager to add users and groups. Refer to the section in this guide on, "Connecting to SPD Server via ODBC, JDBC, and htmSQL," in the online SPD Server 4.4 Administrator's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html.

   After you set up and configure your SPD Server host environment, take time to examine the files in your *InstallDir*\ directory. It contains various SAS programs to help you understand how to use various SPD Server features. The sample files are:

   - **doc_examples.sas** contains sample SAS code used in the SPD Server User's Guide documentation.
   - **verify.sas** is an installation verification SAS job to be run following SPD Server installation.
   - **spdsinst.sas** demonstrates several SPD Server features.
   - **passthru.sas** demonstrates the use of SQL pass-through. It demonstrates simple single level pass-through as well as secondary LIBREF and connection scenarios.
   - **tempwork.sas** demonstrates the use of temporary LIBNAME domain support. Files created in a temporary LIBNAME domain are automatically deleted when the SAS session ends.
   - **paraload.sas** demonstrates a technique for performing parallel loads from an existing table into an SPD Server table. The technique exploits the parallel load capability in the SPD Server LIBNAME proxy. The LIBNAME proxy uses the same technology as the SQL LOAD TABLE statement.
   - **aclcolrw.sas** demonstrates the use of ACL row/column security features.
   - **symbsub.sas** demonstrates how the use of symbolic substitution in pass-through SQL can provide row-level security to sensitive tables.
   - **fmtgrpby.sas** demonstrates the use of formatted parallel groupby in pass-through SQL.
   - **rcperf** is a Bourne shell script to start up a "standard" SPD Performance and Profiling Server.
   - **dynamic_cluster*.sas** shows how to use dynamic clusters with a minmax variable list.
   - **minmax*.sas** shows how to use a minmax variable list on an SPD Server table.
   - **paralleljoin*.sas** shows the use of the SQL Parallel Join performance enhancement.
   - **starjoin*.sas** shows the use of the SQL star join performance enhancement.
   - **index_scan*.sas** shows the use of the SQL index scan performance enhancement.
   - **materialize_view*.sas** shows the use of the materialized view performance enhancement.

Upgrading SPD Server 3.x with SPD Server 4.4

If you have SPD Server 3.x installed on your Windows server, consider copying the **libnames.parm** file from your SPD Server 3.x

*InstallDir***site** directory into the SPD Server 4.4 *InstallDir***site** directory. Your SPD Server 3.x **libnames.parm** file overwrites the temporary file that SPD Server created when it verified the installation. Your existing SPD Server 3.x **libnames.parm** file gives you access to all the LIBNAME domains that you defined in your SPD Server 3.x server environment. Consult the Help section on, "SPD Server 3.x and SPD Server 4.4 Compatibility," in the online SPD Server 4.4 Administrator's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html, for information on known issues that exist between SPD Server 3.x and SPD Server 4.4.

You should now be in a position to have both SPD Server 3.x and SPD Server 4.4 coexist with each other. You are configured to permit dual access to the appropriate LIBNAME domains for your installation.

Running SPD Server 3.x Concurrently with SPD Server 4.4

You may run SPD Server as a completely independent environment from SPD Server Version 3.x. This means you will have two SPD Server Name Servers. Be careful with your port assignments if you run two separate SPD Server releases concurrently; SAS clients must know whether they need to connect to the SPD Server 3.x Name Server or to the SPD Server Name Server to get their data.

Converting SPD Server 3.x Tables to SPD Server 4.4 Tables

If you are converting an SPD Server 3.x environment to SPD Server 4.4 and do not plan on retaining the 3.x server, you may use the "SPDSCONV utility" in the online SPD Server 4.4 Administrator's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html, to convert all existing SPD Server 3.x tables to SPD Server 4.4 tables. When you run the SPDSCONV utility, the **dulibv3.dll** file must be located in the same directory as the **spdsconv.exe** file.

Copying SPD Server 3.x Data to SPD Server 4.4

With the latest version of SAS, it is not necessary to use the SPDSCONV utility to migrate SPD Server 3.x data for use with SPD Server 4.4.

If you are using SAS 9.1.3 and SPD Server 4.4, you can access SPD Server 3.x tables using the sasspds3 client. (The sasspds3 client is shipped with SPD Server in the same location as the SPD Server client.)

To migrate the data for use in the SPD Server 4.4 environment, copy the SPD Server 3.x tables to a location on the SPD Server 4.4 server. Before you copy the tables to the new server, you will need to issue LIBNAME statements for both new and old sasspds engines.

The code example below issues the LIBNAME statements, one for the SPD Server 4.4 engine and server and one for the SPD Server 3.x engine and server. After issuing the two libname statements, use standard SAS syntax to copy the tables to the location that you specify on your SPD 4.4 Server..

```
Libname newspds sasspds 'newspds'
   server=host.new-port
   user='anonymous' ;

Libname oldspds sasspds3 'oldspds'
   server=host.old-port
   user='anonymous' ;

/* SAS statements to copy data */
```

Installing and Configuring SPD Server Clients

Once you configure and validate your SPD Server host environment, you must install and configure some software on the client platforms that will connect to SPD Server. This may include the computer that is actually running the SPD Server, so some of these steps outlined here may have already been covered during installation of the SPD Server host software.

SPD Server media contains SAS client software modules for SAS 9.1.3 installations on Solaris by Sun, AIX by IBM, and HP/UX by Hewlett-Packard.

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

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

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

The **spdsname** service defines the port number for named services (if you are using the **spdsnsrv** name server process). Make sure that the port number you enter matches the port number that was used during the SPD Server host installation. If you are already running SAS with an earlier version of SPD Server, this service name is probably already defined. You can either define another service name for the SAS client to use (for example, **sp44name**) or you can directly include the SPD Server port number in your SAS statements.

2. SPD Server is designed to be accessed from multiple platforms. You will find subdirectories for the SAS components in the root directory where your SPD Server was installed..

Platform	Subdirectory
Solaris2	SOL2
AIX/RS-6000	AIX
Windows/Intel	WIN
Solaris x64	SOLX64
HP/UX Itanium	HPIA64
HP/UX	HPUX

Using *!SASROOT* as a placeholder for the full path specification to the root directory of your SAS installation, create the following subdirectory.

*!SASROOT***spds44.**

Enter the directory that corresponds to your operating system. (If your SPD Server clients are using Windows, be sure to navigate to the WIN\core directory). All of the platform-specific directories contain two subdirectories called **sasexe** and **sasmsg**. Copy all of the component files that you find the **sasexe** subdirectory and the **sasmsg** subdirectory into the *!SASROOT*\spds44 directory that you just created on your SPD Server client computer.

These component files in the **sasexe** and **sasmsg** subdirectories exist in both Windows and UNIX-specific forms. Windows client files will have .dll extensions.

- **sasspds** is LIBNAME engine required to access SPD Server from SAS 9.1.3.
- **sasspdo** is the SPD Server operator procedure required to access SPD Server 4.4 from SAS.
- **spds.msg** is the SAS message file for the SPD Server LIBNAME engine and SPDO operator procedure.

Note: For SAS 9.1.3 Service Pack 3 and earlier releases, you must **rename** the **sassqlu_ for_ sas913_sp3_ and_ earlier** modules from the SPD Server client installation to **sassqlu**. If you do not rename this module for SAS 9.1.3 Service Pack 3 and earlier releases, problems will occur with SPD Server implicit pass-through SQL that utilizes three part names. You will get an SQL parse error from SPD Server that causes the implicit pass-through SQL to fail.

3. Now that SPD Server client software is on your SAS computer, SAS needs pointers to the SPD Server software in the SAS configuration file, **sasv9.cfg**. Choose the appropriate configuration instruction below, depending on whether your SAS client is on a UNIX or Windows operating environment.

If your SAS and SPD Server clients are on UNIX: Open your **sasv9.cfg** file with a text editor and insert both of the following lines. Make sure that the new SPD Server directory is searched before any other -path() and/or -msg() directories in your **config.sas** file. Otherwise, SAS will not pick up the upgraded SAS System appendages and message files.

```
-path !SASROOT\spds44
-ins_msg !SASROOT\spds44
```

If your SAS and SPD Server clients are on Windows: Use an editor to open the **sasv9.cfg** file located at

C:\Program Files\SAS\SAS 9.1\nls\en\sasv9.cfg

and you can paste in the following sections. Remember that you must substitute the full path to the root directory of the SAS installation on your client computer where the placeholder *!SASROOT* is used:

```
/* Setup the SAS System message directory definition */

-MSG ("!sasroot\spds44"
      "!sasroot\core\sasmsg"
```

```
        "!sasroot\dmine\sasmsg"
        "!sasroot\dquality\sasmsg"
        "!sasroot\hpf\sasmsg"
        "!sasroot\inttech\sasmsg"
        "!sasroot\irp\sasmsg"
        "!sasroot\mddbserv\sasmsg"
        "!sasroot\access\sasmsg"
        "!sasroot\af\sasmsg"
        "!sasroot\ets\sasmsg"
        "!sasroot\genetics\sasmsg"
        "!sasroot\gis\sasmsg"
        "!sasroot\graph\sasmsg"
        "!sasroot\iml\sasmsg"
        "!sasroot\insight\sasmsg"
        "!sasroot\intrnet\sasmsg"
        "!sasroot\lab\sasmsg"
        "!sasroot\or\sasmsg"
        "!sasroot\qc\sasmsg"
        "!sasroot\sview\sasmsg"
        "!sasroot\stat\sasmsg"
        "!sasroot\tmine\sasmsg"
        )

    /* Setup the SAS System load image search paths definition */

    -PATH ("!sasroot\spds44"
           "!sasroot\core\sasexe"
           "!sasroot\dmine\sasexe"
           "!sasroot\reporter\sasexe"
           "!sasroot\dquality\sasexe"
           "!sasroot\hpf\sasexe"
           "!sasroot\inttech\sasexe"
           "!sasroot\irp\sasexe"
           "!sasroot\mddbserv\sasexe"
           "!sasroot\access\sasexe"
           "!sasroot\af\sasexe"
           "!sasroot\connect\sasexe"
           "!sasroot\eis\sasexe"
           "!sasroot\ets\sasexe"
           "!sasroot\fsp\sasexe"
           "!sasroot\genetics\sasexe"
           "!sasroot\gis\sasexe"
           "!sasroot\graph\sasexe"
           "!sasroot\iml\sasexe"
           "!sasroot\insight\sasexe"
           "!sasroot\intrnet\sasexe"
           "!sasroot\lab\sasexe"
           "!sasroot\or\sasexe"
           "!sasroot\qc\sasexe"
           "!sasroot\securwin\sasexe"
           "!sasroot\share\sasexe"
           "!sasroot\sview\sasexe"
           "!sasroot\stat\sasexe"
           "!sasroot\toolkt\sasexe"
           "!sasroot\tmine\sasexe"
           )
```

The SPD Server host can be accessed via the SAS/ODBC driver, SAS JDBC Driver, and SAS/htmSQL. Each of these drivers can be downloaded from the support link of the SAS Web at

http://support.sas.com

See, "Using SPD Server with Other Clients" in the online SPD Server 4.4 User Documentation located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html for more information on connecting and configuring these applications.

ODBC client applications require installation of the **spds.dll** application extension. Install the ODBC client application extension as follows

- Install the SAS/ODBC Driver Version 9

- Copy

  ```
  InstallDir\WIN\core\sasexe\spds.dll
  ```

  into

  ```
  <drive letter>:\Program Files\sas\shared files\general
  ```

- Configure an ODBC Data Source for direct SPD Server access. (For more details about configuring and using these clients, see, "Using SPD Server with Other Clients" in the online SPD Server 4.4 User Documentation located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html.)

4. Verify that the SAS Management Console has been installed on Windows clients by checking for the presence of the SAS Management Console directory

   ```
   <drive letter>:\Program Files\sas\SASManagementConsole
   ```

5. Copy the SMC files from the SPD Server host to the Windows SPD Server clients. Remembering that *InstallDir* is a placeholder for your SPD Server host installation directory, copy the SPD Server host JAR file

   ```
   InstallDir/spdssmc/sas.smc.SpdsMgr.jar
   ```

   onto the Windows SPD Server clients at

   ```
   <drive letter>:\Program Files\sas\SASManagementConsole\9.1\plugins
   ```

Testing Your SPD Server Installation Using SAS

Testing your SPD Server installation is relatively simple. To verify, you only need to make two SAS LIBNAME assignments using the SPD Server LIBNAME engine. The examples in this section refer to the SASSPDS engine, the engine for SAS 9.1.3.

1. Start the SPD Server environment as described in step 3 of the section Verifying Default Port and Library Assignments.

2. On a properly configured client system invoke SAS and make the following LIBNAME assignments:

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

   ServerNode is the Name Server's network node name and ***port*** is either the numeric value used to start the Name Server or the named service you chose to use to access the SPD Server Name Server. Remember that name services allow you to connect to a name server using a character string instead of specifying a port number. Assuming you used the default numeric port assignment of 5200, your assignment would look like:

   ```
   libname test sasspds 'tmp'
   server=serverNode.5200 user='anonymous';
   ```

   Or if you were using **spdsname** to provide named services, your assignment would resemble this:

   ```
   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;
   ```

   Verifying these simple statements ensures that you have connectivity between the SAS client and the SPD Server host components. Successfully performing the LIBNAME assignments means that the network configuration is correct and that most of the SPD Server host configuration is correct. You will need to fill in ***serverNode*** with the network node name where the Name Server and Data Server

processes reside. This assumes you left the 'tmp' LIBNAME definition that was included in the sample **libnames.parm** file you received with your distribution.

Watch the SAS log for error messages which indicate failure to properly locate one of the required SPD Server components. The messages that you will most likely encounter are included below.

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

If you see this error, check the -path option is properly set to make sure you are accessing the directory where you loaded the SAS System client-side components.

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

If you see the following or similar errors regarding failures to access messages, check the -sasmsg option to make sure the directory where you loaded the SAS System message file for SPD Server components is properly set.

3. Once the SPD Server host LIBNAMES are assigned, you can further verify your installation by running the sample SAS program, *InstallDir***samples\verify.sas.** Submit the following SAS command to execute the test stream:

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

This SAS stream exercises many of the features of the SPD Server LIBNAME engine and proxy to provide further confidence that the installation has been made correctly. It performs a sequence of DATA and PROC steps against a generated data set and performs self checking on the results expected from various DATA step queries of the test data set. If any one of these queries fails to produce the expected result, the SAS job is aborted. The job **verify.sas** assumes the SAS LIBREFs TEST and TESTRL are assigned from Step 2 above.

4. You should also verify that SQL pass-through services are working in SPD Server by performing the following sequence of SAS commands:

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

where ***serverNode*** and ***port*** are the same as in the previous LIBNAME assignment step.

SPD Server Command Reference

SPD Server executables support of command line options that override default features or supply site-dependent configuration information for the program to use. The command options for the following programs are explained below .

Name Server Commands

The SPD Server LIBNAME engine connects to the name server. The name server resolves LIBNAME domain names into physical file system paths for LIBNAMEs. The name server also resolves host node and end-point (TCP port) addresses for each LIBNAME. Each SPD Server host process (**spdsserv** process) registers LIBNAME domain information from its configuration file with its appointed name server process (**spdsnsrv** process). Multiple SPD Server hosts may 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 connecting to the name server.

The SPD Server name server is invoked using the following command line syntax:

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

The spdsnsrv command supports the following options:

-listenport port#

Used to specify 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, the system chooses a dynamic port number for the name server to use.

-licensefile lic-file

License file keys are generated by SAS Institute and supplied to you. With this production release of SPD Server, you receive an SPD Server license key for each machine where you license the SPD server and that key must be entered into this license file by the SPD Server administrator. The SPD Server will not run on a given machine without first entering a valid license key in this file. License keys are plain text strings that product, site, and machine information along with the password required to allow you to use 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.

The SPD Server host is invoked using the following command line syntax:

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

Part of the function of the SPD Server host process is to startup SPD Server logging processes. The spdslog process performs message logging functions. The spdsaud process performs audit logging functions. Message and audit logging functions are controlled using spdsserv command line options.

Both message and audit logging facilities include automatic logfile naming and periodic logfile cycling support. **Spdsserv** command line options control automatic logfile naming and cycling properties. Server availability improves because you can periodically switch to a new server log and/or audit log file without halting and restarting the SPD Server environment.

Audit log records are kept for all resource accesses 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 which processes the audit log and produces a report is provided. Check **samples\audit.sas** for information on processing the audit log and generating the report. To enable the audit trail log, use the **spdsserv** command with the -auditfile option.

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

The spdsserv command supports the following options:

-parmfile file-spec

Allows you to specify 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 spdsserv.parm in the process's current working directory. Option declarations in this file are of the form:

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

The recognized -parmfile option names are listed here but are described in the online documentation. Most sites need not modify the defaults in *InstallDir***site\spdsserv.parm**. For information on server parameters, see the Help section on, "Setting Up SPD Server Host Parameter Files," in the online SPD Server 4.4 Administrator's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html.

-acldir pwd-dir-path

Specifies the directory path to the SPD Server host password file. This option may be omitted if the PASSPATH= option is declared in the SPD Server host's *-parmfile*. A valid SPD Server password file is required even when running with the *-noacl* option. You must use the SPD Account Manager utility to create the password file and

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 does not need to be the same node that is hosting the SPD Server host processes. This option is required.

-nameserverport port#

Allows you to specify an explicit TCP port number for the SPD Server host to use to connect to its name server. If no port is specified, the name server queries the system for a registered port address 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 may span multiple lines and must begin with the LIBNAME=name keyword. Each LIBNAME definition must be terminated with a ";" character. A brief summary of the valid LIBNAME definition keywords follows:

LIBNAME=domain-name

Declares the logical LIBNAME domain name for this definition. Must be first keyword in each definition.

PATHNAME=lib-dir-path

Primary directory path specification for the LIBNAME domain. This corresponds to the SAS LIBNAME statement physical path specification.

OWNER=SPDS-owner

Specifies the SPD Server user ID of the owner of the LIBNAME domain itself. This allows ACL controls to be placed on the SPD Server users that are allowed to create members in the LIBNAME domain. This keyword is optional. If omitted, no specific user ID ownership is asserted for the LIBNAME domain.

ROPTIONS=SPDS-opts

Specifies restrictions to SPDS-specific LIBNAME statement options that are carried along with the LIBNAME assignment when you refer to the logical LIBNAME domain name. LIBNAME options specified by ROPTIONS= always take precedence over those specified by OPTIONS=, or on the LIBNAME statement itself. This enables the SPDS administrator to ensure that certain LIBNAME options cannot be overridden by the SAS user. You specify this using the same option names and syntax as on the SAS LIBNAME statement. The entire string must be included in double quotes as shown in the following example:

```
ROPTIONS="DATAPATH=(' \data\trials' ' \data2\trials')
INDEXPATH=('\index1\trials' '\index2\trials')"
```

OPTIONS=SPDS-opts

Similar to ROPTIONS, but supplements rather than overrides LIBNAME statement options.

-logfile *fileSpec*

Selects automatic server log file creation by the logger process. *fileSpec* specifies a partial path/file name spec that is used to generate the complete log file path. For example if you specified *fileSpec* as **\DOWNlogs\spds**, the generated name would appear as follows:

```
\DOWNlogs\spds_
mmddyyyy_hh:mm:ss.spdslog
```

where *mmddyyyy* and *hh:mm:ss* are taken from the system time when the log file is created.

-logtime *hh:mm*

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

-auditfile *fileSpec*

Enables audit logging for the server and automatic audit log file creation by the audit process. *fileSpec* specifies a partial path/file name spec that is used to generate the complete audit file path. For example if you specified *fileSpec* as \audit\spds, the generated name would appear as follows:

```
\audit\spds_mmddyyyy_yyyy.spdsaudit
```

where *mmddyyyy* are taken from the system date when the log file is created.

-audittime *hh:mm*

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

SNET Server Commands

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

The SNET server is invoked with the following command line syntax:

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

The spdssnet command supports the following options:

-listenport *listen_port*

Specifies the listen port number spdssnet will use to accept connections from ODBC, JDBC, or htmSQL clients. If not specified, spdssnet will use the named service *spdssnet* from the \etc\services file to determine its listen port.

PSMGR Password Utility

The SPD Server **psmgr** password utility allows the SPD Server administrator to create and maintain the data set containing the authorized SPD Server user IDs. If you choose to run SPD Server ACL support, you will need to create and populate the SPD Server password file before starting the SPD Server environment. You can use the **psmgr** utility or the SAS Management Console utility to manage passwords. More information on the **psmgr** utility is available in the Help section on SPD Server Password Manager Utility in the online SPD Server 4.4 Administrator's Guide, located at http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html..

SPD Server and the SAS Management Console

The SAS Management Console (SMC) 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 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 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, application modules designed to create metadata for a specific type of resource.

For example, administrators can use the SAS Management Console to configure SPD Server user and group passwords and ACLs instead of using the traditional SPD Server **psmgr** utility and PROC SPDO statements.

The SAS Management Console plug-in file for SPD Server is located at

SASROOT/**spds44/spdssmc/sas.smc.SpdsMgr.jar**

Note: *SASROOT* represents the path to the base directory of the SAS software installation on your client machine. *Spds44/* represents the installed SPD Server software directory. The name of the installed SPD Server software directory varies according to the specific version and release of your SPD Server software. For example, the path to your SPD Server Java plug-in file might begin with *SASROOT/spds44*, *SASROOT/spds44tsm1*, or *SASROOT/spds44tsm2,* depending on if you have the original SPD Server 4.4 software, or the first or second maintenance release of the SPD Server 4.4 software.

Copy the SPD Server Java plug-in file to the SAS SMC **plugins** directory:

SASROOT/**SASManagementConsole/9.1/plugins**

Unzip the SPD Server Java documentation file to the SAS SMC documentation directory. If you are using Winzip, select the check box option to include the folder names when you unzip the files. The spdsmgrdoc.zip file is located at

SASROOT/**spds44/spdssmc/spdsmgrdoc.zip**

and should be unzipped to:

SASROOT/**SASManagementConsole/9.1/doc**

SPD Server and SAS Data Integration Studio

You can integrate the processing power of SPD Server with other SAS software tools, such as SAS Data Integration Studio.

SAS Data Integration Studio is software that enables data warehouse specialists to create and manage metadata objects that define sources, targets, and the sequence of steps for the extraction, transformation, and loading of data into data marts or warehouses.

To integrate SPD Server functionality into the SAS Data Integration Studio graphical user interface, copy the SPD Server Java plug-in file into the SAS Data Integration Studio **plugins** subdirectory.

The SPD Server Java plug-in file is located at:

SASROOT/ **spds44/plugins/sas.smc.SpdsMgr.jar**

Note: *SASROOT* represents the path to the base directory of the SAS software installation on your client machine. *Spds44/* represents the installed SPD Server software directory. The name of the installed SPD Server software directory varies according to the specific version and release of your SPD Server software. For example, the path to your SPD Server Java plug-in file might begin with *SASROOT/spds44*, *SASROOT/spds44tsm1*, or *SASROOT/spds44tsm2,* depending on if you have the original SPD Server 4.4 software, or the first or second maintenance release of the SPD Server 4.4 software.

Copy the SPD Server Java plug-in file to the SAS Data Integration Studio **plugins** directory:

SASROOT/ **SASETLStudio/9.1/plugins**

Lightweight Directory Access Protocol (LDAP) Authentication

In SPD Server 4.4, clients can be authenticated by either the PSMGR password facility, or by a Lightweight Directory Access Protocol (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 UserID and password management. SPD Server clients that use LDAP authentication should have accounts in the domain in which the LDAP and SPD Servers are running. The UserID and password information must be stored on an LDAP server that the SPD Server can access. The UserID must also be entered into the SPD Server's password database through PSMGR or the SPD Server 4.4 SAS Management Console Utility to record all other SPD User information.

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 receives the client's username and password. The LDAP server authenticates the client, then returns the result to the SPD Server. 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, the following parameters must be added to the SPD Server's spdsserv.parm file:

Parameter Description	Values	Default Setting
(NO)LDAP: directs user authentication to LDAP Server	LDAP/NOLDAP	NOLDAP
LDAPSERVER: LDAP Server IP address	a valid IP address	LOCAL_HOST
LDAPPORT: LDAP Server port number	0-65536	LDAP_PORT
LDAPBINDDN: LDAP bind distinguished name	char string	Null

The LDAP parameter turns on LDAP Authentication. If the LDAP parameter is present during start up, the SPD Server creates a context for LDAP authentication.

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

The LDAPPORT parameter specifies the TCP/IP port that is used to communicate with the LDAP server. This is usually the default "LOCAL_HOST" or port 389.

The LDAPBINDDN parameter is the "Distinguished Name" (DN), or the location in the LDAP Server's database where the client's information is stored. The form of this string is

```
"ID= , rdn1=RDN1, rdn2=RDN2, ...".
```

"ID" is the identifier for the Relative Distinguished Name of a UserID that exists in the LDAP Sever database. The default value of the DN is

```
"uid= , dc=DOM1, dc=DOM2, dc=DOM3".
```

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's UserID becomes the value for "uid". The result becomes the default user location for LDAP database members.

For example, let the LDAP host machine be **sunhost.unx.sun.com** and the UserID be "sunjws." The resulting default Distinguished Name would be

```
"uid=sunjws, dc=unx, dc=sun, dc=com".
```

The Distinguished Name is used to locate the user "sunjws," then compares the sunjws user password to the one that is stored in the LDAP database. If there is a specific location for SPD Server users in your LDAP database, be sure to specify it using LDAPBINDDN utility.

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

Note: Entering the LDAP_HOST value for the LDAPSERVER can cause SPD Server to fail during start up. It is recommended that SPD Server and LDAP Server use the same hosts. The user password is sent to the LDAP server in clear text. If someone is "sniffing" the network, user passwords could potentially be intercepted.

Notes for SPD Server Administrators

The SPD Server administrator has the role of performing many of the maintenance and configuration functions for the SPD Server system. The following are some guidelines and ideas for helping out in this capacity.

SPD Server User IDs

The SPD Server system uses its own layer of system access controls as a clean layer over the file system access permissions. SPD Server processes run in the context of a Windows user ID, and that user owns all of the file resources that are created from this SPD Server.

The SPD Server password file allows you to control access to the SPD Server's data resources at a finer level of granularity than the UNIX user ID. Many sites will not want to give Windows accounts to SPD Server system users, but still want protection and ownership of the data resources created in the SPD Server environment. SPD Server user IDs allow for this extra layer of access control.

The SPD Server administrator needs to be familiar with the **Account Manager** utility provided with the SPD Server system.

If you choose not to use SPD Server user IDs, you will still need the SPD Server password file for the Data Server process to function properly. To disable the use of SPD Server user IDs at your site, supply the `-noacl` option when you startup the Data Server process.

If you use SPD Server user IDs, you will need to add them to the SPD Server password file created during the installation phase. The **Account Manager** command reads its commands from its **stdin** so you can pipe commands to it from another command, script, or from an input file.

Troubleshooting

Troubleshooting networked applications is often difficult. Key ingredients to SPD Server troubleshooting are the name server and SPD Server host process log files. With those two log files, you can reconstruct some of the 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 Startup Failed

Check the name server log file. The log should contain good clues about the problem. Some common things to watch for include:

1. Invalid *-licensefile* file specification.
2. *-licensefile* specifies a file with invalid contents.
3. The name server port is in use by another process. Check for another name server process running already on the same node.

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

SPD Server Host Startup Failed

Check the SPD Server host log file for clues. Some common things to watch for include:

1. *-nameserve*r node name is incorrect.
2. *-nameserverport* specifies wrong port number if SPD Server name server is running with a non-standard port assignment.
3. *-parmfile* file specification is invalid or specified file does not exist.
4. *-libnamefile* file specification is invalid or specified file does not exist.
5. Contents of specified *-libnamefile* does not conform to expected syntax. Check SPD Server host's log for messages about which entries are invalid.
6. *-acldir* option was omitted from the command line.
7. *-acldir* option specifies an invalid directory path for locating the SPD Server password file or the specified directory path does not contain a valid SPD Server password file.

SAS LIBNAME Assignment Failed

On the SAS side, first attempts to diagnose a failure depend on the error messages from the SPD Server LIBNAME engine via the SAS LOG output. In most circumstances you should be able to diagnose the reason for the failure from this message. Some common problems include:

1. **Invalid specification of the LIBNAME engine selector in the LIBNAME statement.** The SPD Server engine name is sasspds and is misspelled in the following LIBNAME statement.

```
1? 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.
```

2. **Invalid specification of the logical LIBNAME domain name in the LIBNAME statement.** The domain name 'test' is not defined in the SPDS name server sunspot.spdsname.

```
2? 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.
```

3. **No name server is running on the specified node name or no name server is available at the specified port address.** In this case, no name server is running on the specified node `stelling`. This same message would be generated if the port address is incorrect.

```
3? 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.
```

4. **An invalid or unknown node name is specified in the LIBNAME statement.** In this case, node `xxx` is not accessible in the network.

```
4? 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.
```

5. **Invalid SPD Server user password specified in the LIBNAME statement.** In this case, the SPD Server user ID is derived from the UNIX user ID running the SAS session. The SPD Server password file has an entry for this SPD Server user ID, but the password is not xxx.

```
8? 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.
```

6. **Invalid SPD Server user ID specified in the LIBNAME statement.** In this case, the SPD Server user ID xxx does not exist in the SPD Server host's password file. The resulting failure message is the same as for the invalid password case.

```
10? 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.
```

Using SETINIT to Extend SPD Server Software

When you receive SPD Server software, the licensing information is pre-initialized. When you contract to renew the license, a paper SETINIT provides you with the licensing information.

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

Create the SETINIT.LIC File

1. Create or modify a file named **spds.lic**. This is the file that the SPD Server name server points to with the license file parameter.

   The following is an example of the spds.lic file:

```
PRODUCT     = SPDS
VERSION     = 4.4
OSNAME      = AIX
SITENAME    = TEST SPDS SITE
SITENUM     = 00999999
NUSERS      = 0
GRACE       = 062
WARN        = 031
SERIALCHECK = 0
EXPIRE      = 03OCT2006
MODEL       =
CPUSERIAL   =
PASSWORD    = 12345678.1
```

2. After you modify the spds.lic file, you can start your SPD Server environment.

SAS Scalable Performance Data Server 3.x and SAS Scalable Performance Data Server 4.4 Compatibility

---

Introduction

This document addresses compatibility questions about upgrading to SPD Server 4.4 from SPD Server 3.x. It also explores migration planning possibilities for SPD Server 3.x and SPD Server 4.4 data stores.

SPD Server 4.4 introduces a new on-disk storage structure for most SAS standard data objects, including tables, catalogs and utility files. The new technology makes SPD Server 3.x data stores incompatible with SPD Server 4.4 data stores. SPD Server 2.x customers that want to upgrade to SPD Server 4.4 must first convert their data stores to SPD Server 3.x format, and then convert the data stores from SPD Server 3.x format to SPD Server 4.4 format. The SPD Server 3.x documentation contains more information about converting SPD Server 2.x files to SPD Server 3.x files.

Some SPD Server 3.x customers that already have data stores in SPD Server 3.x format will be content to continue using SPD Server 3.x on the existing data stores, and use SPD Server 4.4 for new projects and new data stores. Other SPD Server 3.x customers will want to convert their 3.x data stores to SPD Server 4.4 format without having to copy the entire SPD Server 3.x data store. Both SPD Server 3.x and SPD Server 4.4 formatted data stores can share the same physical storage domain on an SPD Server, but you must specify the specific SPD Server engine component used to access the differing SPD Server 3.x and 4.4 data stores.

SPD Server 4.4 is designed for use with SAS 9.1.3. SPD Server 4.4 is distributed only as a 64-bit environment application for the following operating environments: Solaris by Sun, AIX by IBM, and HP/UX by Hewlett-Packard. The SPD Server media contains empty directory structures where 32-bit binaries resided in earlier releases. The

SASSPDS client module for Windows is still a 32-bit application.

SPD Server 3.x and SPD Server 4.4 Sasspds Engines

SPD Server 4.4 includes a **sasspds** engine component for SAS 9.1.3 clients. The **sasspds** distributed with SPD Server 4.4 provides client access to SPD Server 4.4 data stores. SPD Server 3.x includes a **sasspds** engine component for SAS 8.x clients. The **sasspds** distributed with SPD Server 3.x provides client access to SPD Server 3.x data stores. The **sasspds** engine component for SAS 9.1.3 clients is not compatible with SAS 8.x clients, and vice versa.

You must use the **sasspds** engine for SAS 9.1.3 clients to take advantage of SPD Server 4.4. For example, the following LIBNAME statement automatically associates to an SPD Server 4.4 environment when submitted:

```
libname clients8 sasspds 'CLIENTS99'
     host='RUMBLE'
     user='ADMIN'
     prompt=YES;
```

Tables that are created using the SPD Server 4.4 **sasspds** engine become SPD Server 4.4 tables. Both SPD Server 3.x and 4.4 formatted data resources can share space in the same physical storage domain, but you must configure the SPD Server 3.x and 4.4 jobs to use the appropriate version of the SPD Server **sasspds** engine.

The SAS client engines for SPD Server 3.x and SPD Server 4.4 are both called **sasspds**, but they are different engines that will only work with their respective versions of SPD Server and SAS. As a result, you must properly set the *–path* option for the appropriate **sasspds** engine in your SAS configuration file, or you must specify separate data store paths as a command option when you invoke SAS.

If your site maintains working installations of both SPD Server 3.x using SAS 8.x and SPD Server 4.4 using SAS 9.1.3, SAS clients can only access the SPD Server that corresponds to the version of the SAS session.

SPD Server 3.x and SPD Server 4.4 Compatibility Issues

If you have SAS 8.x and SPD Server 3.x data stores in place, you should decide how

to configure SPD Server 4.4 and manage existing data stores in a manner that best exploits the new processing power of SPD Server 4.4 and SAS 9.1.3.

Existing SAS 8 / SPD Server 3.x environments cannot access data stores created by a SAS 9.1.3 / SPD Server 4.4 environment, and vice versa. SPD Server 3.x component files operate with SAS 8.x and end with **.spds8** file extensions. SPD Server 4.4 component files operate with SAS 9.1.3 and end with **.spds9** file extensions. The different file types require two different namespaces. Separate namespaces and different versions of SAS isolate the two different SPD Server installations. However, both SPD Server applications can share the same physical directories when storing and retrieving data resources.

Operating isolated versions of SPD Server 3.x and SPD Server 4.4 in the same environment has tradeoffs. It may be advantageous for SPD Server 3.x and SPD Server 4.4 data stores to coexist in the same storage environments. It may be disadvantageous to have two versions of SPD Server in the same environment, requiring two versions of SAS (SAS 8.x *and* SAS 9.x), and common resources such as password files and ACL files might be shared between the two applications.

SPD Server 3.x and SPD Server 4.4 Coexistence

Administrators will want to use a phased plan to migrate applications and data stores from SAS 8.x and SPD Server 3.x to SAS 9.1.3 and SPD Server 4.4. Administrators should test SPD Server 4.4 installations before going 'live' with the software. System owners seek a comfort level and reassurance that SPD Server 4.4 will not interrupt service in their computing environment. During the phased testing and assessment period, it can be useful to share the same LIBNAME domains between separate SAS and SPD Server environments.

Administrators must decide whether to run both versions of SAS and SPD Server (and the formatted data stores) concurrently, or to convert all existing SAS 8 / SPD Server 3.x data stores to SAS 9 / SPD Server 4.4 format.

What are the issues with running SPD Server 3.x and SPD Server 4.4 environments in isolation? How do you manage this configuration?

You will need to operate two complete SPD Server environments (3.x and 4.4) as well as two SAS environments (8.2 and 9.1.3) on your machine. SAS 8 jobs that are configured to work with your SPD Server 3.x installation require no changes. To migrate SAS 8 jobs to SAS 9 jobs for SPD Server 4.4, you will need to modify the

following SAS entities to reflect the port number for SPD Server 4.4:

- LIBNAME statements
- SQL CONNECT statements
- ODBC data source definitions

The SPD Server 4.4 port number is the port number that was assigned to the SPD Server 4.4 Name Server *-listenport* during SPD Server 4.4 installation.

Sharing Parameter Files

Although SPD Server 3.x and SPD Server 4.4 cannot share the same data files, you can use SPD Server 3.x parameter files to configure your SPD Server 4.4 installation. You can copy your existing SPD Server 3.x parameter files into your SPD Server 4.4 installation directory and use them without modification. SPD Server 4.1 does introduce some new parameters of interest, but the SPD Server 3.x server parameter files will work with SPD Server 4.4.

However, there are required separations to maintain. Identically-named SPD Server 3.x and SPD Server 4.4 CATALOGs *cannot* coexist in the same LIBNAME domain.

Migrating SPD Server 3.x Data to SPD Server 4.4 Data

What should legacy users do with their SPD Server 3.x data stores? The answer depends on how often you plan to continue using SAS 8 in your computing environment after you install SAS 9.

If your organization exclusively uses SAS 9 to drive its SAS processing, there may be no good reason to keep SPD Server 3.x data stores. On the other hand, if your organization utilizes data resources from both SAS 8 and SAS 9, you should decide whether to transform your SAS 8 / SPD Server 3.x data stores into SAS 9 / SPD Server 4.4 format.

SPD Server 4.4 includes a conversion utility called SPDSCONV that is used to convert SPD Server 3.x formatted tables to SPD Server 4.4 formatted tables. If your organization does not intend to maintain a SAS 8 environment, then the SPDSCONV utility is the quickest way to to convert the tables to SPD Server 4.4 format. After running the SPDSCONV utility, the 3.x tables will no longer exist, so it is *essential* that all tables to be converted are backed up prior to conversion, to avoid the possibility of

lost data. See the SPDSCONV documentation for further information on using the SPDSCONV utility.

In addition to the SPDSCONV utility, there are two methods that can be used to copy SPD Server 3.x formatted tables into SPD Server 4.4 tables. These methods are documented as follows:

1. Start SAS 8 and the SPD Server 3.x **sasspds** client.
2. Use PROC CPORT to export each SPD Server 3.x table that you wish to convert to SPD Server 4.4 format. After you export all the tables you want to upgrade, close SAS 8.
3. Start SAS 9 and the SPD Server 4.4 **sasspds** client.
4. Use PROC CPORT in SAS 9 to import each of the tables you exported, then close SAS 9.

If you have the SAS/SHARE product, start SAS 8 with the SPD Server 3.x **sasspds** client, and follow the instructions below:

1. Make the SPD Server 3.x LIBNAME assignments that you need.
2. Start PROC SERVER to make your SAS 8 session a SAS/SHARE server.
3. Start a separate SAS 9 session with the SPD Server 4.4 **sasspds** client. Use the REMOTE engine via the SAS/SHARE server to submit LIBNAME assignments for the SPD Server 3.x domains that you want to copy.
4. In your SAS 9 session, submit SPDS LIBNAME assignments for the SPD Server 4.4 domains that you want to populate.
5. In your SAS 9 session, use PROC COPY to transfer the exported SPD Server 3.x tables to the SPD Server 4.4 domains.

Note: If you are working with Windows and SAS 8, and if you have SAS/ACCESS for ODBC, you can configure ODBC data sources that can access the SPD Server 3.x environment and/or SPD Server 4.4 environment and transfer the data through a PROC COPY via the LIBNAME assignments to the ODBC data sources.

Backing Up and Restoring SPD Server 3.x and SPD Server 4.4 Files

What should you do with previously saved SPD Server backups? Should you back up tables from LIBNAME domains that can contain a mixture of SPD Server Release 3.x and SPD Server 4.4 tables?

SPD Server 3.x and SPD Server 4.4 backup utilities and files are not compatible. You must use the SPD Server Release 3.x backup utility and connect to SPD Server 3.x to back up SPD Server 3.x tables. To restore tables that were backed up with SPD Server 3.x, you must connect to SPD Server 3.x and use the SPD Server 3.x restore utility.

Once a dataset is either created or converted to an SPD Server 4.4 table, you must connect to an SPD Server 4.4 server and use SPD Server 4.4 backup and restore utilities. The SPD Server 4.4 backup utility prints an error message if you try to backup an SPD Server 3.x table. When an SPD Server 3.x table is converted to an SPD Server 4.4 table, the table is considered to be a new table for backup purposes, and the first backup that will be done is a full backup on the table.

SAS Scalable Performance Data Server 3.x to SAS Scalable Performance Data Server 4.4 Conversion Utility

Contents

Introduction

SPD Server 4.4 uses improved architectures that enable features like the ability to support data tables that contain over 2G observations. The new architectures provide functionality that is needed in today's data marts, but the current generation of SPD Server tables use metadata architectures that are not backwards compatible with SPD Server 3.x software.

SPD Server 4.0 to 4.4 tables use architectures that utilize SAS 9 metadata. SPD Server 3.x tables use architectures that utilize SAS 8 metadata. *The two metadata architectures are not compatible.* However, SPD Server 4.4 provides a conversion utility SPDSCONV that permits SPD Server 3.x customers to convert existing tables for use with SPD Server 4.0 and 4.4. The SPDSCONV utility is designed to be run by the SPD Server Administrator.

Note: SPD Server Administrators should *ensure* that all images of the SPD Server 3.x data sets to be converted are completely backed up before beginning the conversion process.

Overview of the SPDSCONV Utility

The SPDSCONV utility converts SPD Server 3.x metadata files for use with SPD Server 4.4. The conversion updates the physical structure of the metadata file and renames the files. The SPDSCONV utility will also update the data partition files if the SPD Server 3.x tables being converted contain compressed data.

You can identify SPD Server 3.x table files by the filename extension. SPD Server 3.x table files end with the filename extension **.**spds8. SPD Server 4.4 table files end with the filename extension **.**spds9. All tables that are upgraded to be compatible with SPD Server 4.4 will have the filename extension **.**spds9.

SPD Server 4.4 index files differ from SPD Server 3.x index files. SPD Server 4.4 index files allow greater numbers of observations than the SPD Server 3.x index files. SPD Server 3.x

index files are not compatible with the SPD Server 4.0 and 4.4 environment.

The SPDSCONV table conversion utility does not recreate index files. When you use the SPDSCONV utility to convert tables from SPD Server 3.x to SPD Server 4.4 format, the utility automatically deletes physical files that were associated with the old 3.x indexes and are now obsolete. The SPDSCONV utility does offer an option to create a SAS job file that you can run in the SPD Server 4.4 environment to recreate the SPD Server 3.x index files for use with SPD Server 4.4.

If you choose to create the SAS job file to recreate SPD Server 3.x indexes for use in SPD Server 4.4, the code will resemble the following:

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

quit;
```

When you use the SPDSCONV utility, you can specify the destination directory for the SAS job file that you create, but the SPDSCONV utility names the job file that your create. The utility generates SAS job file names by adding the text string *_v4ix.sas* to the table name, resulting in names such as mytable*_v4ix.sas*. It's a good idea to defer index recreation because of the intensive computing it can require. Performing index re-creation as an off-peak batch job can be advantageous in busy computing environments.

After converting the indexes, some users may notice that SPD Server 4.4 metadata files are slightly larger than the SPD Server 3.x metadata files. The file size increase is related to the new structures that facilitate SPD Server 4.4's capability to use large tables that more than 2G rows of data.

How does SPDSCONV work? When converting a table, the SPDSCONV utility first reads the original SPD Server 3.x metadata file and creates a new SPD Server 4.4 metadata file. Both these files are locked during the conversion process. The lock prevents any outside access to the files by other users while changes are being made. If the conversion process encounters problems, the SPD Server 4.4 metadata file is deleted and the original SPD Server 3.x table remains intact.

SPDSCONV reads the SPD Server 3.x metadata file a section at a time, recreating each structure in the SPD Server 4.4 metadata file as it is read. After the SPD Server 4.4 metadata file is fully populated, the data partition file component is checked to determine if updates are required.

If SPDSCONV detects the presence of compression block headers, then the data partition file

contains SAS 8 compression information that is not compatible in SPD Server 4.4, and the data partition files must be updated. SPDSCONV updates the data partition file by overwriting the compression block headers. SPDSCONV does not change the size of the data partition file, of any file components, or any of the data contained in the files. The increase in metadata file size is very modest and represents only a small percentage of storage space when compared to its corresponding data partition file component.

Once SPDSCONV updates the data partition file, there is no provision to restore or recreate the original SPD Server 3.x data partition file. You should ensure that you have complete backup images of the SPD Server 3.x datasets that you intend to convert prior to running the conversion.

After the SAS job file recreates the SPD Server 3.x indexes for use with SPD Server 4.4, all remnants of the SPD Server 3.x table are deleted. The SPDSCONV utility does not perform ACL checks during the conversion. The individual running the SPDSCONV utility (usually an SPD Server Administrator) 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 4.4 table retains the same SPD Server owner as the SPD Server 3.x table.

Using SPDSCONV

The SPDSCONV program is a command-line utility. You use a set of command-line options and parameters to specify the name and location of tables you wish to convert, then specify the options you desire for your conversion. If your SPD Server software is installed on a UNIX platform, refer to the UNIX SPD Server installation guide for information on setup required prior to running SPDSCONV.

The form of the command line is as follows:

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

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

Options for the SPDSCONV command are:

-d *pathname*
: the directory path corresponding to an existing SPD Server LIBNAME domain. The *pathname* specification should be the same as the PATHNAME= directory path found in the *libnames.parm* file.

-l *logpath*
: the directory path to store SAS job files created during the conversion

process. The default logpath setting is the directory where the SPDSCONV command is issued.

-a

converts *all* SPD Server 3.x compatible tables in the -d *pathname* directory to SPD Server 4.4 compatible tables.

-j

creates a SAS job in the log directory for each SPD Server 3.x table conversion where indexes are present. When run, the SAS job recreates the indexes on the SPD Server 4.4 table. The SAS job must be run after the SPDSCONV utility completes. Because index recreation may be computation-intensive, users may want to schedule SAS index recreation jobs as a SAS batch jobs for off-peak hours. The utility generates the name of the SAS job file as follows:

TableName_*v4ix.sas*

where TableName is the name of the SPD Server 4.4-compatible table. The SAS job file contains the SAS language statements necessary to recreate the indexes that the SPD Server 3.x table used. The job file will need to be edited prior to execution to ensure that the proper SPD Server 4.4 LIBNAME is used with the *PROC DATASETS*statement.

-v

create verbose output for the conversion process.

SPDSCONV Utility Examples

- Converting a Table
- Converting Tables and Recreating Indexes

Suppose you have the 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 SPD Server 4.4 use. The table CT010299 has no indexes and you want a verbose output of the table conversion.

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

Converting Tables and Recreating Indexes

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

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

Using the SAS Scalable Performance Data Server Name Server to Manage Resources

---

Using the Name Server to Manage Resources

Managing Computing Resources with a Name Server

The name server gives system administrators the ability to manage computing resources without disturbing system users because it maps logical names referenced in SAS programs to the physical location of SPD Server tables. Thus, a name server allows the system administrator to add, remove or reallocate disk space and computing power without having to change SAS source code. Nor is there the necessity of informing others about changes in resource allocation as long the name of the machine that hosts the name service remains the same.

To demonstrate how you can configure so that you are spreading the processing load among a number of machines in a network, we present here an example configuration scenario.

An Example Configuration Scenario

Assume that a corporation maintains a network of computers. Within the network there is a mix of 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. Among the dedicated servers, we find

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, etc.

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 periodically must access worldcpu**.** For this reason 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 which contain only their data, allowing them to bring the information to their desktops for analysis.

The SPD Server system administrator runs the name server on the namecpu machine. Consequently, namecpu must be accessed by every machine in the network that wants to locate an SPD Server table. Additionally, the administrator must run a data server on the worldcpu, and asiacpu machines. Here is how the administrator should configure the servers to distribute the processing load.

Running the Name Server on Machine Namecpu

Invoke the name server with the '-listenport' option. The value specified for the option should be a valid TCP/IP port number. You will use this same port number when invoking SPD Server on each server, worldcpu and asiacpu.

Configuring SPD Server on Worldcpu

The file libname.parm, which resides on worldcpu, contains the following line:

```
libname=world pathname=/spds;
```

This line 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 directory /spds on machine worldcpu. The SAS LIBNAME statement that invokes the SPD Server engine and makes this association is

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

A name server means that users do not have to remember on which machine a domain resides. All they need to remember is that the SAS domain 'world' contains the tables they need. This reflects the users' reality. From the users' point of view, they do not need to know the name of the machine that stores the domain. The machine can change without the users' knowledge; users just want to know that their SAS programs will continue to run as before.

Invoke SPD Server specifying namecpu as the value for the "-nameserver" option. The value for the "nameserverport" must match the port number that you used to start the name server on that machine.

Setting Up asiacpu, a Departmental Server for the Asia Department

The file libname.parm, which resides on asiacpu, contains the line:

```
libname=asia pathname=/spds;
```

This line instructs the SPD Server running on asiacpu to register the combination

```
(asia, asiacpu, /spds)
```

with the name server. Thereafter, when a SAS LIBNAME statement contains the domain name 'asia' in combination with the appropriate name server, it will locate SPD Server tables in directory /spds on machine asiacpu. The SAS LIBNAME statement that invokes the SPD Server engine and makes this association is

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

Note that the value following the LIBNAME server= is the same in all these LIBNAME statements. The reason is that both SPD Servers use a common name service. Again, Asia users do not need to know the machine that provides storage for their domain.

Which SAS Program Statement Runs Where?

Assume a user in the Asia department needs to create an SPD Server table on the departmental server asiacpu. To do this requires extracting data from an SPD Server table named 'alldata'. The user knows that the table resides in the domain 'world.' This executes the following SAS (desktop machine) statements:

```
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;
```

The program extracts records from an SPD Server table named 'alldata' which resides in the domain 'world.' As you already know, this domain is stored on machine worldcpu in directory /spds. Because the 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 is running on our Asia user's desktop machine. Consequently, it is the desktop machine that actually examines each row looking for the string 'Japan'. It then forwards the row to the machine where the output table resides, in this case, asiacpu.

Disk space for the output table "mydata" is allocated in the directory /spds which resides on **asiacpu**. The processing work, transferring data received from the user's desktop machine to the SPD Server table, is performed by asiacpu as well.

As you can see, the processing required to create our output SPD Server table was distributed among three machines. However, the user's desktop machine required no permanent disk space. Because the SAS WHERE clause executes on the machine that stores the table, only selected rows (matching the WHERE clause) are sent over the network to the desktop. This significantly reduces network traffic and the time needed to complete a SAS program.

Administering and Configuring SPD Server Using the SAS Management Console

---

The SAS Management Console

The 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, application modules that are designed to create metadata for a specific type of resource.

For example, administrators can use the SAS Management Console to configure SPD Server user and group passwords and ACLs instead of using the traditional SPD Server **`psmgr`** utility and PROC SPDO statements.

By default, SAS Management Console looks for plug-ins in the **`plugins`** subdirectory of each installed SAS product. The SAS Management Console plug-in file for SPD Server is located at

```
SASROOT/spds44/spdssmc/sas.smc.SpdsMgr.jar
```

Note: **`SASROOT`** represents the path to the base directory of the SAS software installation on your client machine. **`spds44/`** represents the installed SPD Server software directory. The name of the installed SPD Server software directory varies according to the specific version and release of your SPD Server software. For example, the path to your SPD Server Java plug-in file might begin with **`SASROOT/spds44`**, **`SASROOT/spds44tsm1`**, or **`SASROOT/spds44tsm2`**, depending on whether you have the original SPD Server 4.4 software, or the first or second maintenance release of the SPD Server 4.4 software.

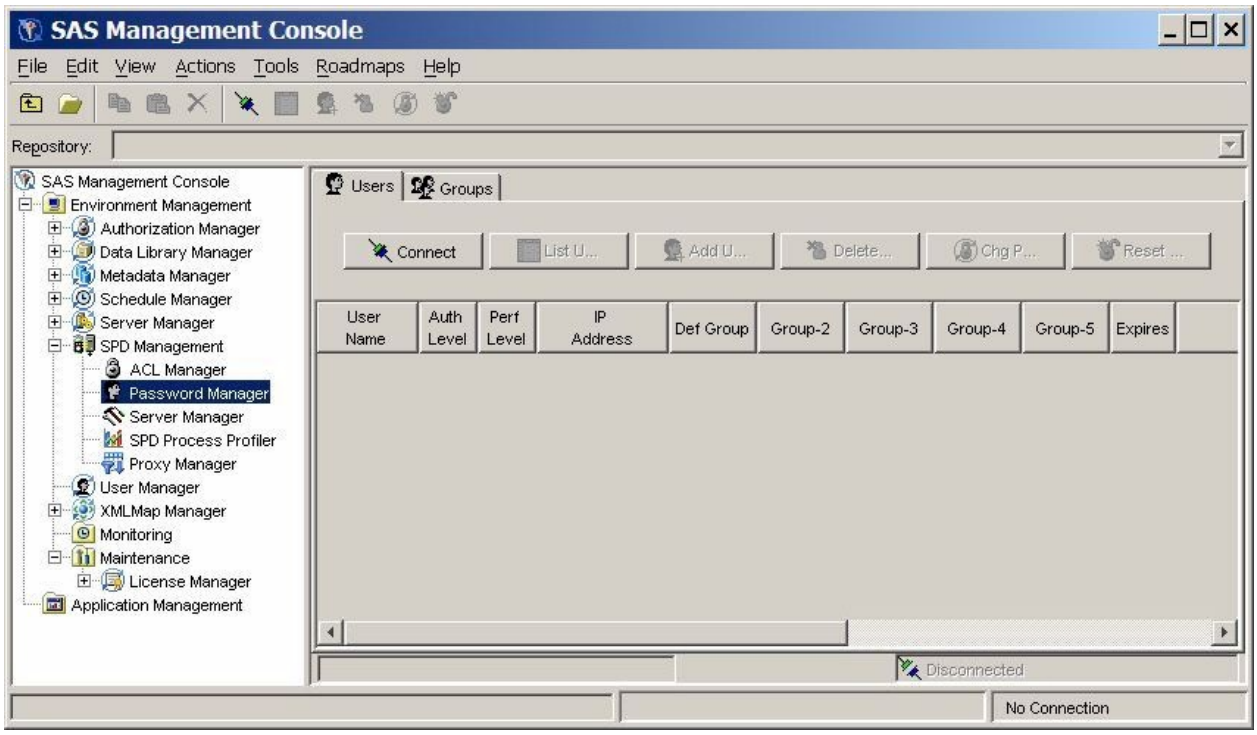
Accessing the SPD Server Manager in SAS Management Console

The left portion of the SAS Management Console contains a navigation tree of available management tools. Select the **`SPDS Manager`** folder to access SPD Server

services. The SPDS Manager folder contains the SPD Server ACL Manager, Password Manager, Server Manager, SPD Process Profiler, and Proxy Manager.

Password Manager

If you open the Password Manager in the SAS Management Console window when no server connection exists, the display resembles the following:



There are two window tabs: **Users** and **Groups**. When no server connection exists, no data is displayed and the only permitted operation is connecting to an SPD Server.

Connecting to an SPD Server

Select **Connect** button on the **Users** tab of the SAS Management Console window to open the Connect to SPD Server window:

The Connect to SPD Server Manager window contains input fields for the following components. The components follow the same usage as a LIBNAME statement to connect to an SPD Server host.

Server — the name of the SPD Server host

Port — the SPD Name Server listen port

Domain — the SPD domain

User — user name that has privilege to be ACL special

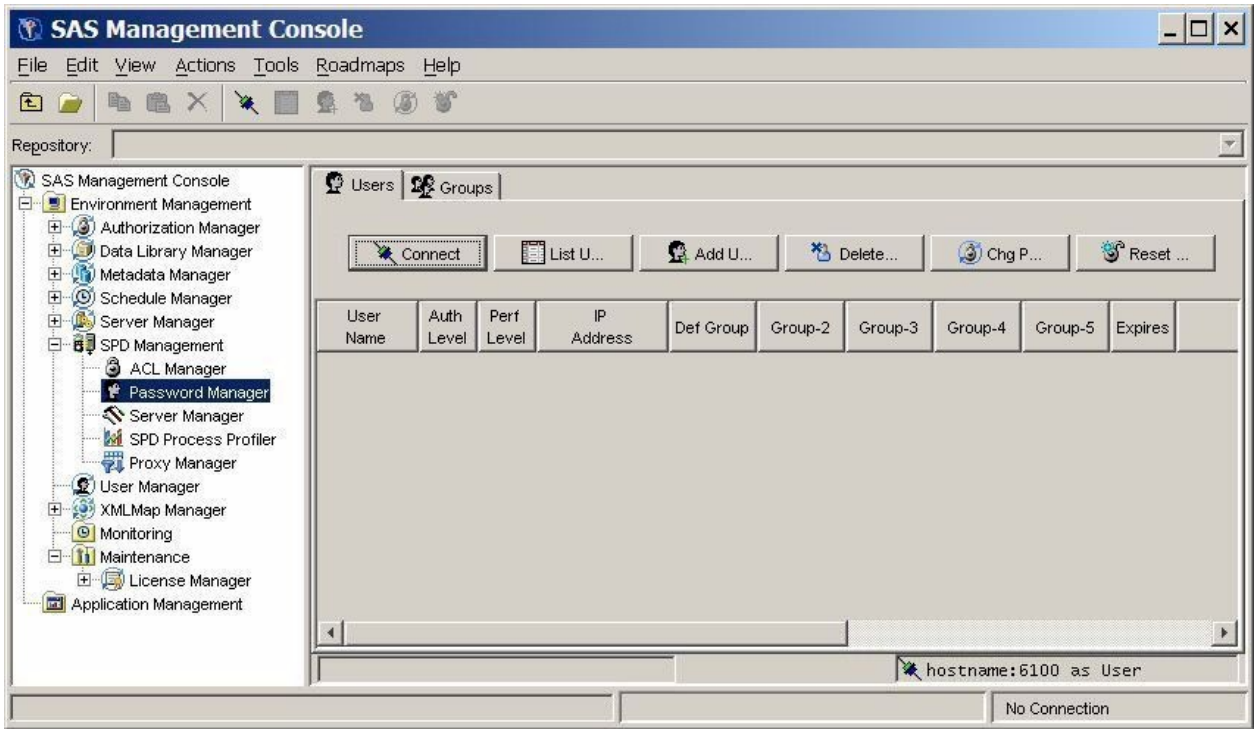
Password — password associated with the user name

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, the remaining command buttons on the **Users** tab are enabled. The status bar at the bottom of the **Users** 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 **Users** tab.

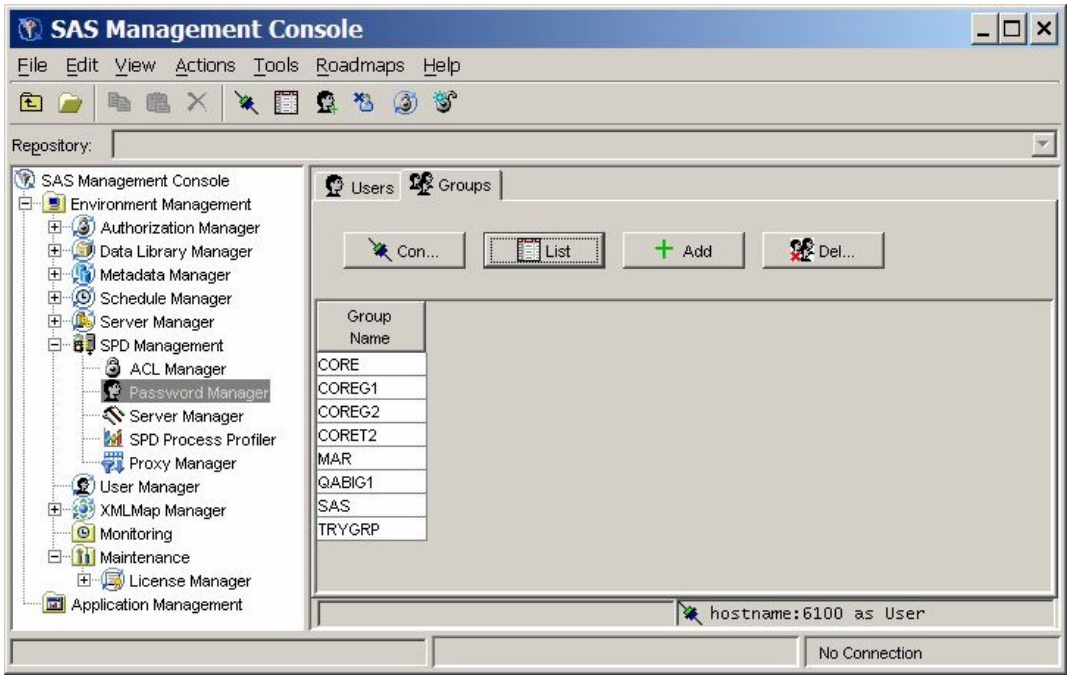


Users and Groups

Use the **Users** and **Groups** tabs in the SAS Management Console window to access either individual user or user group configuration information.

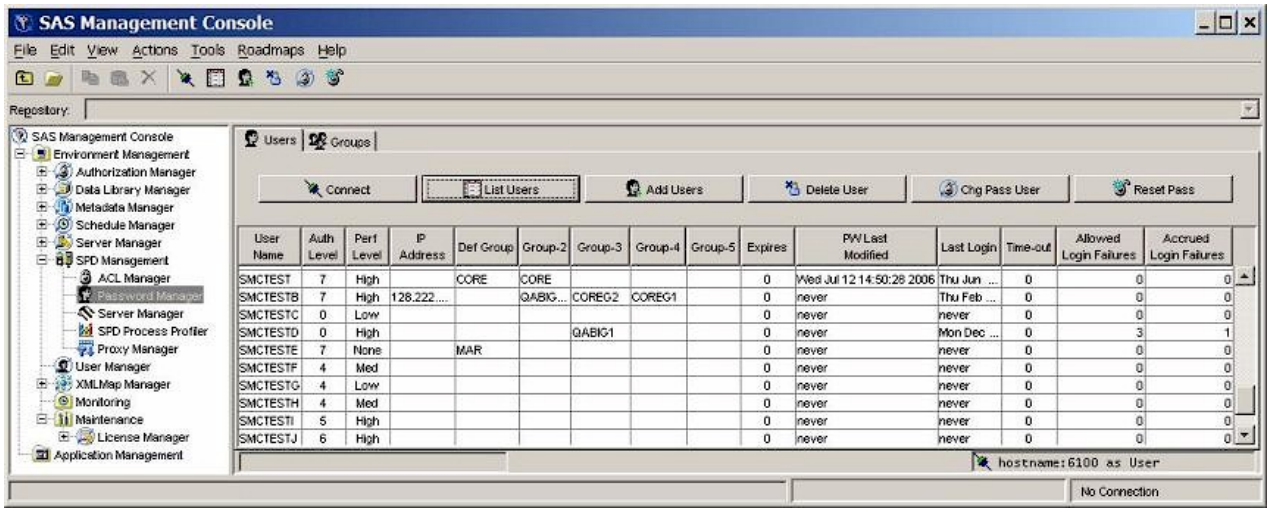
Groups Tab

On the **Groups** tab of 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 existing defined groups by default.



Users Tab

On the **Users** tab of the SAS Management Console window, click **List** to display the defined users and groups.



The window contains the following components:

User Name — the name of the user. This field cannot be changed directly. To change a user name, delete the user and then add a new user.

Auth Level — the numeric authorization level, ranging from 0 to 7. To change the value, select the field and edit it.

Perf Level — the Perf Level 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 — the IP address of the workstation that the individuals listed in the User Name column are using.

Def Group — the default group for a user. Use the drop-down list for the field 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 — the number of days that are remaining until the current password expires. A zero value represents infinity.

PW Last Modified — the date and time of the last modification to a user's password.

Last Login — the date and time of the last user login.

Time-out — the 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**

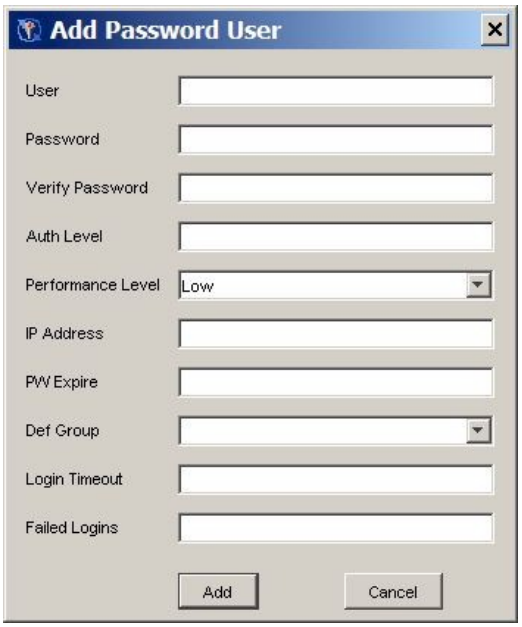
value, the user's access is disabled. For example, if the **Time-out** value is 7, and if a given user does not log on at least every seven days, the user's access is disabled. A zero value in the **Time-out** field represents infinity, so a user account with a zero value in the **Time-out** field never times out.

Allowed Login Failures — the number of consecutive login failures that is allowed before the user's access is disabled. A zero value indicates that unlimited login failures are allowed.

Accrued Login Failures — the current number of consecutive failed login attempts by this user.

Adding a User

To add a user, click **Add**. Complete the values for **User**, **Password**, **Auth Level**, **IP Address**, **PW Expire**, **Def Group**, **Login Timeout**, and **Failed Logins**.



User, **Password**, **Auth Level**, **Def Group**, and **Failed Logins** are required values. The **IP Address**, **PW Expire**, and **Login Timeout** are optional. If the optional settings are not specified, they default to "no limits".

Adding a Group

To add a group using the **Groups** tab of the SAS Management Console window, click **Add**. Enter a group name in the Add Group window and click **Add**. The group name is added and the list is updated.

Deleting a User

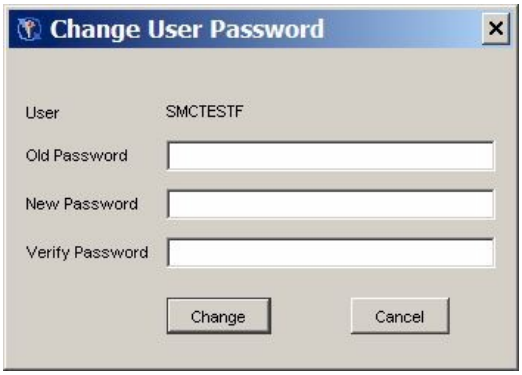
To delete a user using the **Users** tab of the SAS Management Console window, select the user from the list and click **Delete**. The user is deleted and the list is updated.

Deleting a Group

To delete a group from the Groups tab of the SAS Management Console window, select the group from the list and click **Delete**. The group is deleted and the list is updated.

Changing a Password

To change a user's password from the **Users** tab of the SAS Management Console window, select the user and click **Chg Pass**. Enter the user's old password and new password in the Change User Password dialog box, and click **Change**.



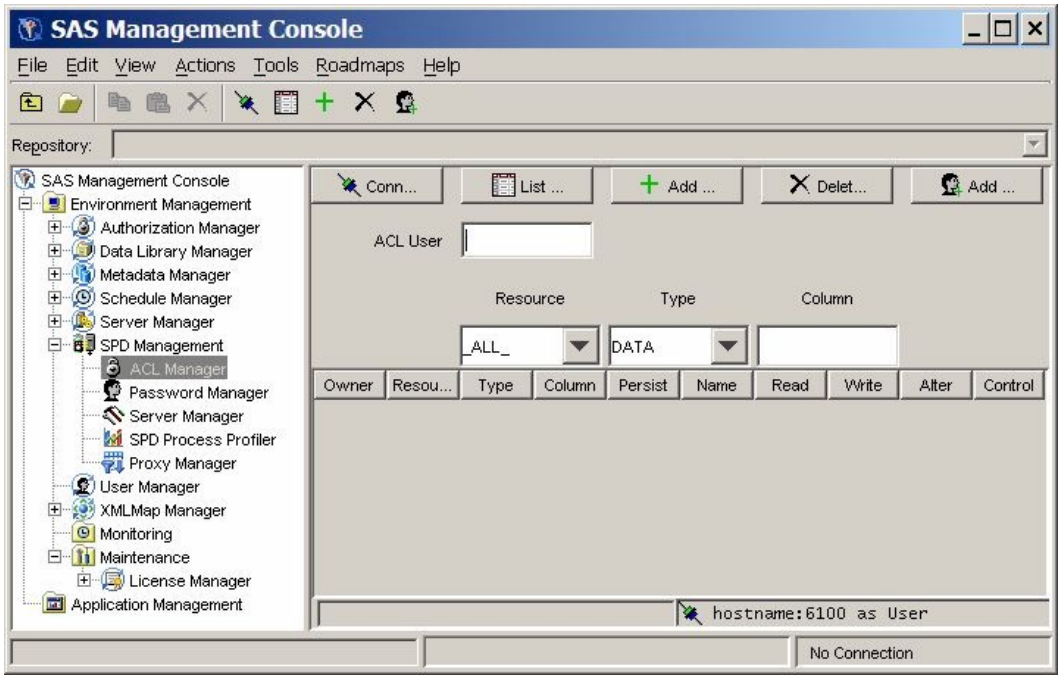
Resetting a Password

To reset the password for a selected user using the **Users** tab of the SAS Management Console window, click **Reset Pass**. Specify the user's new password and then click **Change**. Resetting the password does not require that you know the user's current password. The user will be required to subsequently change the password before he can connect to the server.

You use the **Reset Pass** command to reset a user's password after a user has been disabled. Users can be disabled for excessive login failures or a login timeout.

ACL Manager

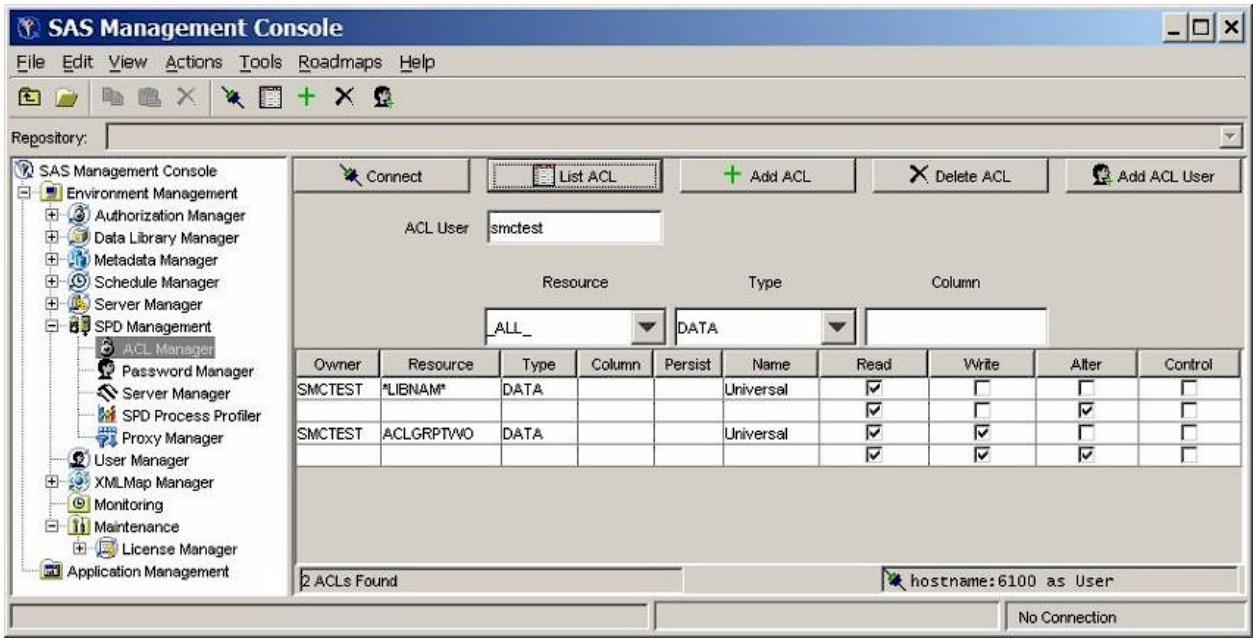
The ACL Manager's display area resembles the following display when no server connection exists:



You must connect to an SPD Server host machine before you can use the SPD Management utilities. The section Connecting to an SPD Server provides detailed instructions on connecting to an SPD Server host.

Listing ACL Resources

Click **List ACL** in the ACL Manager of the SAS Management Console window to display the ACL resources that have been defined.



The ACL Manager display contains the following components:

Owner — the resource owner. This field cannot be changed directly. To change a resource owner, delete the resource and then add a new one.

Resource — the resource name. This field cannot be changed directly. To change a resource name, delete the resource and then add a new one.

Type — the type of resource (for example, DATA, CATALOG, VIEW, or MDDB). The **Type** field cannot be changed directly. To change the **Type** value, delete the current resource and then add a new one.

Column — the column name, if the resource is limited by a column constraint. The column name cannot be changed directly. To change the column name, delete the existing resource and then add a new one.

Persist — a Boolean flag. When set to Yes, **Persist** indicates that the ACL resource definition continues to exist if the referenced resource is deleted. When the **Persist** setting is left 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** represents the default setting for all unnamed groups or users.

Read — if selected, the specified user/group has permission to read this resource.

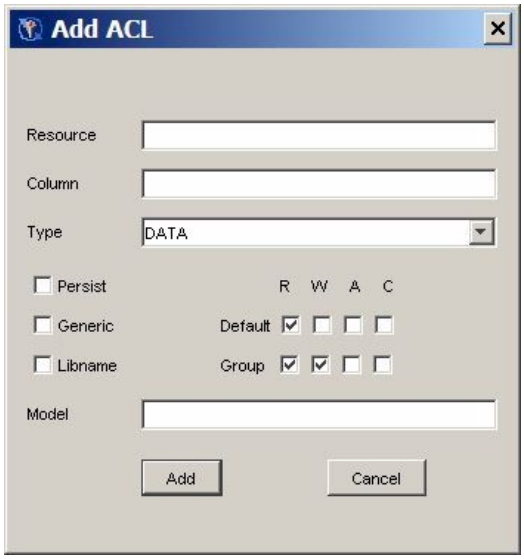
Write — if selected, the specified user/group has permission to write to this resource.

Alter — if selected, the specified user/group has permission to alter this resource.

Control — if selected, the specified user/group has permission to modify permissions of other users and groups associated with this resource.

Adding an ACL Resource

To add an ACL resource, click **Add** in the ACL Manager of the SAS Management Console window, and then complete the values in the Add ACL window.



Resource — the name of the resource to add.

Column — the column restrictions for the resource to be added. If there are none, leave blank.

Type — the type of resource (for example, DATA, CATALOG, VIEW, or MDDB).

Persist — a Boolean flag. When set to Yes, **Persist** indicates that the ACL resource definition continues to exist if the referenced resource is deleted. When the **Persist** setting is left blank, the ACL resource definition is deleted when the referenced resource is deleted.

Generic — select if the resource name is a generic name.

Libname — select 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 as appropriate.

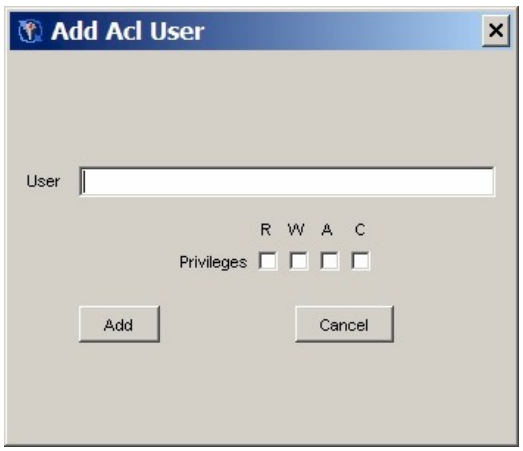
Model — specify the name of another existing ACL resource for this ACL resource to be modeled after.

Deleting an ACL Resource

To delete an ACL resource, select any row in the ACL resource table and click **Delete ACL**. The ACL resource is removed and the list is automatically updated.

Adding 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. When the Add Acl User window opens, enter the **User** or group name, select the boxes that correspond to the default Read, Write, Alter, and Control permissions that you want to grant, and then click **Add**.



The user is added and the ACL listing is automatically updated. An individual user or group cannot be deleted from an ACL. To delete a user, delete the entire ACL resource, then add it back in without the user.

Changing Resource Permissions

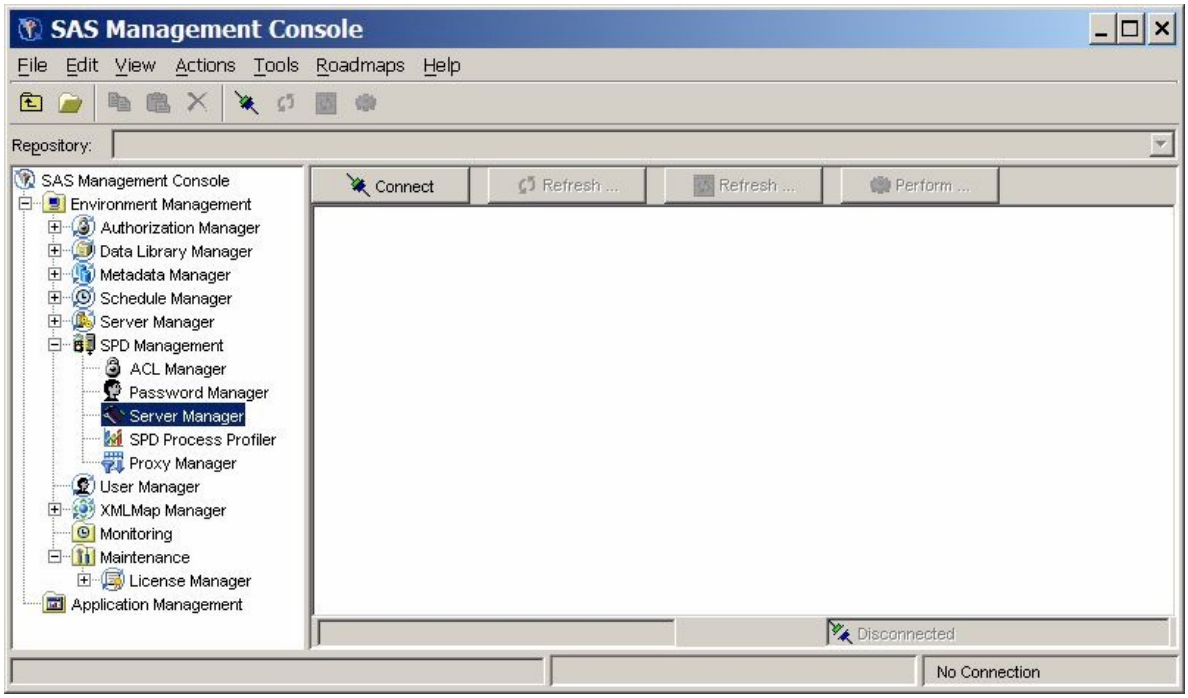
Each ACL resource has at least one set of permissions called universal permissions. Universal permissions are the default permissions for the ACL resource if no other permissions are applied. If any group or user names exist that have permissions for the ACL resource, they will be displayed.

Each set of permissions has four attributes (Read, Write, Alter, Control). To enable the permission simply select its box.

R, W, A, C — select the appropriate default and group permissions to grant Read, Write, Alter, and Control as appropriate.

Server Manager

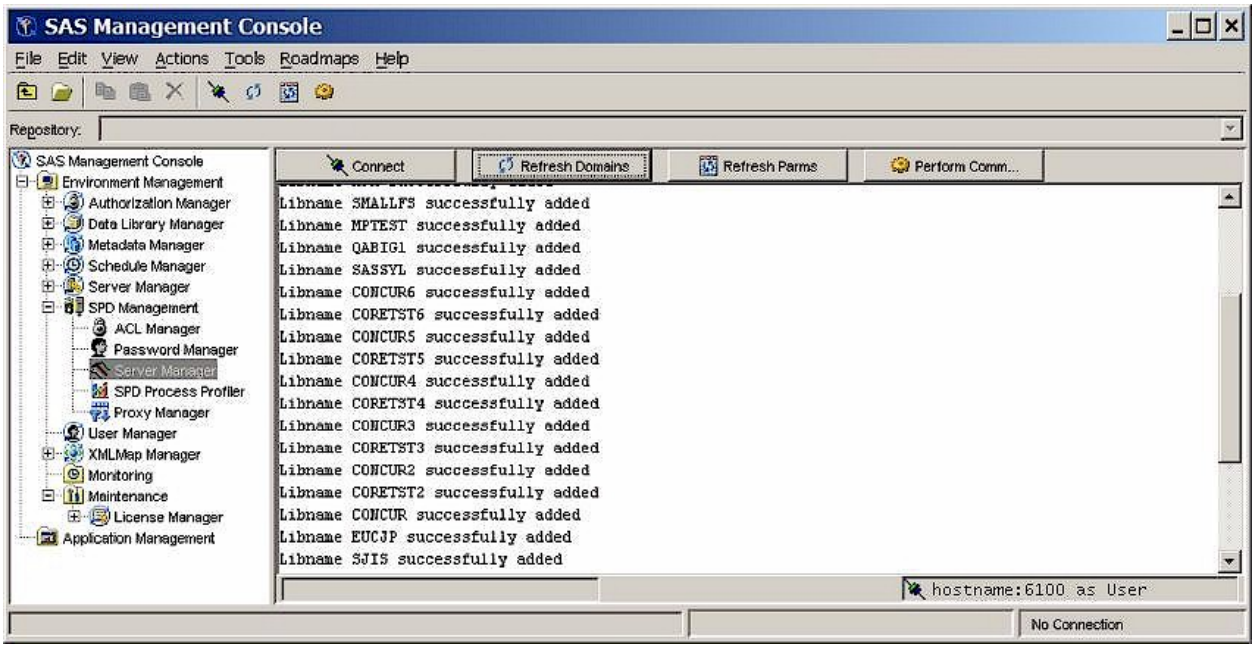
The Server Manager utility in the SPD Management folder of the SAS Management Console window is used to refresh the server's configuration and to run selected SPD Server utilities.



You must connect to an SPD Server host machine before you can use the SPD Management utilities. The section Connecting to an SPD Server provides detailed instructions on connecting to an SPD Server host.

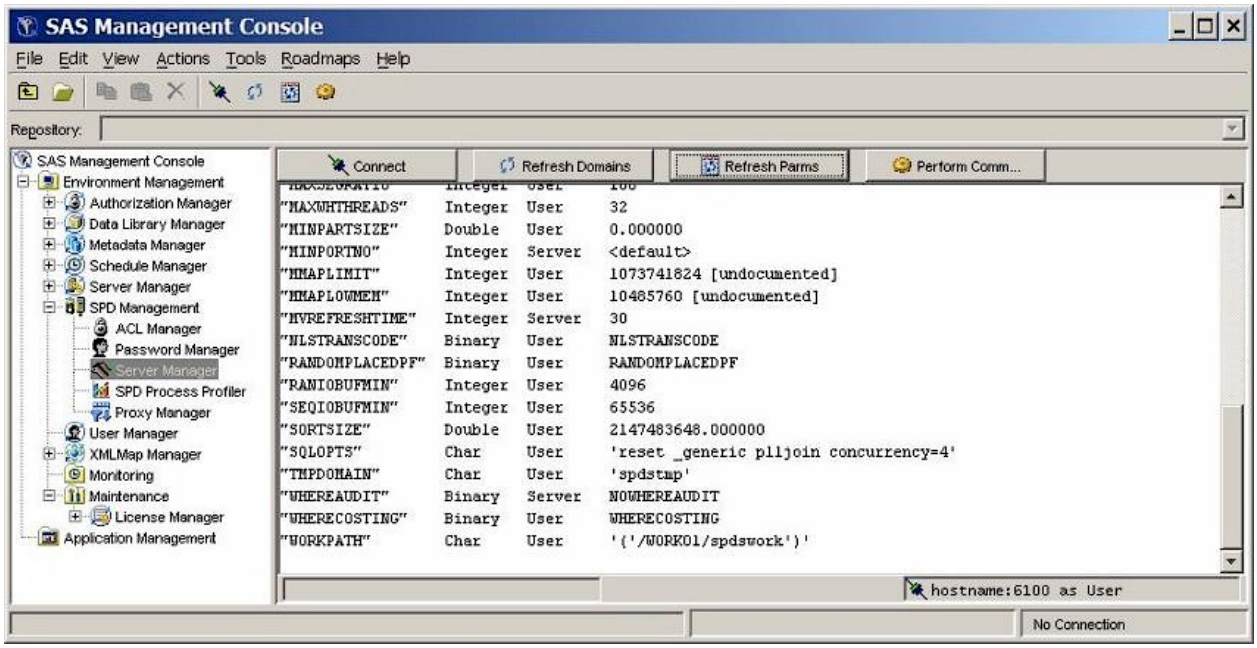
Refresh Domains

Click **Refresh Domains** in the Server Manager to reload the current libnames.parm file. The libnames.parm file describes the list of available domains.



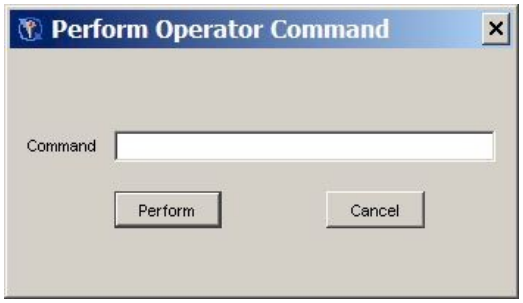
Refresh Parms

Click **Refresh Parms** in the Server Manager to reload the current spdsserv.parm file. The spdsserv.parm file controls server configuration and options.

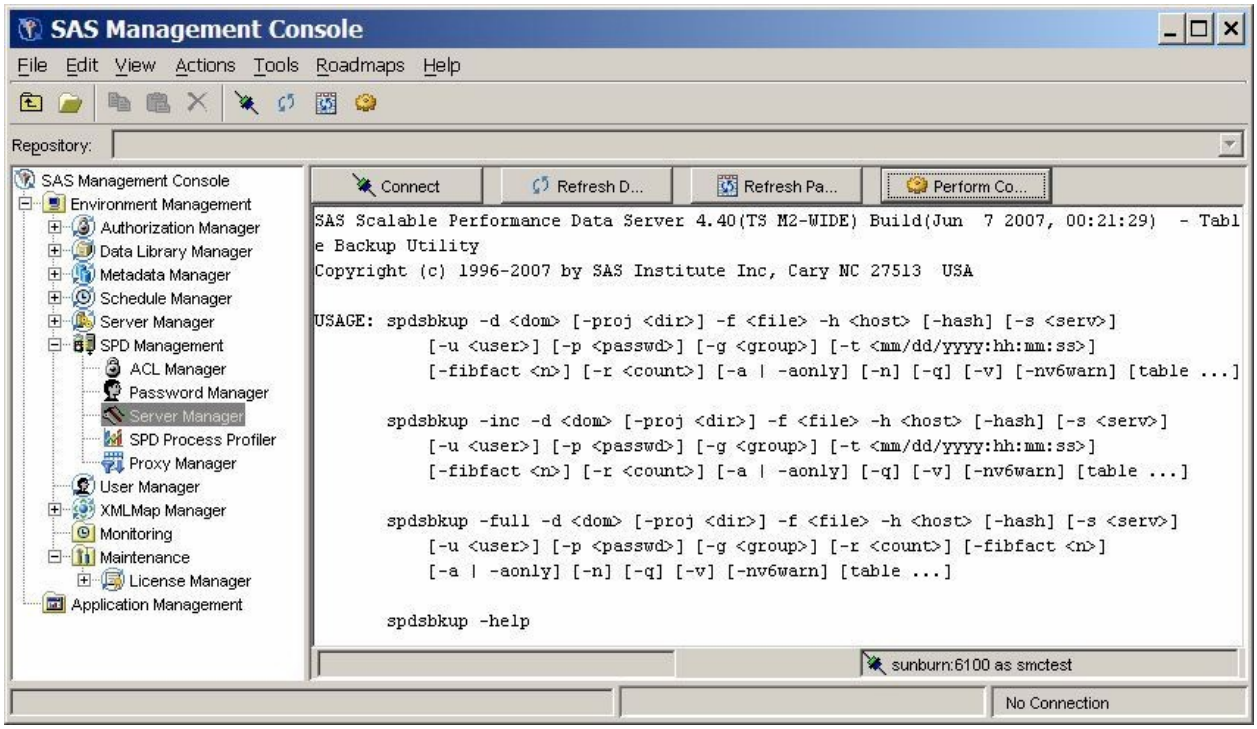


Perform Commands

To run an SPD Server operator command or utility function, click **Perform Command** in the Server Manager. Enter the command or utility in the window, and then click **Perform**.

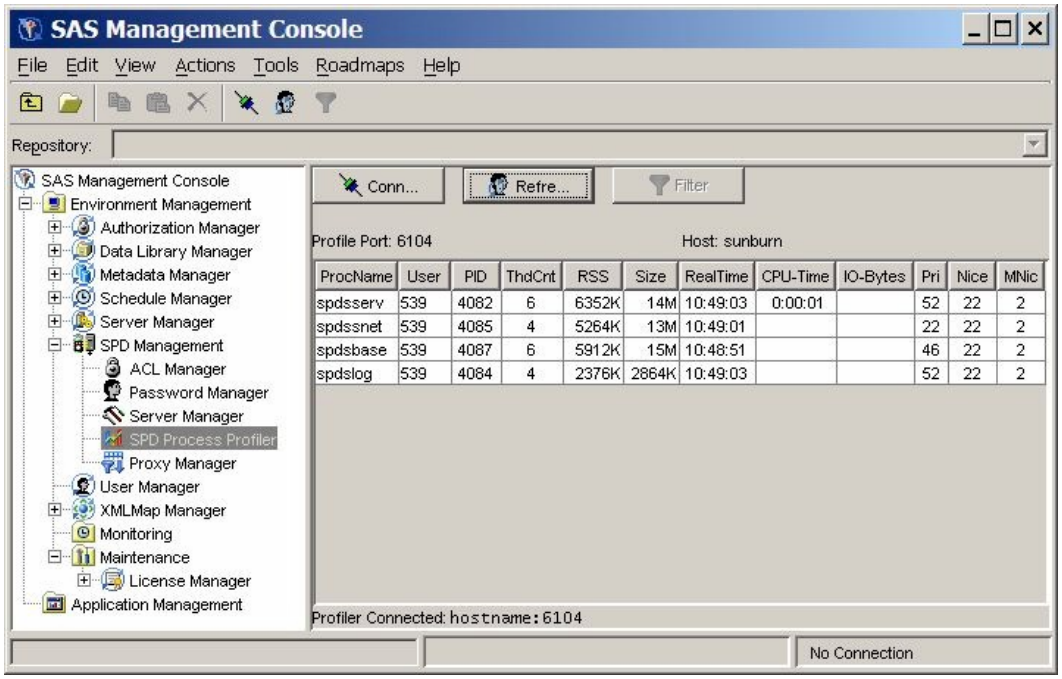


The following example below shows the Server Manager after using **Perform Command** to run a **spdsbkup** command:



SPD Process Profiler

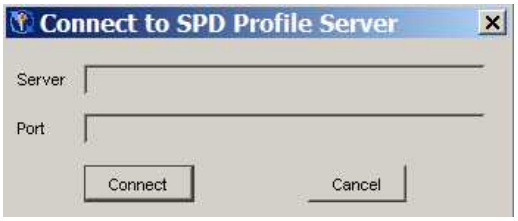
The SPD Management folder in SAS Management Console contains an SPD Process Profiler utility. Use the SPD Process Profile Manager to view server resources that are monitored by the SPD Server Performance Server.



The SPD Server Performance Server gathers SPD Server process performance information and distributes it across the SPD Management section of the SAS Management Console. The SPD Server process performance 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 that are created and terminated by SPD Server users and jobs.

To access a server's process performance information, the SPD Server Performance Server application **spdsperf** must be running for the targeted SPD Server Name Server. SAS Management Console must be able to connect to the SPD Server Performance Server's listening port.

Like the other SPD Management utilities, you must first connect the SPD Process Profiler to the SPD Server Performance Server. Click **Connect** in the SPD Process Profiler to open the Connect to SPD Profile Server dialog box.



Enter your host name or server name, and the SPD Server Performance Server's listening port, and then click **Connect**. (The SPD Server Performance Server's listening port number is specified when you start the SPD Server Performance Server application.)

Once SAS Management Console is connected to the SPD Server 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.

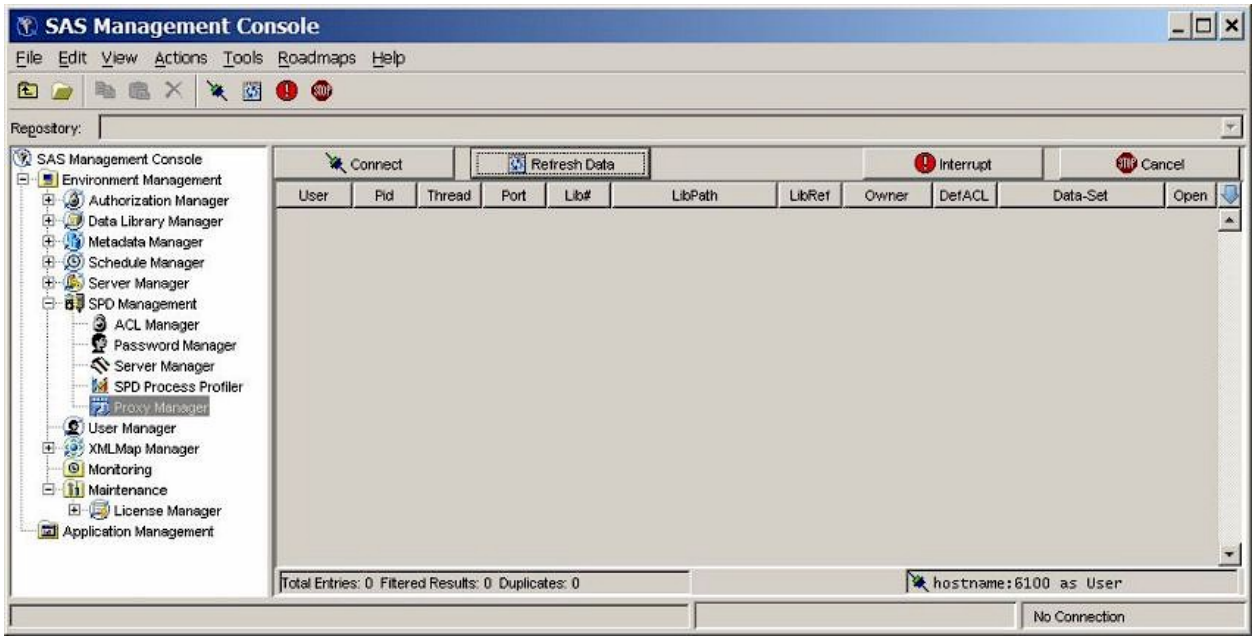
The host performance profile information is automatically updated whenever the SPD Server Performance Server performs another capture. The frequency of the SPD Server Performance Server's information capture frequency is specified by the `-c` option when the SPD Server Performance Server is started. More information on Performance Server startup settings is available in the SPD Server Help chapter on Setting Up the SAS Scalable Performance Data Server Performance Server.

If the SAS Management Console user also connects to the same SPD Server Name Server via the ACL Manager, the Password Manager, or the Server Manager, the User information that is displayed will be 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

The SAS Management Console tree for SPD Server contains a folder called Proxy Manager. You use the Proxy Manager 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. The section Connecting to an SPD Server provides detailed instructions on connecting to an SPD Server host. After you connect to an SPD Server host, click **Refresh Data** in Proxy Manager to update the view on the table data.



You can filter, sort, reorder, and hide Proxy Manager table columns to display proxies of interest. You perform filtering and sorting by clicking on the column headings and selecting the appropriate choice from the menu.

Filtering can be specified for any number of the columns. Column filters offer three possible states:

- 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 located just above the vertical

scroll bar on the right side of the Proxy Manager.

Proxy Refresh

Click **Refresh Data** to update the table with the most current proxy data. Refreshing is necessary after an initial connection or after a proxy state has been manipulated. Because a proxy's state is dynamic, each refresh provides only an instantaneous snapshot of the proxies. Data status can change immediately after the data is refreshed, which is the nature of a dynamic client/server environment. If no users are currently logged onto the server, a window delivers the message.

Proxy Interrupt

Click **Interrupt** to interrupt a selected libref's proxies. The proxy's activity is halted when it next processes data from its socket. The frequency with which the proxy accesses its socket is unpredictable and can vary depending on many variables, but the proxy interrupt operation is normally the first method used to halt a proxy from accessing a given data set or domain.

Proxy Cancel

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. Clicking **Cancel** to stop a proxy from accessing a data set or a given domain is recommended when the interrupt operation is unacceptably delayed.

SPD Server and SAS Data Integration Studio

You can integrate the processing power of SPD Server with other SAS software tools, such as SAS Data Integration Studio. The same plug-in file that SPD Server uses to integrate with the SAS Management Console can be used to integrate SPD Server resources with the SAS Data Integration Studio user interface.

SAS Data Integration Studio is software that enables data warehouse specialists to create and manage metadata objects that define sources, targets, and the sequence of steps for the extraction, transformation, and loading of data into data marts or warehouses. SPD Server can be an excellent tool for managing the large tables of data associated with large data marts and warehouses.

To integrate SPD Server functionality into the SAS Data Integration Studio graphical user interface, copy the SPD Server Java plug-in file with the SAS Data Integration Studio **`plugins`** subdirectory.

The SPD Server Java plug-in file is located at:

```
SASROOT/spds44/spdssmc/sas.smc.SpdsMgr.jar
```

Note: ***`SASROOT`*** represents the path to the base directory of the SAS software installation on your client machine. **`Spds44`** represents the installed SPD Server software directory. The name of the installed SPD Server software directory varies according to the specific version and release of your SPD Server software. For example, the path to your SPD Server Java plug-in file might begin with **`SASROOT/spds44`**, **`SASROOT/spds44tsm1`**, or **`SASROOT/spds44tsm2`**, depending on if you have the original SPD Server 4.4 software, or the first or second maintenance release of the SPD Server 4.4 software.

Copy the SPD Server Java plug-in file to the SAS Data Integration Studio **`plugins`** directory:

```
SASROOT/SASETLStudio/9.1/plugins/sas.smc.SpdsMgr.jar
```

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

Contents

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 separately evaluated. 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; it only expedites the processing that is required to calculate them. 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 may vary if you compare the optimized query answer set with the query answer set 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. SPD Server administrators must provide adequate space for the

generation and storage of the intermediate results tables. 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 the SPD Server administrator specifies. 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 utilizes the SPD Server TEMP=YES option setting when accessing the LIBNAME domain for intermediate result tables. The TEMP=YES option creates a processing environment where multiple SPD Server users can concurrently create tables with no 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 both specify LIBNAME domains for SQL Query Rewrite Facility intermediate results storage. SPD Server administrators can configure use the TMPDOMAIN= parameter in the spdsserv.parm file to specify the SQL Query Rewrite Facility intermediate results storage domain:

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

where <*DomainName*> is the name of a LIBNAME domain that is defined in the SPD Server's associated libnames.parm file.

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 wanted to use the EMATMP LIBNAME domain for SQL Rewrite Facility intermediate results, the user would submit the following RESET command in his or her SQL job stream:

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

Setting 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 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 SQL Query Rewrite Facility intermediate results storage location does not affect TMP=YES environment setting that permits concurrent access to tables in the domain by multiple SPD Server users. This means that multiple SPD Server users

can specify and share remapped TMPDOMAIN= locations without 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. That means 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 will optimize 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 */
```

Using SAS Scalable Performance Data Server With Other Clients

Tables and Figures

This chapter describes using SPD Server to connect with ODBC, JDBC, htmSQL, and SQL C API clients.

Overview

Scalable Performance Data 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 PROC COPY, and, with the proper drivers installed on the network, allows queries on the tables from client machines that do not have SAS software.

There are four possible options:

- **ODBC**: Open Database Connectivity - This is an interface standard that provides a common interface for accessing databases. Many software packages running in a Windows environment are compliant with this standard and can access data created by other software. This is a good choice if you have client machines running Windows applications, such as Microsoft Excel or Microsoft Access.

- **JDBC**: Java Database Connectivity - This option allows users with browsers to log on to a Web page and make a query. The results of the request are formatted and returned to a Web page. This makes information available across a wide range of client platforms because all you need, after installing the JDBC driver on SPD Server, is a Web page with some Java code, and a client machine with a Java-enabled browser.

- **htmSQL**: HyperText Markup Structured Query Language - This option allows users with browsers to log on to a Web page and make a query. The results of the request are formatted and returned to a Web page. This makes information available across a wide range of client platforms. Why? After installing the htmSQL driver in SPD Server, all you need is an htmSQL Web page and a client machine with a browser.

- **SQL C API**: This option allows access to SPD Server tables from SQL statements 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 is designed to support multi-threaded applications and is available on all supported SPD Server platforms.

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

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

Read this section if you do not have Base SAS software on the network client, but you want to access SPD Server tables on the network, using an ODBC compliant program, such as Microsoft Word, Query, Excel, or Access, and you have SPD Server tables available for use, somewhere on the network, or Scalable Performance Data Servers and SPD SNET servers running, or client machines in a Windows environment.

Why Use ODBC?

You have SPD Server tables available on your network, and one or more of the following may be 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.

Installing OBDC Drivers on the Server

- Instructions for installing the OBDC driver are included in the download package.

Configuring ODBC on the Client

1. Configure an ODBC data source.
2. Make your query using a Windows program.

Figure 6.1: Configure ODBC to Connect SPD Server Client to SPD Server Host

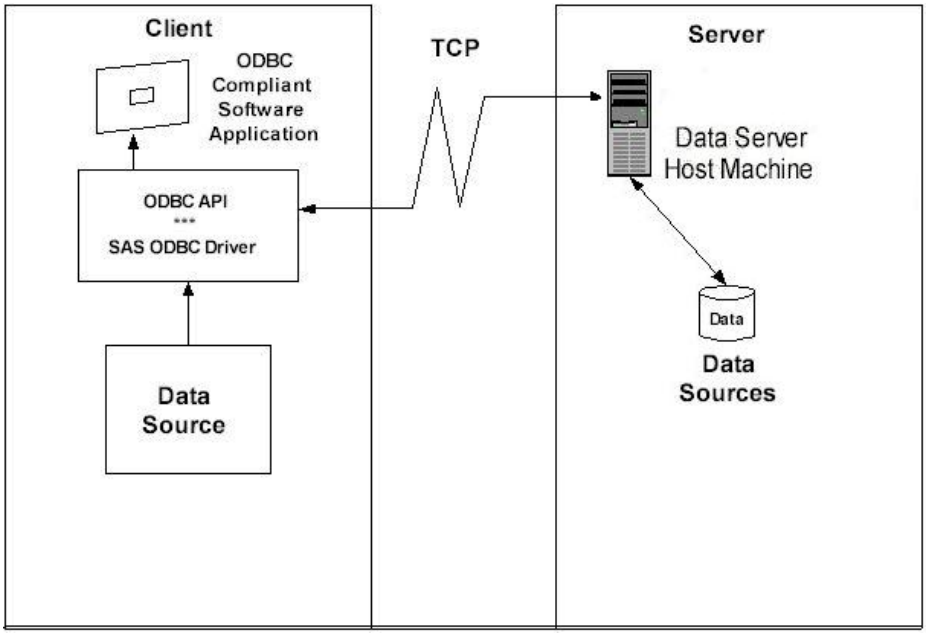
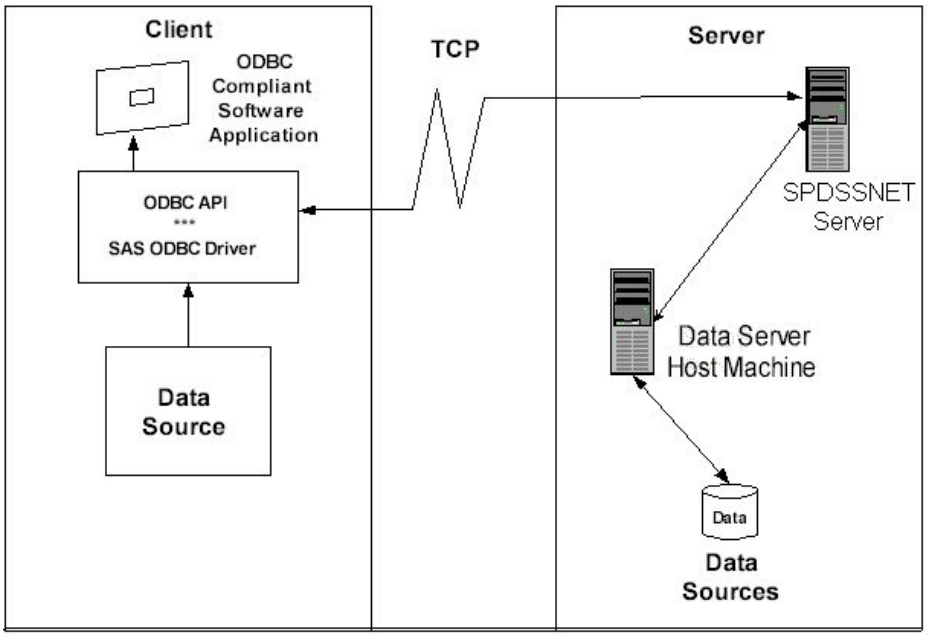


Figure 6.2: Configure ODBC to Connect SPD Server Client to SPD SNET Server



Preparing your Client Machine for ODBC Installation

Before you create OBDC data sources driver, you'll need the following information from your network administrator:

- a Username 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 you wish to assign to the ODBC connection.

Two Types of ODBC Connections

With SPD Server software you can connect directly to an SPD Server without going through the SPD SNET server. Although connecting directly is the preferred method, connections via the SPD SNET server are still supported.

Note that connections via the SPD SNET server are not supported in the SAS 9 ODBC Driver software. If you intend to connect via the SPD SNET Server you must install the SAS 8 ODBC Driver.

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 which are configured through the Libraries tab of the SAS ODBC Driver Configuration window.

ODBC Connections via the SPD SNET server must have an **odbc.parm** file configured on the SPD SNET Server machine.

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

Once the SAS ODBC driver is installed, you will need to configure your ODBC data source. When you open the ODBC manager, you'll get a display screen that allows you to enter information that points the OBDC driver to the data on the SPD Server.

1. From the Windows Start button, select

   Start ➧ Settings ➧ Control Panel

2. Locate the ODBC Data Sources icon and open the Microsoft ODBC Data Source Administrator . The exact location of this program depends on your version of Windows.

3. Select the Add button, then select the SAS ODBC driver.

4. Enter a data source name (and description if desired.)
5. Select the Servers panel and type in your two-part server name.
6. Click on the Configure box. The TCP Options window appears:
    - **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 (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 (SPD Server primary LIBNAME domain) on the SPD Server host to which you will connect.
    - **Connection Options**: Enter the Connection Options as follows:
        - `DBQ='SPD Server primary LIBNAME domain'`, this is the SPD Server LIBNAME domain
        - `HOST='nameserver node name'`, this is the location of the host computer
        - `SERV='nameserver port number'`, this is the port number of the SPD Server name server running on the HOST.
        - Any other SPD Server LIBNAME options. For more information, see the User's Guide section on LIBNAME Options.
9. Click OK, then click Add, and select the Libraries panel.
10. Enter the DBQ name of a secondary LIBNAME domain in both the Name and Host File text fields.
11. Enter “spdseng” in the Engine text field.
12. Use SQL Pass-Through syntax rules for LIBREF statements when you enter a value in the Options text field.

Configuring an ODBC Data Source for SPD SNET

Once the SAS ODBC driver is installed, you will need to configure your ODBC data source. When you open the ODBC manager, you'll get a display screen that allows you to enter information that points the OBDC driver to the data on the SPD Server.

1. From the Windows Start button, select

    Start ➧ Settings ➧ Control Panel

2. Click on the ODBC icon and select the Add button.
3. Select the SAS ODBC driver.
4. Enter a data source name (and description if desired).
5. Select the Servers panel and type in the two-part server name. The second part of the server name should match the entry in the services file. In the example that follows that shows you how to edit the services file, the server name is **spdssnet**.
6. Click on the Configure box. The TCP Options window appears with four input fields that you fill:
    - **Server Address**: Enter the network address of the machine on which the SPD SNET server is running.
    - **Server User Name:** Enter the user name as configured for a DBQ (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 (SPD Server primary LIBNAME domain) on the SPD Server host to which you will connect.
    - **Connection Options:** Enter the connection options as follows:
        - `DBQ='SPD Server primary LIBNAME domain':` this is the SPD Server LIBNAME domain.
        - `HOST='nameserver node name':` this is the location of the host computer.
        - `SERV='nameserver port number':` this is the port number of the SPD Server name server running on the HOST.
8. Click OK, and then click Add.

Editing the Services File on Your Machine - ODBC Details

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

1. Find the Services file on your Windows machine. In 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 may be sorted in port number order. Find the closest port number to the SPD Server port number, which you obtained from the network administrator (see "Preparing for Installation"). This is where you insert the new information.
4. Add an entry to the Services file, on its own line, in proper numeric order, using the following syntax:

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

Table 6.1: *How to Add Service Name and Port Number to the Services File*

Remember: The service name, **spdssnet** must match the server name that you used in step 6 of Configuring an ODBC Data Source for SPD SNET. The port number must match the port number on which the SPD SNET server is running.

Creating 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

Read this information if you do not have Base SAS software on the network client, but you want to use the power of the Java programming language to query SPD Server tables from any client on the network that has a browser. 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 in order to use JDBC to access SPD Server tables.

- Why Would I Want to Use JDBC?
- How Is JDBC Set Up on the Server?
- How Is JDBC Set Up on the Client?
- How Do I Use JDBC to Make a Query?
- JDBC Code Examples and Tips
- Limitations of Using JDBC with SPD Server

Why Would I Want to Use JDBC?

You might want to use JDBC if you have SPD Server tables available on your network and one or more of the following is 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. One thing they might have in common is browser access to your intranet.
- The audience for the information understands Web browsing and wants point-and-click access to the information.
- You want to distribute the information over the World Wide Web.

- Your planned application requires the power of the Java programming language.

How Is JDBC Set Up on the Server?

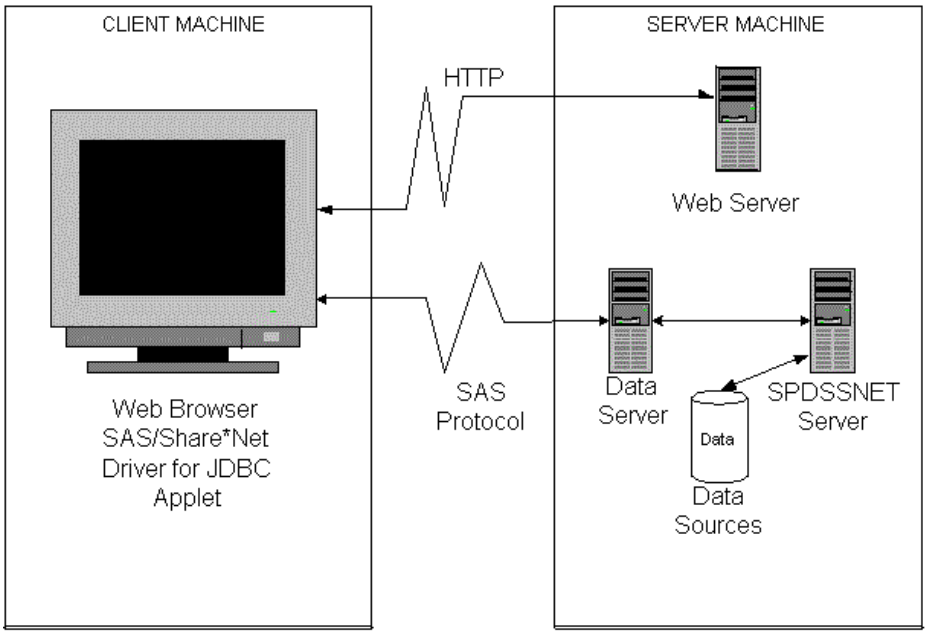
- JDBC is usually set up on the server at the time the SPD Server is installed. The process is covered in the SPD Server installation manual.

How Is JDBC Set Up on the Client?

The client needs a browser set up to accept Java applets, such as

- Netscape Navigator, Release 3.0 or later
- Microsoft Internet Explorer, Release 3.02 or later.

Figure 6.3: **JDBC Set Up on an SPD Server Client**



How Do I Use JDBC to Make a Query?

1. Log on to the World Wide Web and enter the URL for the Web page that contains the JDBC code.
2. Click on the desired information.
3. JDBC handles the request, formats the information, and returns the result to the Web page.

JDBC Code Examples and Tips

The following lines must be a 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="spdsuser" 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 set up 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=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:

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

Line 5:

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

Line 6:

- "**shareRelease**" value="V9" is the version of the driver you are using. This must not be altered.

Line 7:

- Sets the foreign database property on the JDBC driver. This means that the server is not SAS and JDBC should not create a DataBaseMetaData object. See the examples below for how to get around this.

Limitations of Using JDBC with the SPD Server

JDBC Used with SAS Versus JDBC Used with the SPD Server

SPD Server is treated as a foreign database. SPD Server clients can't query the JDBC metadata class for available tables and other metadata. Users must write their own queries to do this.

Example JDBC Query for Getting a List of Tables

(JDBC Used with the 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

(Your data file)

```
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

Read this section if you do not have Base SAS software on the network client, but you want to use the point-and-click convenience of a Web page to query SPD Server tables from any browser-enabled client on the network. You must have SPD Server tables available for use, htmSQL loaded and configured on a UNIX or Windows operating system, and Scalable Performance Data Servers and SPD SNET servers running.

Why Would I Want to Use htmSQL?

You may want to use htmSQL if you have SPD Server tables available on your network and one or more of the following is 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. One thing they might have in common is browser access to your intranet.
- The audience for the information understands Web browsing and 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 World Wide Web.
- Your developers are familiar with SQL and HTML.

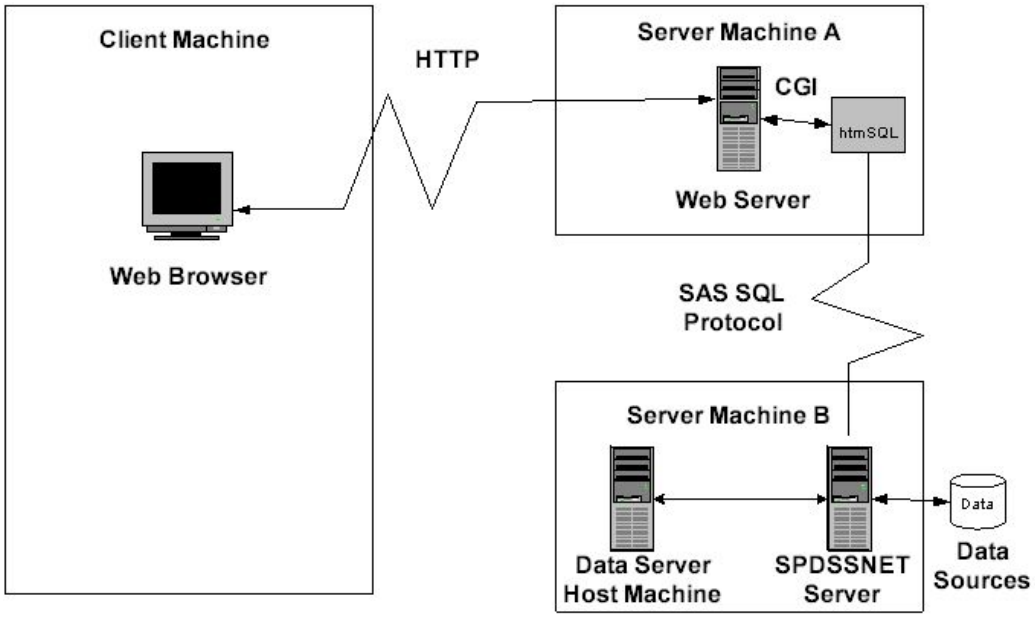
How Is htmSQL Set Up on the Server?

- 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 and 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.

How Is htmSQL Set Up on the Client?

HtmSQL requires nothing more than a browser on the network or Web client.

Figure 6.4: **htmSQL Configured on an SPD Server Client**



How Do I Use htmSQL to Make a Query?

1. Log on to the World Wide Web and enter the URL for the Web page that contains the htmSQL code.
2. Click on the desired information.
3. htmSQL handles the request, formats the information, and returns the result to the Web page.

Examples of Setting 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

```
http://support.sas.com/rnd/web/intrnet/htmSQL/index.html
```

Using 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.

Why Would I Want to Use SQL C API?

You have SPD Server tables available on your network and one or more of the following may be true:

- You do not have Base SAS software on the network client to process the data sets.
- You wish 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++.

The chapter "SPD Server SQL Access Library API Reference" in the SPD Server User's Guide contains additional information on SQL C API.

Configuring Disk Storage for SAS Scalable Performance Data Server

---

Configuring Disk Storage for SPD Server

Introduction

This section discusses management of very large SPD Server data stores. Read this topic if you manage large SPD Server data files with sizes that consume gigabytes of disk storage.

How you configure this disk storage, either for many SPD Server users or a few large-scale users, is important. To do the job effectively, you need to know which files the SPD Server creates, when they are created, and their relative size.

The Relative Size of SPD Server File Types

SPD Server uses four types of component files. Component files are physical file entities that SPD Server uses to track table and index metadata. Component files, when combined, form a logical structure that SPD Server understands and interprets as a single table. The relative sizes of the four types of SPD Server component files are:

File Type	File extension	Relative size	Number of component files
Table Metadata	.mdf	Very small	1
Table Data	.dpf	Large	1 to many
Hybrid Segmented Index(es)	.idx	Medium to Large	0 or more
Hybrid Global Index(es)	.hbx	Medium to Large	0 or more

Which Component Files Are Created and When?

How Many Files Are Created for SPD Server Tables?

Minimally, an SPD Server table consists of two files, the metadata and data files (an **.mdf** and **.dpf** pair). The size of the data component depends on two factors: the size of a table column and the number of columns.

Potentially, the .dpf data component can be many gigabytes in size because SPD Server is not constrained by an operating system file system size limit. (Many readers are familiar with the 2-gigabyte limit imposed by some UNIX systems.)

How Many Files Are Created for Indexes on SPD Server Tables?

The SPD Server hybrid index uses two index file types: **.hbx** and **.idx** files. The .hbx file maintains a global view of the index that contains a single entry for each key in the index. The .idx file maintains a segmented view of the index that includes a list of the segments each key is in and for each segment a bitmap used to determine the per-segment observations for that key.

The size of the .hbx file depends on the cardinality of the index keys. The higher the cardinality, the larger the file. The size of the .idx file is much more difficult to determine because it is based on the distribution of the data for the index keys. A key that is in many segments will require a larger segment list and therefore a larger number of per-segment bitmaps than a key that is in a smaller number of segments.

Because of this, the best case scenario for the size of .idx file is achieved if the table is sorted by the indexed columns to minimize the number of segments the key is in. The worst case scenario for the size of the .idx file is the case where keys are in a large number of segments with a low cardinality of rows for each segment.

Configuring LIBNAME Domain Disk Space

You, the acting or designated system administrator, define the primary file system for each LIBNAME domain that you must support for your SAS user base. Optionally, you can define initial and overflow stores for the data component (.dpf) and 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 with the PATHNAME= statement in the SPD Server's LIBNAME parameter file, libnames.parm. For example:

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

An example of a libnames.parm file entry for a UNIX system:

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

An example of a libnames.parm file entry for Windows:

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

When users create new tables in a LIBNAME domain, there is a hidden system detail to consider. The metadata component (.mdf) must start in the primary file system. This detail is very important. If the primary file system fills up, you cannot create more new tables until you free up some disk space in that file system.

Example 1 stores all the component files: metadata, data and index data in the primary file system. This can present a problem for large tables. Large tables can quickly fill up the primary file system, preventing further table

creation.

We recommend storing 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

```
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')
     indexpath=('/disk4/userindexes' '/disk5/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 reason: the primary file system is reserved for only very small metadata files. The larger data and index files will be stored in the other file systems specified with the DATAPATH= and INDEXPATH= options in the LIBNAME parameter file.

What if the data and index files do eventually fill up their file systems? The solution is simple: you must terminate SPD Server. Then, as shown in Example 3, invoke it again, adding more file systems to the path option that has exhausted its available space:

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

```
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

libnames.parm may now contain

```
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 might try to expand storage for the ".mdf" components by adding the METAPATH= specification to your ROPTIONS= value in your LIBNAME parameter file. Unfortunately, this will not solve your problem. Remember the SPD Server restriction mentioned earlier: all ".mdf" components must have their initial partition file created in the primary file system (the directory which was first specified by the PATHNAME= option for the LIBNAME domain).

To solve your problem, your only recourse in this situation is to create a new LIBNAME domain.

Recommended: Use ROPTIONS=

Here is why you should use ROPTIONS= instead of OPTIONS= in your libnames.parm file.

ROPTIONS= specifications override any corresponding options that your SAS users include in their programs. Example 3 demonstrates explicit control of disk usage by the system administrator. In the example, even if SAS users specify file systems using LIBNAME 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 (the administrator) 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.

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

Setting Up SAS Scalable Performance Data Server Parameter Files

---

Setting Up SPD Server Parameter Files

Introduction

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

Syntax for the -Parmfile Option

```
-parmfile file-spec
```

where *file-spec* is an explicit file path for SPD Server's parameter file. The spdsserv.parm file is **required** because it maintains options that control SPD Server's processing behavior and/or 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 spdsserv.parm Options

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

where *value* is option dependant. 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 and can support an "m" or "M" suffix to specify megabytes, or a "g" or "G" suffix to specify gigabytes.

The following examples of SPD Server parameter files (see *InstallDir*/samples/spsdsserv.parm) contain the recommended default settings for SPD Server. *InstallDir/* is a placeholder which represents the path to the root directory of your SPD Server installation.

Spdsserv.parm Options

BINBUFSIZE=

Controls the amount of memory that is used for each sort bin merge buffer. Each bin is equal to the value specified for SORTSIZE. When the sort operation overflows a single sort bin, SPD Server writes the bins to disk and then merges the bins to produce the final, sorted run. This parameter sets the size of the I/O buffer that must be used to read each bin.

Note: If you specify a value smaller than the record length of the sort bin, the software creates a bin buffer large enough to hold one record.

FMTDOMAIN=

Specifies a user-defined format. The FMTDOMAIN= option specifies the domain where the user-defined format is stored. To utilize user-defined formats, you must create a domain called "FORMATS". FMTDOMAIN= is used in conjunction with FMTNAMENODE= and FMTNAMEPORT=.

Usage:

```
FMTDOMAIN=FORMATS ;
```

FMTNAMENODE=

Specifies a user-defined format. The FMTNAMENODE= option specifies the server on which the user-defined formats are stored. FMTNAMENODE= is used in conjunction with FMTDOMAIN= and FMTNAMEPORT=.

Usage:

```
FMTNAMENODE=d8488 ;
```

FMTNAMEPORT=

Specifies a user-defined format. The FMTNAMEPORT= option specifies the port number of the server on which the user- defined formats are stored. FMTNAMEPORT= is used in conjunction with FMTDOMAIN= and FMTNAMENODE=.

Usage:

```
FMTNAMEPORT=5200 ;
```

GRPBYROWCACHE=

Specifies the maximum amount of memory threads use during parallel group aggregations. Parallel group SELECT uses multiple threads up to the MAXWHTHREADS= limit to perform partial group aggregations. The threads evenly share the memory specified using GRPBYROWCACHE to cache groups in memory; each thread receives 1/ MAXWHTHREADS= of the cache.

Once a thread accumulates enough distinct groups to fill its cache, the groups are flushed to secondary bins. At the completion of the parallel groupby, the partial group aggregations in memory and secondary storage are merged to assemble the final results. If omitted, the default is a 2 megabyte cache per thread. Increasing the default value can result in improved aggregation performance in cases with large numbers of groups. The trade-off is the potential of allocating more memory than is needed for caching, taking away that potential from the processing load.

IDLE_TIMEOUT=

Specifies the interval of idle time permitted before the SPD Server client process automatically

terminates the client connection. When the IDLE_TIMEOUT= is specified as a value greater than 0, the option is enabled. If the value is less than or equal to 0, SPD Server disables idle timeouts. The default setting is 0.

Usage:

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

INDEX_MAXMEMORY=

Affects read operations on SPD Server tables. The value specified restricts the amount of memory to allocate each open index.

INDEX_SORTSIZE=

Controls the amount of memory to be allocated for asynchronous (parallel) sort index creation or appends. This specified value is divided by the number of indexes, n, to be created or appended in parallel; each receives 1/nth of the memory allocation.

LDAPSERVER=

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

Usage:

```
LDAPSERVER= <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 is "LOCAL_HOST" or port 389.

Usage:

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

LDAPBINDMETH=

a string that indicates the LDAP authentication security level. The default value for LDAPBINDMETH= is ANONYMOUS. The ANONYMOUS setting is not recommended for use secure environments. The following are valid LDAPBINDMETH= values:

- LDAP_AUTH_SIMPLE — The SPD Server user password is sent to the LDAP Server in cleartext.
- LDAP_AUTH_SASL — SPD Server user authentication is performed via the SASL using the DIGEST-MD5 mechanism.
- LDAP_AUTH_NEGOTIATE (Windows platforms only) — The most appropriate authentication method is chosen from a list of available services on both the SPD Server and LDAP Server machines. Note that LDAP_AUTH_NEGOTIATE is not guaranteed to be secure.

Usage:

```
LDAPBINDMETH= <LDAP BIND METHOD-string>;
```

LDAPBINDDN=

The value for the LDAPBINDDN parameter is a string that is composed of Relative Distinguished Names, or RDNs. The SPD Server Administrator can obtain RDN strings from the LDAP Server Administrator when the SPD Server is being configured to use LDAP authentication. The LDAPBINDDN= option specifies the RDN, or the location in the LDAP Server database where the information for the connecting client is stored.

Usage:

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

MINPORTNO=/MAXPORTNO=

Specifies a range of port numbers that can be used by the SPD Server user proxy processes to communicate with the client. You must set both of the MINPORTNO= and MAXPORTNO= parameters. This option is provided to support the use of SPD Server ports through an Internet firewall, in order to limit the range of ports that will be used by the server. If MINPORTNO= and MAXPORTNO= are not defined, then the SPD Server user proxy processes use any port that is available to communicate with the client..

MAXGENNUM=

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

MAXSEGRATIO=

Controls segment candidate pre-evaluation for WHERE-clause predicates with a hybrid index. The WHERE planner pre-evaluates to determine the candidate segments for the predicate. Only the candidate segments are actually searched to resolve the WHERE-clause.

Some queries can benefit from cutting off the pre-evaluation based on the ratio of the number of segments containing candidates to the total number of segments in the file. If the percentage of possible segments exceeds this threshold, the pre-evaluation is skipped and all segments are searched to resolve the WHERE-clause. If omitted, the default value is 75 percent.

MAXWHTHREADS=

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

MINPARTSIZE=

Ensures that large SPD Server tables cannot be created with arbitrarily small partition size. Large SPD Server tables with small partition sizes create an excessive number of physical files, which increases clutter and penalizes I/O performance. The default setting for MINPARTSIZE= is 16 MB. The most common settings for the MINPARTSIZE parameter range from 128 MB to 256 MB.

[NO]COREFILE

Controls whether the LIBNAME proxy creates a core file in the event of an unexpected process trap. The default is NOCOREFILE.

[NO]LDAP

Turns SPD Server LDAP user authentication on or off. If the LDAP parameter is present during SPD Server start-up, then the SPD Server host creates a context for LDAP authentication.

Usage:

```
LDAP ;
```

and

```
NOLDAP ;
```

[NO]LIBACLINHERIT

You use the LIBACLINHERIT option to create a LIBNAME ACL on a LIBNAME domain that you specify. LIBNAME ACL permissions grant 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 the resource is owned by the same ACL group as the accessor, the accessor gets the resource's GROUP permissions.
3. If the resource is in a domain that is named in a LIBACLINHERIT statement, the accessor gets permission to *all* resources in the LIBNAME domain.
4. If the resource is owned by a different ACL group than the accessor, the accessor gets the resource's UNIVERSAL permissions.

LIBACLINHERIT can be specified in two places. LIBACLINHERIT can be enabled in the global spdsserv.parm file, or LIBACLINHERIT can also be specified in the libnames.parm file, or in a LIBNAME statement.

If the LIBACLINHERIT parameter is not specified, it is turned off, and the ACL precedence of permissions is as shown above, except step 3 is omitted.

Usage in spdsserv.parm:

```
LIBACLINHERIT ;
```

and

```
NOLIBACLINHERIT ;
```

Usage in libnames.parm or LIBNAME statement:

```
LIBACLINHERIT=YES
```

and

```
LIBACLINHERIT=NO
```

[NO]NLSTRANSCODE

Enables or suppresses the experimental server-side SPD Server NLS processing. More detailed information is available in the chapter on SPD Server Experimental NLS Support.
The default setting for NLSTRANSCODE is NONLSTRANSCODE if the option is not present in the spdsserv.parm file. The default spdsserv.parm file for SPD Server is shipped without the NLSTRANSCODE option in it. Users must take explicit action to activate server-side transcoding in SPD Server 4.4.

Usage:

```
NLSTRANSCODE ;
```

```
/* Enables experimental SPD Server */
/* server-side NLS processing      */

NONLSTRANSCODE ;
/* Disables experimental SPD Server */
/* server-side NLS processing       */
```

When NONLSTRANSCODE is in effect, 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 4.4 (in conjunction with SAS 9) performs normal server-side processing of tables, ignoring the CEI of the table. SAS 9.1.3, however, will read the CEI value of the table and apply a transcoding to any pertinent character variables in the rows returned from SPD Server.

When NLSTRANSCODE is in effect, SPD Server reads the table's character set encoding (CEI) value and looks at the CEI value of the associated SAS 9.1.3 session. SPD Server performs no transcoding if they are the same. If the CEI values are different, SPD Server places restrictions on the kinds of WHERE-clause predicates permitted in indexed lookups, and SPD Server ensures that data is returned to SAS 9.1.3 using the same encoding that the SAS 9.1.3 session uses.

[NO]WHEREAUDIT

Enable or disable audit trails for WHERE clauses that are executed by SPD Server. The SPD Server administrator enables WHEREAUDIT in the spdsserv.parm file. Unless specified in the spdsserv.parm file, the default setting for WHEREAUDIT is disabled audit logging.

The **spdslog** process performs message logging functions and the **spdsaud** process performs audit logging functions. If you use WHEREAUDIT both the log and audit file will contain WHERE statement information.

Usage:

```
WHEREAUDIT ;
/* Enables WHERE clause audits  */
/* for all SAS users            */

NOWHEREAUDIT ;
/* Disables WHERE clause audits */
/* for all SAS users(default)   */
```

You enable the audit trail logger facility (the spdsaud process) by using the *-auditfile* option with the spdsserv command. Submit the following to the spdsserv command line if you have not done so:

```
-auditfile fileSpec
```

The *-auditfile* option enables audit logging for the server and automatic audit log file creation by the audit process. The fileSpec is used to generate the complete audit file path. For example, if you specified fileSpec as /audit/spds, the generated name would be:

```
/audit/spds_mmddyyyy_yyyy.spdsaudit
```

where mmddyyyy is taken from the system date when the log file is created.

[NO]WHERECOSTING

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

RANDOMPLACEDPF

Invokes the 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 sacrificing disk space.

RANIOBUFMIN=

Specifies the minimum random I/O buffer size. The value specified becomes

- the minimum I/O buffer size used by the proxy when it performs random I/O and table requests
- the maximum value for the IOBUFSIZE LIBNAME option.

SEQIOBUFMIN=

Specifies the minimum sequential I/O buffer size. The value specified becomes

- the minimum I/O buffer size used by the proxy when it performs sequential I/O, table requests
- the maximum value for the IOBUFSIZE LIBNAME option.

SORTSIZE=*(valid memory size)*

Controls the amount of memory to allocate for sort operations. A larger value can increase the paging activity for a file and degrade performance.

SQLOPTS=

You specify SQLOPTS on an SQL RESET command to override SQL default options for each SQL connect. If no SQLOPTS= command is specified, the compiled defaults apply. See the SQL RESET command for possible RESET options you can set.

Usage:

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

TMPDOMAIN=

Specifies an SPD Server domain that is defined in the libnames.parm file, which the query rewrite facility uses to store intermediate tables.

Usage:

```
...
```

```
libname=qrw pathname=/IDX1/spdsmgr/spds44_sasdqh/qrw ;
...
TMPDOMAIN=qrw ;
```

WORKPATH=

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

```
"('DirPath1' 'DirPath2' ...)"
```

SPD Server Parameter File Configurations for LDAP

- LDAP Server Running on SPD Server Host
- LDAP Server Running on SPD Server Host Using Port Other Than LOCAL_HOST
- LDAP Server and SPD Server Host Running On Different Machines
- SPD Server UserIDs and Passwords Not at Default Location in LDAP Database
- SPD Server UserIDs and Passwords Not at Default Location in LDAP Database and LDAP Server Using TCPIP_PORT

LDAP Server Running on SPD Server Host

For this scenario, 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, running at port LOCAL_HOST, on the SPD Server host.

LDAP Server Running on SPD Server Host Using Port Other Than LOCAL_HOST.

For this scenario, assume that all other LDAP settings use the default configuration, and that you want to perform LDAP user authentication where the LDAP server is using a port number that is different from the port assigned to LOCAL_HOST. To run an LDAP server on the SPD Server host using a port assignment other than LOCAL_HOST, add the LDAP option and the LDAPPORT= port specification to your SPD Server parameter file.

LDAP Server and SPD Server Host Running On Different Machines

For this scenario, assume that you want to perform LDAP user authentication, but the LDAP server and the SPD Server hosts reside 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. This results in LDAP user authentication where the LDAP server is running at port LOCAL_HOST on "*host.domain.company.com*".

The default SPD Server LDAP authentication mechanism is ANONYMOUS. ANONYMOUS LDAP authentication is not secure. When the SPD and LDAP Server hosts are different, use the SASL DIGEST-MD5 authentication mechanism for secure authentication. To use SASL DIGEST-MD5 secure authentication, add the statement LDAPBINDMETH=LDAP_AUTH_SASL to your parameter file.

SPD Server UserIDs and Passwords Not at Default Location in LDAP Database

For this scenario, assume that you want to perform LDAP user authentication, but the SPD Server UserIDs and Passwords are not in their default locations in the LDAP database. Assume that all other LDAP settings use default configurations.

First, add the LDAP option and the LDAPBINDDN= specification, where the LDAPBINDDN= property setting is: "*ou=people, dc=domain, dc=company, dc=com*" ;. This arrrangement results in LDAP user authentication, with the LDAP server running at port LOCAL_HOST on the SPD Server host machine. The LDAP server will look for SPD Server users at the location that corresponding to "*ou=people, dc=domain, dc=company, dc=com*" in its database.

SPD Server UserIDs and Passwords Not at Default Location in LDAP Database and LDAP Server Using TCPIP_PORT

For this scenario, assume that you want to perform LDAP user authentication, the SPD Server UserIDs 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.

First, add the LDAP option and set the LDAPPORT= port specification to TCPIP_PORT in your SPD Server parameter file. Then, add the LDAPBINDDN= specification, where the LDAPBINDDN= property setting is: "*ou=people, dc=domain, dc=company, dc=com*" to the SPD Server parameter file.

User authentication will be performed using the LDAP Server, running at port TCPIP_PORT on the SPD Server host. The LDAP server will look for SPD Server users at a location that corresponds to "*ou=people, dc=domain, dc=company, dc=com*" in its database.

Setting Up SAS Scalable Performance Data Server Libname Parameter Files

---

Setting Up SPD Server Parameter Files

Introduction

On start-up, the SPD Server server 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 where 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 also will contain the data tables, index tables, and intermediate tables that the domain creates. SPD Server administrators and users can use domain path options 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.

Examples of simple **libnames.parm** file domain declarations follow:

```
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 example SAS code below connects to the first domain that was declared previously:

```
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.

SPD Server domain optional path parameters are specified 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 via SAS code.

Use the roption (reserved option) specification to ensure that the domain options that you declare in **libnames.parm** cannot be modified by subsequent libref statements submitted to SPD Server via SAS code.

All options 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.

The syntax you use to specify optional path parameters in the libnames.parm file is identical for 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 are LIBNAME domain path options for SPD Server:

- DATAPATH=
- INDEXPATH=
- WORKPATH=
- METAPATH=

DATAPATH=

Specifies a list of paths that will contain SPD Server data tables associated with the declared domain. DATAPATH= can be specified as an option or as an roption.

Usage:

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

INDEXPATH=

Specifies a list of paths that will contain SPD Server index tables associated with the declared domain. INDEXPATH= can be specified as an option or as an roption.

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 associated with the declared domain. WORKPATH= can be specified as an option or as an roption.

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 becomes filled, *new* tables cannot be added to the domain. The primary path should be located 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.

METAPATH= can be specified as an option or as an roption.

Usage:

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

Consistency in Nomenclature

It is a suggested practice (but not a requirement) to match or closely match the LIBNAME, path name, and other optional path names for consistency. The following examples illustrate a domain declaration that is easy to follow, and a domain declaration that requires more concentration to follow.

Example of intuitive names in a libnames.parm file:

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

In the previous example, the declared domain name, path name, data path name, and index path name are all "spds123".

Example of non-intuitive names in a libnames.parm file:

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

The non-intuitive names example uses different names for the declared domain name, path name, data path name, and index path name. The structure is technically valid, but is also unnecessarily complex.

In summary, it is a good idea to use the same name that is declared as a LIBNAME

domain as the destination directory name for path name, data path, and index path specifications.

The directories that are specified in domain path name, data path, and index path statements should correspond to one and only one domain. In the intuitive names example, the path name, data path and index path specifications point to separate, unique paths that end with the directory name, "spds123", which correspond to the domain "spds123". If a domain "spds456" exists, it should have its own unique domain path name, data path, and index path specifications, and share no specified path with "spds123" or 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=
- LIBACLINHERIT=
- DYNLOCK=

OWNER=

Specifies the owner of a domain. The SPD Server owner controls the resources of the domain, and can 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.

Usage:

OWNER=*owner-id*

LIBACLINHERIT=

The LIBACLINHERIT parameter file option controls the ACL precedence of permission checks. Turning on LIBACLINHERIT creates a LIBNAME ACL on the specified LIBNAME domain and grants users rights to all resources within the domain. When a LIBNAME ACL is specified for a domain, the following is the ACL

precedence of permission checks:

1. If user-specific permissions are defined, the accessor gets these permissions.

2. The accessor gets GROUP permissions to all resources owned by the ACL group.

3. LIBNAME ACL permissions are used for domains where LIBACLINHERIT is turned on.

4. If the resource is owned by an ACL group that the accessor does not belong to, the accessor gets the resource's UNIVERSAL permissions.

The following is an example of SAS code submitted to SPD Server using LIBACLINHERIT. The example begins by showing information in the **libnames.parm** file where domain names and paths are declared.

Contents of the **libnames.parm file**:

```
LIBNAME=libinher
PATHNAME=/IDX1/spdsmgr/spds44test/libinher
LIBACLINHERIT=YES
OWNER=admin ;

LIBNAME=noinher
PATHNAME=/IDX1/spdsmgr/spds44test/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 ;
```

DYNLOCK=

Overview of Dynamic Locking

Dynamic locking is an SPD Server feature that allows multiple users concurrent access to SPD Server tables. The tables can be concurrently accessed by multiple users for reading and writing, while maintaining the integrity of the table contents. When dynamic locking is enabled, users can insert, append, delete, and update the contents of an SPD Server table while doing concurrent reads on the table.

Dynamic locking is enabled or disabled at the domain level. All tables stored within the domain are subject to the enabled or disabled state of the dynamic locking feature. The DYNLOCK=

statement should be used in **libnames.parm** file domain declarations.

How is dynamic locking 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 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 hitting 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 a 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 a SPD Server record-level proxy, it all client connections are affected.

Dynamic locking may 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 is due to the more distributed processing and parallelism of that occurs when multiple SPD Server user proxies are utilized. The performance benefit depends on the opportunities for parallelism and should be measured on a case-by-case basis.

Dynamic Locking Details

In order 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 will automatically use dynamic locking. The SPD Server clients do not need to set any additional parameters to take advantage of the domain's dynamic locking benefits.

When SPD Server proxy processes receive concurrent update, append, insert, and delete commands, they 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 while an update is in progress. 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 in cases where the user requires SAS-style record-level integrity across multiple clients. Reading a record using dynamic locking does not guarantee that the record cannot change before a subsequent read or update is executed. If a true record-level lock is needed by a client, then the record-level locking protocol should be used.

It is not possible to use record-level locking on a domain that has dynamic locking enabled. Dynamic locking is also not supported for tables that use dynamic clusters.

Usage:

To enable dynamic locking, use the DYNLOCK= statement in the **libnames.parm** file domain declarations. If the DYNLOCK= option is not specified, the default SPD Server setting for DYNLOCK is NO.

DYNLOCK=<*YES/NO*>

Organizing Domains for Scalability

SPD Server performance is based on scalable I/O. You can use the **libnames.parm** file to optimize the way SPD Server stores files in order to exploit scalable I/O. The domain path options section in this document provides instructions on how to specify named paths for the three data components of SPD Server tables (observation data tables, index data tables, metadata tables) as well as paths for temporary intermediate calculation tables. LIBNAME domain declaration statements can specify the system paths that are associated with each table space component, but the SPD Server administrator 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. SPD Server administrators should 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 a domain is declared in a LIBNAME statement, data tables are stored in the space defined in the PATHNAME= specification, unless the DATAPATH= option is specified. The PATHNAME= space is designed to contain metadata tables for a domain, but it can also contain data tables. As a domain's size and complexity increases, so do the benefits for organizing data tables into their own DATAPATH= space.

Organizing your data table space significantly impacts I/O scalability. The disk space allocated to data tables stores permanent warehouse tables that users will access. It is important for this disk space to 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 is likely to see the greatest proportion of read/write I/O.

Tables in the data table space are typically loaded or refreshed using batch processes during evenings or off-peak hours (such as weekends and holidays). Access to data table space is often restricted to read-only for all users except for the administrators who perform the load and refresh processes.

To ensure reliability, data table space is typically organized into RAID 1+0 or RAID-5 disk configurations. Very large warehouses should consider a RAID-5 configuration with a second storage array to mirror the data.

Index Table Space

When a domain is declared in a LIBNAME statement, index tables are stored in the space defined in the PATHNAME= specification, unless the INDEXPATH= option is specified. The PATHNAME= space is designed to contain metadata tables for a domain, but it can also contain index tables. As a domain's size and complexity increases, so do the benefits for 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 work spaces need for I/O performance. When a process is using an index, the read access pattern is very different from a parallel I/O pattern of data or multiple user patterns against data.

Index space is typically configured 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 will support some type of redundancy to ensure index space availability.

Metadata Table Space

When a domain is declared in a LIBNAME statement, metadata tables are stored in the space defined in the PATHNAME= specification. If the space configured in PATHNAME= fills, SPD Server stores overflow metadata for existing tables in the space defined in the optional METAPATH= specification, if it is declared. The PATHNAME= and METAPATH= spaces are specifically designed to 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, there will be approximately 10 gigabytes of metadata.

As a rule of thumb, when 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, additional metadata space should be added within the primary path of the server.

A table's metadata becomes larger when there are rows in the table that 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 deleted and on their distribution within the table.

Although the space required for the metadata is small, the setup and configuration of the disk space is very important. The disk space must be expandable, mirrored, and backed up.

Work File Space

SPD Server administrators use statements in the body of the **spdsserv.parm** file to reserve a space for intermediate calculations and temporary files. The work space that is configured in **spdsserv.parm** is shared by all SPD Server users.

Some SPD Server users have data needs that might be constrained by using the common intermediate calculation and file space reserved for all users. SPD Server administrators can use the **libnames.parm** file to create and reserve a work space that is specifically associated with a single domain and its approved users. This presents improvement opportunities for both security and performance. As a domain's size and complexity increases, so do the benefits for organizing temporary and intermediate tables into their own workspace defined by WORKPATH=.

Work space refers to the area on disk that SPD Server software uses to store required files when the available CPU memory cannot contain the entire set of calculations. During events like these, some utility files are written to disk. Work space is important to scalability. Tasks such as large sorts, index creation, parallel group-by operations, and SQL joins can require dedicated work space to store temporary utility files.

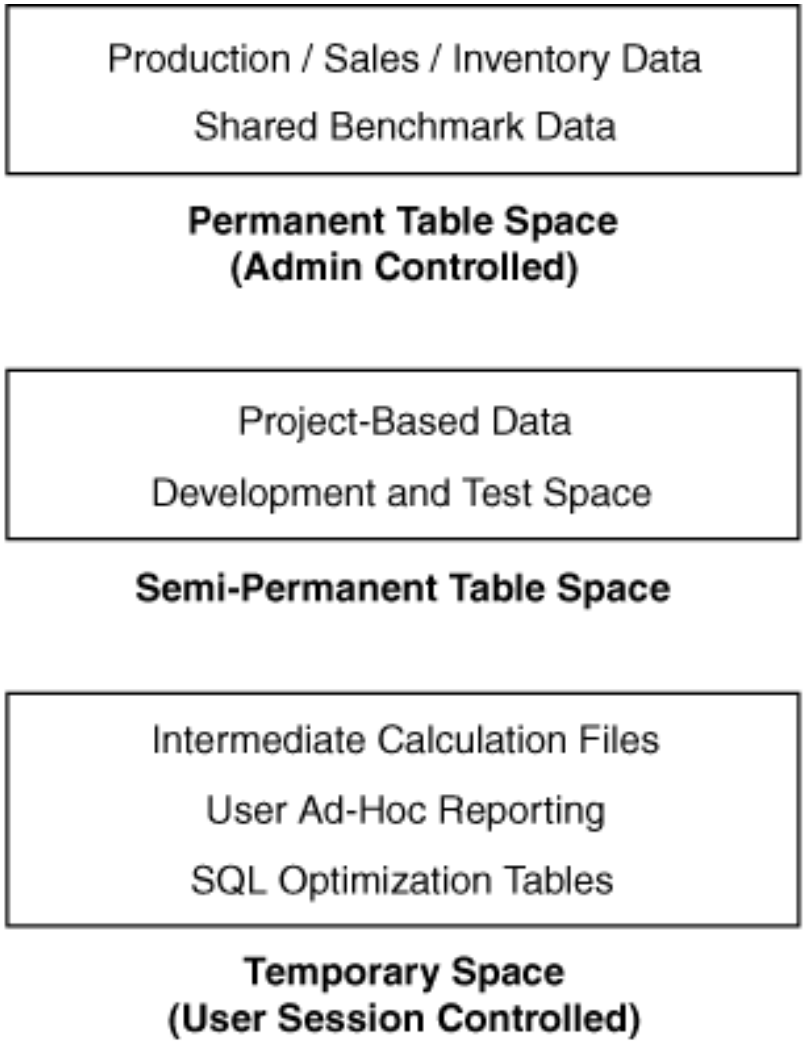
Work space is typically configured as part of a large striped file system that spans as many disks and I/O channels as possible. Workspace I/O can critically impact the performance behavior of an SPD Server host.

Work space on disk is typically a RAID 0 configuration or some hardware-redundant RAID design. RAID 0 configurations are risky to the extent that if the RAID 0 disk goes down, the system will also be affected and any process that was running at the time of failure will probably be affected.

Domains and Data Spaces

SPD Server is a tool that can be configured to meet different organizational data requirements. When an organization needs different types of SPD Server domain space, administrators 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 type (or types) of data space that the user needs. There are three basic types of domain space.

- Permanent Table Space
- Semi-Permanent Table Space
- Temporary Table Space



Permanent Table Space

In SPD Server, large production, inventory, and sales data storage areas work best using permanent table space. A rolling five-year sales data table by division and company is an example of an SPD Server structure that is best suited to permanently allocated space on the enterprise computers. Organizations can rely on large quantities of production, inventory, or sales data that is updated on a day-to-day, or even shift-to-shift basis. These data repositories require permanent, secure processing space that can only be accessed by a select group of key users. Allocating permanent space for the data ensures that sufficient disk space required for the combination and manipulation of 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, while not as critical as the production, inventory, or sales data, should also use permanently defined data spaces that are separate from the permanent production reporting table spaces. In such instances, permanent table space should be accessible to the specific group (such as analysts) of regular SPD Server users.

The SPD Server administrator can use the **libnames.parm** file to configure paths that map to an area of reserved disk space on a host computer, creating a safe place for permanent tables with limited user access. To reserve permanent table space, the LIBNAME domain statement in the **libnames.parm** file should use the optional DATAPATH=, INDEXPATH=, and OWNER= statements to specify unique, appropriately sized disk areas for data tables and index tables. The OWNER= statement configures ownership and access. It is up to the SPD Server administrator to ensure that the paths named in domain declarations have access to sufficient disk space.

User access to permanent table spaces can be established via individual user account access privileges, or by establishing, through the owner of the domain, an ACL group of approved users. LIBNAME domain statements will 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, but modify the way the data is processed, or augment the production, inventory, or sales data in some manner with additional information. Those projects should be conducted in a test data space, safely isolated from the permanent space that is dedicated to critical production, inventory, or sales data. This design allows development trials to be conducted without risk of corrupting mission-critical data.

For example, the test data space used for a month-long development project could be considered a semi-permanent space; it requires the SPD Server administrator to grant access to an area where data can safely exist, isolated from production, sales, or inventory data, for a specific period of time longer than a single SPD Server user session. The "test" environment should persist long enough for works-in-progress to mature to production-quality (if so destined), 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.

Semi-permanent table space can be configured by an SPD Server administrator or by SPD Server users. It is recommended that administrators 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 information requests might be as important as standard production, inventory, or sales reports, but ad-hoc reporting has different data space needs. Ad-hoc reports tends to have a lower frequency of repetition and broader query scope than standard daily production, inventory, or sales reporting. Ad-hoc reports are usually best suited to temporary table space. The life spans of temporary table spaces begin and end with the user's SPD Server sessions.

Temporary table space is used for more than ad-hoc user reporting. Even data warehouse queries and reports that use permanent table space utilize 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 the query. Intermediate SPD Server tables and calculation metadata are normally deleted when the job terminates.

Any of the report types listed previously can require temporary table space for intermediate calculation tables. Temporary table space can be configured by an SPD Server administrator in the **libnames.parm** file, or by normal SPD Server users via LIBNAME domain statements submitted during an SPD Server session. The key to creating temporary table spaces is to use the optional TEMP=YES specification when the LIBNAME domain statement is issued, either in **libnames.parm**, or in submitted SPD Server

job code. All tables residing 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 illustrate the range of LIBNAME domains that can be created using the **libnames.parm** file. The code examples begin with the simplest forms of LIBNAME domain declaration and increase in complexity.

Example 1: Minimum Configuration for Domain Declaration

Example 1 demonstrates the syntax necessary 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) will 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

Example 2 demonstrates the syntax necessary to declare a LIBNAME domain with separate paths allocated for the domain data tables, index tables, and intermediate data files. The domain metadata will continue 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')";
```

Example 2 uses the domain path options DATAPATH=, INDEXPATH=, and WORKPATH=. Optimal performance can be achieved in this configuration when each domain path resides on a separate disk or on network components that can take advantage of parallelism opportunities.

The INDEXPATH= is designed to take advantage of multiple file systems. Beginning with SPD Server 4.3, index components can take advantage of the SPD Server RANDOMPLACEDPF feature. The RANDOMPLACEDPF feature enables administrators to configure smaller disk partitions for index space, which benefits SPD Server backup and recovery operations.

The WORKPATH= specified for the Example 2 SKULIST domain allows domain users to override the default workpath, if any, specified in the **spdsserv.parm** file.

Example 3: Query-Rewrite Domain Configuration

Example 3 demonstrates how to use temporary tables to configure a LIBNAME domain for performance when utilizing the SPD Server SQL query rewrite facility.

The SPD Server SQL query rewrite facility does 'behind the scenes' work to find the most processor-efficient method to evaluate submitted 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.

Example 3 creates a query rewrite domain named 'QRW' that uses distributed temporary SPD Server tables. In order to use SPD Server QRW, the following must be done:

- A specific domain for the query rewrite operations should be created in the **libnames.parm** file. This example names the query rewrite domain 'QRW'.

- The **spdsserv.parm** file should contain a TMPDOMAIN=<*QRW-domain-name*> statement that references the QRW domain that was created in the **libnames.parm** file.

Libnames.parm file code (note 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')";
```

Spdsserv.parm file code (note 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

Example 4 uses a combination of **libnames.parm**, **spdsserv.parm**, and user-issued SAS code submitted to SPD Server to create multiple domains that house the following:

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

In the Example 4 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, and at the same time, use temporary tables with sufficient disk space to perform large or optimized intermediate table calculations. Multiple data and index paths are specified to take advantage of RAID-configured disk arrays.

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. Content in USERTBLS can be saved and deleted by SPD Server users.

```
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')";
```

Spdsserv.parm file code uses the TMPDOMAIN=SPDTEMP statement to reference the domain that was created for temporary tables, and uses the WORKPATH= statement to identify an array of RAID-enable 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')";
```

SAS code submitted to SPD Server by the user connects to the PROD, USERTBLS, and SPDTEMP domains, and configures SPDTEMP as a temporary domain space. Tables in the SPDTEMP domain will be 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;
```

Setting Up the SAS Scalable Performance Data Server Performance Server

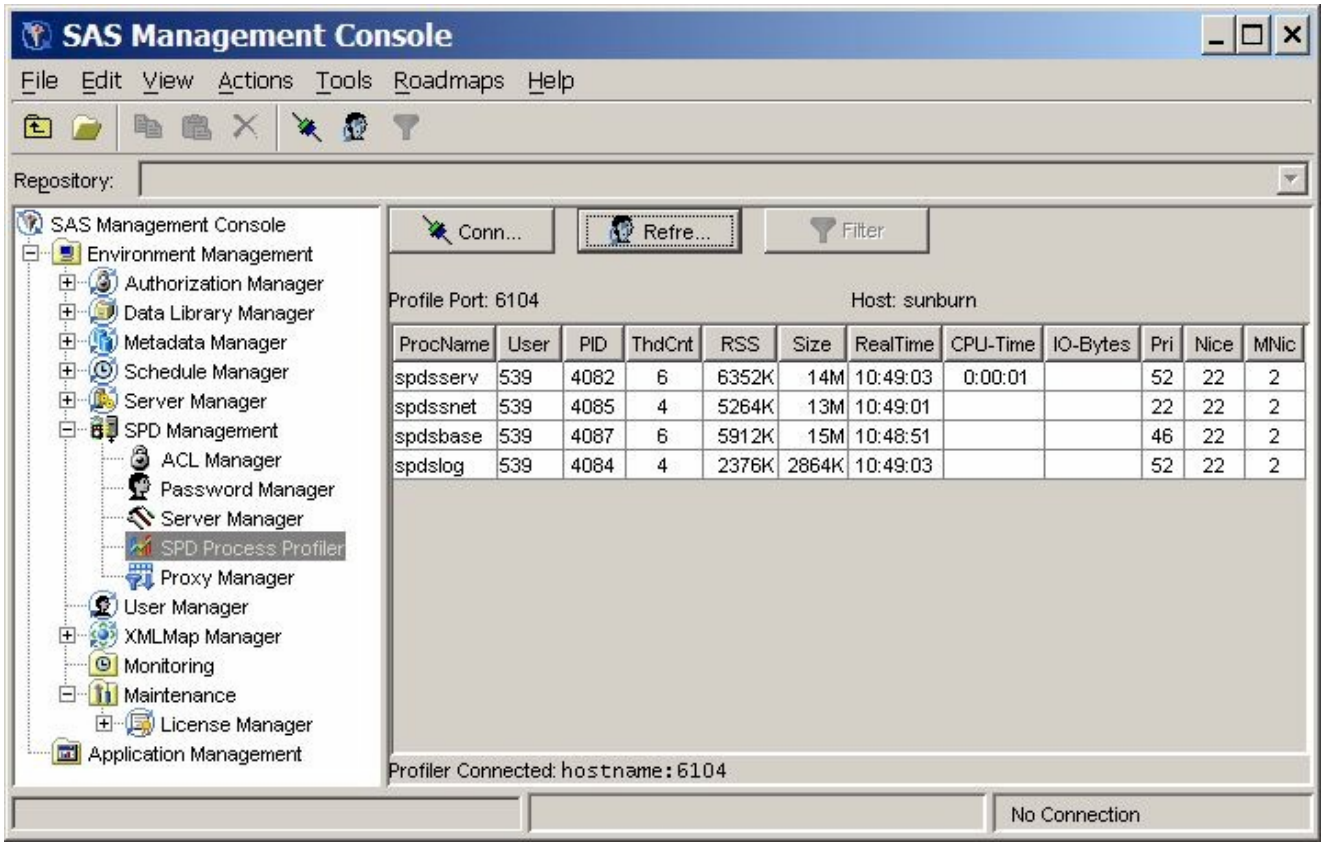
---

Introduction

SAS SPD Server 4.4 provides a performance monitoring server called **spdsperf**. The SPD Server Performance Server is an optional component and is not required for normal operation of SPD Server.

The purpose of the SPD Server Performance Server is to gather SPD Server process performance information and to post it to the SPD Server Management section of the SAS Management Console application. The SPD Server performance information consists of memory and resource allocations by users, and SPD Server processes 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 are created and terminated by SPD Server users and jobs.

The SPD Server Performance server information 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 to the **Environmental Management** folder in SAS Management Console. When you expand the **SPD Management** folder, the bottom-most utility is the **SPD Process Profiler**. When you highlight the **SPD Process Profiler** utility, the process information table that is located in the right panel displays performance summary statistics. Each row in the performance summary statistics table provides information about a single SPD Server process that was spawned on the SPD Server Name Server residing at the specified PID, or port ID.



The display of memory and resource allocations in the **SPD Process Profiler** allows SPD Server Administrators to use the SAS Management Console as a handy view point 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.

The SPD Server Performance Server is not limited to displaying its 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 was created by the SPD Server Performance Server.

Starting the SPD Server Performance Server

The SPD Server Performance Server can be started in two different ways. The Performance Server can be invoked by command line, or it can be launched by calling an **rc.perf** script that is configured for your location's SPD Server installation.

The SPD Server Performance Server process is configured by default to output the captured performance data to the user's screen. To disable the user screen display, redirect **stdout/stderr** to **/dev/null**. Redirecting the screen output also makes it easier to run **spdsperf** in the background, or as an orphan.

Start Performance Server from 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 SPD Server must be restarted, the SPD Server Performance Server must also be shut down and restarted after SPD Server is restarted. The Performance Server utility is not compatible with SPD Server releases that preceded SPD Server 4.4.

Usage:

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

where

SMA is the shared memory address for the SPD Server Global Control Block (GCB). The GCB is an internal SPD Server data map that contains all of the options that are passed to the SPDSBase server when it starts. To get the value for the GCB address, SPD Server must be started. After the SPD Server Name Server starts the first SPDSBase process, issue a UNIX **ps** command and look in the output to view the address parameters that were passed to SPDSBase. The GCB shared memory address should be found in the **ps** output data.

NSP is the process-ID number of the SPD Server Name Server whose family of spawned processes you want to monitor

SNP is the process-ID number of the SPD Server SNet server.

PLP is the listening port number that SAS Management Console will use to contact the Performance Server

LOG is the path specification that you want to write the profile log to.

SEC is the optional property that specifies the number of seconds that transpire between instances of performance monitoring data captures. The permissible range for SEC property values is integers that are greater than or equal to 1, and less than or equal to the declared SPD Server MAXINT value.

CNT is the optional property that specifies the count, or number of performance monitoring data captures that you want the Performance Server to take. The permissible range for CNT property values is integers that are greater than or equal to 0, and less than or equal to the declared SPD Server MAXINT value. A CNT value of 0 requests an infinite number of data captures.

Start Performance Server from Rc.perf Script

You can also start the SPD Server Performance Server by calling the **rc.perf** script during start-up. The next section contains a sample **rc.perf** script that you can cut and paste into an editor of your choice, and customize for use with your SPD Server installation. SPD Server 4.4 also ships with a sample **rc.perf** script that you can modify. The sample **rc.perf** script is located in your SPD Server installation folders at:

.../samples/perfmon/rc.perf.

Either example file can be used to create your custom **rc.perf** script.

You must make the following changes when you customize your version of the **rc.perf** script:

1. The sample script below uses the default SPD Server installation path of

   usr/local/spds.

   If your SPD Server installation uses a custom path, you must update the INSTDIR path setting to update your installation path setting.

2. You must update the UNIX environment setting for DISPLAY. This environmental variable tells the X server where to display the window for the Performance Server program.

3. The sample script uses the default SAS Name Server port (NSPORT) and SAS SNet port (SNPORT) assignments. If your SPD Server installation uses different NSPORT and SNPORT assignments, you must update the NSPORT and SNPORT settings in the sample script to the port addresses that are used in your SPD Server installation.

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 below uses a -PARGS setting of 0, which requests an infinite number of performance information captures. If you do not change the default number of information captures from 0 (infinity), you may wish to modify your **rc.killspds** script to shut down the **rc.perf** process when you shut down SPD Server.

Sample Rc.perf Script

The sample code below is a typical **rc.perf** script that you can modify for use at your own site. Follow the instructions in the section above to customize the script for your SPD Server installation. In order to assist you, the values that you may need to change have been highlighted in a lighter color. It is recommended that you copy and paste the text below into a text editor of your preference, 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

The SPD Server Performance server can also be configured 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 located 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, they will be parsed and formatted for SAS processing.

There is a sample SAS script for importing the parsed log file data. The SAS script is located at

```
.../samples/perfmon/PerfDataSample.sas
```

in your SPD Server installation. .

SAS Scalable Performance Data Server Security

SAS Scalable Performance Data Server Security

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 (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

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 should only be accessed by SPD Servers running ACLs. Likewise, areas created without ACLs should only be accessed by SPD Servers using -NOACL.

SPD Server comes bundled with the SAS Management Console (SMC). 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 SMC also manages ACLs using the same capabilities provided by PROC SPDO.

Enabling ACL Security

You enable SPD Server security with the **-ACL** option on the **spdsserv** 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 can only read or write files that have the appropriate file and directory access permissions to the SPD Server's user's ID. Use the UNIX 'umask' command to restrict the desired 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 userID matches the OWNER= specification can create tables in this domain. (However, that user can grant other users creation 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 spdssrv 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 may 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

You disable SPD Server security by using the **-NOACL** option with the **spdsserv** 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. The following security applies.

- UNIX File-Level Protection without ACL Security
- Control of LIBNAME Domains by the System Administrator without ACL Security
- Example Server Setup without ACL Security

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 -acldir
  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';
```

ACL Security Examples

This section provides a series of examples that administrators and users can use to understand how security is implemented in an SPD Server environment.

- Domain Security
- LIBACLINHERIT
- Anonymous User Account
- Read-Only Tables
- Domain Security and Group Access
- Bringing a Table Offline to Refresh
- Bringing a Domain Offline to Refresh Tables
- ACL Special Users
- Column Level Security

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

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

The ID 'prod2' is in the group 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 can *not* 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 then 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 may 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 priviliges to */
/* the domain                     */

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


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

      PROC SPDO library=d1 ;

/* Assign who owna the ACLs */

      set acluser admin1 ;
```

The two ACL commands in the 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 controling 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    */
/* inlcude groupread if other       */
/* users  in the same group as      */
/* admin2 need to be able to read   */
/* tables that were created by      */
/* admin2.                          */

       add ACL admin1_table1 /
         generic
         read
         groupread
         groupalter ;

       list ACL _all_;
       run;

/* admin1 has write privileges to    */
/* the domain                        */

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

```
        end ;
      run ;

/* generic ACL allows all users to    */
/* read the tables                    */

      PROC SPDO library=d1 ;

/* assign who owns the ACLs */

      set acluser admin1 ;

/* Add a table and modify LIBNAME ACL */
/* for groupread and groupwrite. The  */
/* ACL MUST include groupread to give */
/* users in the same group as admin2  */
/* the ability to read tables created */
/* by admin2                          */

     add ACL admin1_table2 /
       group1=(y,n,n,n)
       admingrp=(y,n,n,y) ;
       list ACL _all_;
     run;

/* admin2 has write privileges to the */
/* domain                             */

     data admin2d1.admin2_table ;
       do i = 1 to 100 ;
       output ;
       end ;
     run ;

/* Admin2 must use PROC SPDO to allow */
/* users read access to the table.    */
/* The PROC SPDO example below uses   */
/* generic syntax with a read.  This  */
/* provides any user outside of the   */
/* admingrp read access to tables     */
/* that were created by acdmin2.  The */
/* groupread and groupalter allow     */
/* access by users within admingrp.   */

    PROC SPDO library=admin2d1 ;

/* Assign who owns the ACLs */

    set acluser admin2 ;

/* Modify LIBNAME ACL for groupread   */
/* and groupwrite. The ACL MUST       */
/* include groupread if other users   */
/* in the same group as admin2 need   */
/* to read tables created by admin2.  */

    add ACL / generic
      read
      groupread
      groupalter ;
```

```
     list ACL _all_;

/* admin (same group) can read the     */
/* table                               */

     PROC PRINT data=d1.admin2_table
       (obs=10) ;
       title 'read by admin' ;
     run ;

/* Admin has been given the ability to */
/* modify or replace tables created by */
/* admin2 with 'groupalter'            */

     data d1.admin2_table ;
       do i = 1 to 100 ;
       output ;
       end ;
     run ;

/* Provide other users in same group   */
/* read access to the table            */

     PROC SPDO library=admin2d1 ;

/* assign who owns the ACLs */

     set acluser user3 ;

/* Modify LIBNAME ACL for groupread    */
/* and groupwrite. The ACL MUST        */
/* include groupread if other users in */
/* the same group as admin2 are to be  */
/* able to read tables that were       */
/* created by admin2                   */

     add ACL user3_table /
       groupread ;
     list ACL _all_;
```

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 may 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. Levels 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, which 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 create a domain where only the owner can create new tables, but everyone in the group can read the data.

The domain 'JILL' is defined as:

```
    LIBNAME=JILL
      pathname=c:\spdsdata\jill
      owner=sasgmc;
```

As the owner, sasgmc must give authority to users/groups to access the domain. The PROC SPDO statement below allows the group (ADMINGRP) to access the domain and to read and write to any existing tables. Without the ability to alter tables, however, ADMIN group members will not be able to replace tables or change their structure (indexes).

Additional ACL statements restrict the activities of the user BOB. Bob can only read the sensitive table LOCK and is not even permitted to read all the columns of LOCK (salary is excluded).

```
     LIBNAME grant sasspds 'grant'
       host='localhost'
       serv='5259'
       user='sasgmc'
       prompt=yes
     ACLSPECIAL=yes;

/* generate some dummy data */

     data grant.nolock
       grant.lock;
       id=1;
       salary=2000;
     run;

     PROC SPDO library=grant ;

/* Assign who owns the ACLs */

     set acluser sasgmc;

/* Clean Up */

     delete ACL/libname;
     delete ACL/generic;
     delete ACL lock;
     delete ACL lock.salary;

/* For the Domain GRANTS */

     add ACL / libname

/* GROUPWRITE */
```

```
/* uncomment to allow group members */
/* to create new tables             */

/* Provide members of the group     */
/* ADMIN read access to the domain  */

     groupread;

/* For all tables in GRANTS domain   */
/* allow Group read and write        */

     add ACL / generic
       groupread
       groupwrite;

/* Create ACL for table LOCK         */

     add ACL lock;

/* Update the ACL for table LOCK     */
/* only allow Bob to read it         */

     modify ACL lock
       bob=(y,n,n,n);

/* Create ACL to the column SALARY  */
/* in the table LOCK                */

     add ACL lock.salary;

/* Prevent Bob from seeing the      */
/* salary column                    */

    modify ACL lock.salary /
      bob=(n,n,n,n);

    list ACL _all_;
    run;

    LIBNAME grant clear;

/* Now test the above application   */
/* Connecting to a LIBNAME domain   */
/* requires at the minimum READ /   */
/* GROUPREAD for the LIBNAME ACL    */

    LIBNAME s2 sasspds 'grant'
      host='localhost'
      serv='5200'
      user='bob'
      prompt=yes;

/* Once connected, reading tables    */
/* requires a table level READ ACL.  */
/* This was granted above using the  */
/* GENERIC ACL instead of listing a  */
/* table name. This causes GROUPREAD */
/* to be granted to all tables in    */
/* the  domain.                      */
```

```
/* NOTE: the salary column is not    */
/* written because BOB was forbidden */
/* to read it in the LOCK.SALARY ACL.*/

     PROC PRINT data=s2.lock;
     run;

/* See above for rules.    */
/* Note salary is written. */

     PROC PRINT data=s2.nolock;
     run;

/* Appending NOLOCK to LOCK requires   */
/* write access to all columns of lock */
/* (or simply write access to the whole*/
/* table with no column restrictions), */
/* and read access to NOLOCK at the    */
/* table ACL level.                    */

     PROC APPENT base=s2.lock
       data=s2.nolock;
     run;

/* Creating new tables requires WRITE   */
/* permissions for LIBNAME ACL.         */

     data s2.test;
       x=1;
     run;

/* ACL ALTER on the table will be       */
/* required when creating an index.     */

     PROC SQL;
         create index id on s2.lock(id);
     quit;

/* Clean Up */

     PROC SQL;
       drop table s2.test;
     quit;

     LIBNAME s2 clear;
```

Controlling SPD Server Resources with PROC SPDO and ACL Commands

- Using PROC SPDO
- Using ACL
- ACL Concepts
- ACL Command Set

Using 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 detail on LIBNAME Proxy Commands are discussed in more detail in PROC SPDO Commands and Resources.

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

Here are some commonly used ACL concepts that are used in SPD Server.

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

Allows you to specify separate ACLs to secure specific columns of a table. Column ACLs take **away** access privileges at the column level which are granted by the table ACL.

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 wild card. 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 wild card 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 wild cards 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, etc.)
- a LIBNAME domain.

Two-Part Resource Name

Two-part names identify a column entry within a table. Use the normal SAS convention *table.column* when specifying the table and column that you wish 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 for more information on user specific ACL entries.

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. Optionally, you may 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 which 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 ...* may be one-part resource names or two-part table column names.

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*
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.

To add ACL definitions, the ACLUSER must be the same as the LIBNAME user or the LIBNAME user must be assigned ACLSPECIAL=YES privileges.

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 *acl1 acl2...* [C=*cat* T=*type*] */options userlist*;

MODIFY ACL _ALL_ */options userlist*;

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 *userlist*.

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 *username = (Y/N,Y/N,Y/N,Y/N)* for (READ, WRITE,ALTER,CONTROL).

MODIFY ACL Examples

- Modify LIBNAME Domain ACL
- Modify ACL MINE
- Modify Generic ACL
- Modify All ACLs
- Modify Column ACL
- Modify Generic Column ACL
- Modify ACL for a Catalog
- Modify Generic ACL for Catalog Entries

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 *acl1 acl2...* [/*options*];

LIST ACL _ALL_ [/*options*];

Lists information about specific ACLs *acl1 acl2...* where ACL entries *acl1 acl2...* may 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.

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 *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...* may 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.

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

```
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;
```

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, for example, "@SPDSUSR" = "SOMEUSER".

Symbolic Substitution Row Level Security

A powerful use of symbolic substitution is deploying row level security on sensitive tables that utilize 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 only be allowed to see their user or group rows.

Symbolic Substitution Example

```
PROC SQL;
   connect to sasspds
     (dbq="path1"
      server=host.port
      user='anonymous');

/* queries comparing literal constants:   */
/* 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;
```

DICTIONARY.PWDB and DICTIONARY.ACLS

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. 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, 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");
```

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

---

Managing SPD Server Passwords, Users, and Table ACLs

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 associated with an individual user. SPD Server uses two means of user authentication. The first type of user authentication is Lightweight Directory Access Protocol, also called LDAP. The second type of 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 in this document on LDAP Authentication contains more details on using LDAP with SPD Server.

The Password Manager Utility Psmgr

The **psmgr** utility manages a password table that enables access to the server. When you start SPD Server, the command line option *-acldir* specifies the location (directory) where the table resides. The owner of the password table, typically the SPD Server administrator, can update the table.

The password table contains the following 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

User IDs are restricted to 8 characters, and do *not* have to correspond to any system user ID.

Passwords are also restricted to 8 characters.

Psmgr table passwords must have a minimum of 6 characters, one of which must be numeric. New passwords must be different from an individual's last six passwords. The password cannot contain the user ID in any part of the password.

If a user makes three consecutive failed attempts to connect to a server, his or her user ID becomes disabled. Until an administrator resets the user ID, the user will not be able to connect to a server.

If you are upgrading to SPD Server 4.4 from SPD Server 3.x, the SPD Server 4.4 password manager utility must be repopulated 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 that contains the password table. (Or you can specify a table that has not yet been created.)

For a UNIX system, use the command

```
psmgr installdir/site
```

The command invokes the password manager utility and specifies the directory for the password table.

For a Windows system, use the command

```
psmgr
```

The command invokes the password manager 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 4.4, it is necessary to convert your SPD Server 3.x password file to 4.4 format. This assumes that you want to continue to use the same set of active SPD Server user IDs in 4.4 that you did in SPD Server 3.x. To upgrade your SPD Server password file to 4.4 format from 3.x format, do the following:

1. Start the SPD Server 3.x psmgr utility using your SPD Server 3.x psmgr table.
2. Export your SPD Server 3.x psmgr password table.
3. Start the SPD Server 4.4 psmgr utility with a new table.
4. Import the export file that you created earlier into the new table.

Example:

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

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

/Installdir4_0/bin/psmgr /Installdir4_0/site

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

This creates a psmgr table from an old format psmgr table that exists at `/installdir3_0/site`.

Adding New Users With Psmgr

Depending on whether you want to create the password for the user, or whether you want the user to create his or her own password, there are two ways to set up a new user.

Add a New User Who Creates His Own Password

This two-part method requires the new user to know how to use one of the following libname options to change his password: **CHNGPASS=**, **NEWPASSWD=,** or **PROMPT=**.

Table 16-1: **Step 1 of 2: Add a New User With Psmgr**

Instructions for **command** or *variable*	**Remarks**
>**psmgr** /SPDS/pwdir	To get started, type the **psmgr** command, plus the *password directory* at the command prompt.
Enter command > **add** *tom*	Type the **add** command, followed by the *user ID.*
Enter password for tom > *password*	Type a temporary *password* for the user.
Verify > *password*	Retype the temporary *password*, for accuracy.
Enter authorization level (0 to 7) for tom: *>0*	Type an authorization level *number from 0 to 7*, depending on the level you want to assign.
Enter IP Address > 11.21.1.217	If you want to limit user access to a specific client machine, type the *IP address of the client machine*; otherwise, skip by pressing Enter.
Enter password expiration time in days > *nn*	Type in the *number of days* you want the password to be valid.
Enter group name >	If the user is part of a group, type the *group name*, or skip by pressing Enter.
User tom added >	At this point, user tom's password is good for one logon.

Table 16-2: **Step 2 of 2: Change Password With NEWPASSWD= or CHNGPASS=**

from SAS, Submit ...	**Remarks**
LIBNAME *mylib sasspds "spdsdata"* **host=***"samson"* **serv=***"5200"* **user=***"tom"* **password=***"xyz123"* **NEWPASSWD==***"abc123"*	Changes the user's password using **NEWPASSWD=.**

or	
LIBNAME *mylib sasspds "spdsdata"* **host=***"samson"* **serv=***"5200"* **user=***"tom"* **password=***"xyz123"* **CHNGPASS=**YES	Prompts the user for a new password.

Add a New User and Set Password for User

This example adds a new user to the password table using **psmgr**. The password expires immediately after the user is created. The SPD Server administrator creates the new password and sends it to the user.

Table 16-3: **Using CHNGPASS= to Assign a New Password**

Instructions> **command** or ***variable***	**Remarks**
Enter command > chgpass *gus*	System administrator issues chgpass command.
Enter old password for Gus > *password*	At the prompt, the system administrator types in user Gus' old password.
Enter new password for Gus > *password2*	At the prompt, the system administrator types in user Gus' new password.
Verify new password > *password2*	At the prompt, the system administrator types in user Gus' new password again.
Password updated for Gus >	The system administrator must now tell the user Gus what his new password is. User Gus' password will expire in the number of days specified in the previous **psmgr** command.

Psmgr Commands

The password manager psmgr is an interactive utility program. It reads commands and operands from your terminal, and prompts you for input when necessary. You can also feed a file of commands to the utility (see details that follow), structuring each command so that no prompt is required.

The command operands are positional and must be separated by blanks. If you give an insufficient number of operands, the utility prompts you for the rest. Password operands, obtained with a prompt, are not echoed back to the terminal.

ADD

Adds a new user to the password table.

AUTHORIZE
Authorizes a given user to change the password table.

CHGAUTH
Changes the authorization level for a given user.

CHGEXPIRE
Changes the expiration date for a given user's password.

CHGIP
Changes IP address from which the user must connect to the SPD Server.

CHGTIMEOUT
Changes the login time-out date for a given user's password.

CHGPASS
Changes the password for a given user.

DELETE
Deletes given user ID.

EXPORT
Exports the current password table to a plain text flat file.

GROUPDEF
Defines a new ACL group entry.

GROUPDEL
Deletes an ACL group entry.

GROUPMEM
Updates the group list for a given user ID.

GROUPS
Lists all the ACL groups in the password table.

HELP
Displays general or command-specific help.

IMPORT
Imports user information from a flat file, created with the **export** command, to the password table.

LIST
Lists all the users in the password table.

RESET
Resets a user's password.

QUIT
Ends the session and exits from psmgr.

Command Details

ADD

Adds a user/password entry to the password table.

Syntax

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

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

Arguments

Use the following arguments with ADD:

username
: the name (user ID) of an SPD Server user, up to 8 characters. The SPD Server user ID does *not* have to correspond to any system user ID.

passwd
: a prompt for the user's password, up to 8 characters maximum. Psmgr tables require 6 characters minimum with at least one numeric character and one alphabetic character. The argument is repeated to verify the password entry.

privilege
: a number from 0 to 7.

  The value assigns access privileges to the user.

  0-3 are equivalent.
  Use to specify a normal, that is, a non-privileged user.

  4-7 are equivalent.
  Use to specify a 'special' user. Special users can update the password table and override any ACL restrictions on SPD Server tables. For this reason, you may choose to restrict

special privileges to only the SPD Server user ID and password for yourself, the SPD Server administrator.

ip_addr
an optional numerical IP address or a dash (-) indicating that no IP address is being specified. Use the optional IP address to restrict the user's access to SPD Server to a specific IP address.

Note: The IP address operand is not validated.

expiretime
an optional password expiration time or a dash (-) indicating that no password expiration time is being specified. The expiration time requires the user to change passwords after the specified number of days has expired. The value, specified in days, represents the number of days from today (the current day) that the password will be valid.

group
default group for the user or a dash (-) indicating no default group is defined. If specified, the group definition must exist by a previous **GROUPDEF** command. Group affiliation may be changed by a **GROUPMEM** command.

timeout
an optional maximum amount of time allowed between successful logins before the account is disabled or a dash (-) indicating no time-out is being specified.

AUTHORIZE

Authorizes the user to modify the password table.

Syntax

```
authorize username userspasswd
```

Arguments

username
the name (user ID) of an SPD Server user previously set up in the password table.

userspasswd
a valid password previously set up for the user in the password table.

Description

Only a 'special' user can update the password table. That is, to use the modify commands such as add and delete, you must be a 'special' user or the owner of the password table.

If you are not the table owner, you can use the authorize command to authorize yourself for password table updates. You do this by entering the name and valid password for a user in the password table, then marking the user as 'special' (specify the user's access privilege as 4 or more).

For example, assume that we use the psmgr list command and obtain the following output:

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

You can grant yourself "bar's" privileges by using the authorize command, specifying "bar" as the user name "bar", along with bar's password "barpwd1".

Example

```
authorize bar barpwd1
```

CHGAUTH

Changes the authorization level for a given user.

Syntax

```
chgauth username authlevel
```

Arguments

Use the following arguments:

username
the name (user ID) of an SPD Server user previously set up in the password table.

authlevel
an authorization level for the user, from 0-7. See the ADD command for an explanation of the values.

CHGEXPIRE

Changes the expiration date of the password for a given user. Initially, all new user IDs are created with an expired password authorization level.

Syntax:

```
chgexpire username exptime
```

Arguments

Use the following arguments:

username
the name (user ID) of an SPD Server user previously set up in the password table.

exptime
a password expiration time. The expiration time requires the user to change passwords after the specified number of days has expired. The value, specified in days, represents the number of days from today (current day) that the password will be valid.

CHGIP

Changes IP address from which the user must connect to the SPD Server. The IP address of the machine on which the SAS, ODBC, JDBC, or SPQL client software is running must match the IP address that is entered in the password database.

Syntax:

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

Arguments

Use the following arguments:

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

IP Address
The IP address of the machine from which the user must connect. The IP address must be specified numerically using the xxx.xxx.xxx.xxx format. The IP address is not verified at the time of entry. Invalid and incorrect IP addresses will be noted as errors in the SPD Server log and will cause the that user's future connection attempts to fail. The default value is blank.

CHGTIMEOUT

Changes the login time-out date of the password for a given user.

Syntax:

```
chgtimeout username timeoutperiod
```

Arguments

Use the following arguments:

username
the name (user ID) of an SPD Server user previously set up in the password table.

timeoutperiod
a password login time-out period. The time-out period requires the user to successfully log in before the specified number of days has expired. The value, specified in days, represents the number of days from the last successful login that the password will be valid.

CHGPASS

Changes the password for a given user.

Syntax

```
chgpass username oldpwd newpwd
```

Arguments

Use the following arguments:

username
the name (user ID) of an SPD Server user previously set up in the password table.

oldpwd
the user's old password. (You must know the user's old password.)

newpwd
a new password for the user. If you are prompted for the new password entry, you are prompted again to re-enter it, because it is not echoed back to your terminal. The new password must be different than the last 6 passwords. The new password must also contain at least 6 characters, one of which must be numeric, and one of which must be alphabetic, and the password must not contain the user ID.

DELETE

Deletes a user from the password table.

Syntax

```
delete username !
```

Arguments

Use the following arguments:

username
the name (user ID) of an SPD Server user previously set up in the password table.

!
verifies that you intend to delete the user from the password table. If you do not specify "!", you receive a "Y" or "N" prompt to confirm the delete.

EXPORT

Extracts the contents of the password table into a flat file.

Syntax

```
export textfile
```

Arguments

Use the following argument:

textfile
names the flat file to create that will contain the contents of the 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 file is the representation of what is stored in the password table.

When you have many global changes (changes that affect many users), it may be easier to edit the file than to use psmgr. After making any changes and corrections in the file, you can use the psmgr import command to construct a new, modified password table.

GROUPDEF

Defines a new ACL group in the password table.

Syntax

```
groupdef groupname
```

Arguments

Use the following argument:

groupname
names a group. The name must be unique, up to 8 characters. The groupname entry verifies that the groups specified with the groupmem command are valid.

GROUPDEL

Deletes a group from the password table.

Syntax

```
groupdel groupname !
```

Arguments

Use the following arguments:

groupname
the name of a group previously set up in the password table.

!
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 confirm the delete.

GROUPMEM

Specifies up to five ACL groups for a given user.

Syntax

```
groupmem username groupname
```

Arguments

Use the following arguments:

username
the name (user ID) of an SPD Server user previously set up in the password table.

groupname
names an ACL group. The name must be unique, up to 8 characters. Enter each group name separated by a space. The first ACL group specified becomes the default ACL group for the user.

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

GROUPS

List the all the ACL groups in the password table.

Syntax

```
groups
```

Arguments

No arguments.

HELP

Displays a brief summary of the **psmgr** command set or command-specific help information.

Syntax

```
help [command]
```

Arguments

Use the following argument:

command
a psmgr command. If you specify a command, you get a short description of the command. If you issue a help statement without operands, you get a list of all available psmgr commands.

IMPORT

Adds user information from a flat file, created with the **export** command, to the password table.

Syntax

```
import textfile
```

Arguments

Use the following argument:

textfile
the flat file to import containing user definitions to add to the password table.

Description

The import command reads in a flat file, interpreting each line as a user definition for the password table. Typically, the file is output from an earlier run of the psmgr export command used on the same or another password table.

If, during import processing, the utility encounters the identical file user name (SPD Server user ID) in the password table, it skips the line. Psmgr also prints a message indicating that the line was ignored.

LIST

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

Syntax

```
list [username]
```

Arguments

Use the following argument:

username
: the name (user ID) of an SPD Server user previously set up in the password table. If no user 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 the password for a given user. You use this command to reset a user's password after three failed attempts to connect to a server. After the third failed attempt, the user is disabled. After the password has been reset, the user must change the password before he or she can connect to a server.

Syntax

```
reset username newpwd newpwd
```

Arguments

Use the following arguments with RESET:

username
: the name (user ID) of an SPD Server user, up to 8 characters. The SPD Server user ID does *not* have to correspond to any system user ID.

newpwd
: a prompt for the users password, up to 8 characters maximum. The password must have 6 characters minimum, and contain at least one numeric character and one alphabetic character. The argument is repeated to verify the password entry.

Note: The new password expires immediately and must be changed at the next logon to SPD

Server or with the psmgr chgpass command.

Example

```
reset tom abc123 abc123
```

This example resets tom's password to "abc123".

QUIT

Ends the session and exits from psmgr.

Syntax

```
quit
```

Using a File as Input to Psmgr

As mentioned earlier, you can enter psmgr commands into a file, then pipe the file into stdin of the utility.

Example

Here is a sample command file named *pscmds*:

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

The above command file contains the password "barpwd" for user "bar". Because the command file contains user IDs and user passwords, you may 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 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 Utility

SPD Server supports the SAS Management Console Utility (SMC). The SAS Management Console is a GUI utility that the SPD Server administrator can use to manage passwords and ACLs. Now administrators that need to configure passwords and ACLs can choose between using the SAS Management Console or the traditional SPD Server management tools of PROC SPDO and the **psmgr** utility.

More extensive information about using SPD Server with the SAS Management Console utility is available in the Administrator's Guide documentation on Administering and Configuring SAS Model Manager using SAS Management Console.

LDAP User Authentication

SPD Server users can be authenticated via either Lightweight Directory Access Protocol (LDAP) or by the **psmgr** password database utility. LDAP Authentication is done by an LDAP server that runs on the SPD Server machine. When you use LDAP Authentication, the operating system handles password maintenance. LDAP authentication also adds the benefit of operating system-level security and convenience.

LDAP Authentication Notes

There are several important points to remember when you use an LDAP server to perform SPD Server user authentication:

- SPD Server users can be either authenticated by an LDAP Server, or by the password database, but not both. The type of authentication to be performed is specified in the **server.parm** file, which is read when SPD Server starts up.
- If you are changing from using LDAP user authentication to using the password database for authentication, all LDAP parameters must be removed from the SPD Server **server.parm** parameter file. You must also 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 password database is still needed. When using LDAP, a password database record is still required for each SPD Server user. SPD Server still uses the password database to perform user access control and other tasks that are not related to user authentication.
- Users that connect to an SPD Server must have a corresponding login information on the LDAP Server. The LDAP Server UserID and the SPD Server UserID formats are the same. The login password follows the host operating system format. UserIDs must be 8 characters or less in length.
- Some LDAP server products may require users to enter host login information. In such cases, confirm with your LDAP server Administrator that the host login information exists in the LDAP database.
- If you are using LDAP user authentication, and create a user connection that uses the NEWPASSWORD= LIBNAME option, the user password will not be 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 also records 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 Windows Installation Guide and in the SPD Server UNIX Installation Guide.

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

PROC SPDO

Note: For additional information on using PROC SPDO, see the chapter "Controlling SAS Scalable Performance Data Server Resources with PROC SPDO and ACL Commands" in the *SAS Scalable Performance Data Server Adminstrator's Guide*.

Special SPDO 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

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*> LIBNAME that was dispatched from the SPD Server host. Accessible proxies are anonymous proxies that are owned by the LIBNAME owner. If the libname has ACLSPECIAL privileges, the privileged members will also own the accessible proxies.

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 to the PROC SPDO *lib*=<*libname*> libname that was dispatched from the SPD Server host and were created with the LOCKING=YES LIBNAME option. Accessible proxies are anonymous proxies that are owned by the libname owner. If the libname has ACLSPECIAL privileges, the privileged members will also own the accessible proxies. 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
```

2. Set the user to ANONYMOU and specify process ID (pid) 17610:

```
set user anonymou 17610;

NOTE: User ANONYMOU connected to proxy operator port with pid=17610.
```

3. Show every LIBNAME for user ANONYMOU for this proxy:

```
showlibname _all_;
LIBREF(FOO):Pathname assigned=/bigdisk/test/qabig1_dev/
LIBREF(FOO):ACL Owner=
LIBREF(FOO):ACL Defaults(R,W,A,C)=(Y,Y,Y,Y)
```

4. Show all of the open tables in LIBNAME FOO:

```
showlibname FOO/data=_all_;

NOTE: No data sets currently opened for LIBREF FOO.
```

5. Show all of the accessible tables in LIBNAME FOO:

```
showlibname FOO/dump=_all_;

LIBREF(FOO):Dataset name=BIGX
LIBREF(FOO):ACL Owner=ANONYMOU
LIBREF(FOO):ACL Defaults(R,W,A,C)=(N,N,N,N)
LIBREF(FOO):Dataset name=X
LIBREF(FOO):ACL Owner=ANONYMOU
LIBREF(FOO):ACL Defaults(R,W,A,C)=(N,N,N,N)
```

6. The user ANONYMOU 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 detail information about 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 is 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 open tables. You can also submit the SHOWLIBNAME *libref* / DATA=_ALL_ command after you submit the OPER INTERRUPT COMMAND, to see if 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

You can use PROC SPDO to dynamically refresh SPD Server parameter and LIBNAME files. If you make changes to your spdsserv.parm file or 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 commands. However, user sessions are finite. The user that you select with SET USER or SET USER/RECORD 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 will 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.

SAS Scalable Performance Data Server Hybrid Index Utility Ixutil

---

SPD Server System Administrator Hybrid Index Utility Ixutil

The Hybrid 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 utilized 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*> ;
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 runstats displays 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 runstats 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.

-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
- Retrieve Disk Usage and Fragmentation Statistics
- Reorganize the Index
- Review Disk Usage Statistics
- Create a Join Index
- Generate Join Index Statistics
- Delete the Join Index

Create a Hybrid Index

Assume there is an SPD Server domain named **mydomain** 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 X -dsn test -libpath /spds

SAS Scalable Performance Data Server 4.30(TS M0) Build(May 9 2005, 01:16:45)
Index File Utility
Copyright (c) 1996-2005 by SAS Institute Inc, Cary NC 27513 USA

Statistics for Hybrid Index X:
-------------------------------------------------
+--segment_size = 8192
+--n_segments_in_tbl = 5
+--n_values_in_index = 2
+--n_seglist_values = 2
+--n_seglist_chunks = 3
+--avg_chunks_per_list = 1.500000
+--idx_file_bytes = 13176.000000
+--idx_garbage_bytes = 4232.000000
+--percent_idx_garbage = 32.119003
```

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 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.

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 4.30(TS M0) Build(May 9 2005, 01:16:45)
Index File Utility
Copyright (c) 1996-2005 by SAS Institute Inc, Cary NC 27513 USA

Reorg for Hybrid 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 4.30(TS M0) Build(May 9 2005, 01:16:45)
Index File Utility
Copyright (c) 1996-2005 by SAS Institute Inc, Cary NC 27513 USA

Statistics for Hybrid Index X:
--------------------------------------
+--segment_size = 8192
+--n_segments_in_tbl = 5
+--n_values_in_index = 2
+--n_seglist_values = 2
+--n_seglist_chunks = 2
+--avg_chunks_per_list = 1.000000
+--idx_file_bytes = 8920.000000
+--percent_idx_garbage = 0.000000
```

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**. A 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 4.40(TS M0-WIDE)
Build(Aug  2 2006, 15:45:44)  Index File Utility
Copyright (c) 1996-2006 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 4.40(TS M0-WIDE)
Build(Aug  2 2006, 15:45:44)  Index File Utility
Copyright (c) 1996-2006 by SAS Institute Inc, Cary NC 27513  USA


Stat of Join Index Table1.jdxid.table2.jdxid.0.0.0.spds9: Nranges=4
-------------------------------------------------------------------------
+-Range 0
+----<Table1,ID>: sobs=1 eobs=25000 (Sorted)
+-------unique_keys=25000, max_occurance=1
+-------obs=25000, joinobs=25000, rangepct=100.00
+----<Table2,ID>: sobs=1 eobs=10000 (Sorted)
+-------unique_keys=10000, max_occurance=1
+-------obs=10000, joinobs=10000, rangepct=100.00
+-Range 1
+----<Table1,ID>: sobs=25001 eobs=50000 (Sorted)
+-------unique_keys=25000, max_occurance=1
+-------obs=25000, joinobs=25000, rangepct=100.00
+----<Table2,ID>: sobs=-1 eobs=0
+-------unique_keys=0, max_occurance=0
+-------obs=2, joinobs=0, rangepct=  0.00
+-Range 2
+----<Table1,ID>: sobs=50001 eobs=75000 (Sorted)
+-------unique_keys=25000, max_occurance=1
+-------obs=25000, joinobs=25000, rangepct=100.00
+----<Table2,ID>: sobs=-1 eobs=0
+-------unique_keys=0, max_occurance=0
+-------obs=2, joinobs=0, rangepct=  0.00
+-Range 3
+----<Table1,ID>: sobs=75001 eobs=100000 (Sorted)
+-------unique_keys=25000, max_occurance=1
+-------obs=25000, joinobs=25000, rangepct=100.00
+----<Table2,ID>: sobs=-1 eobs=0
+-------unique_keys=0, max_occurance=0
+-------obs=2, joinobs=0, rangepct=  0.00
```

```
Table Table1, Column ID average range join row pct=100.00
Table Table2, Column ID average range join row pct= 25.00

Ixutil completed successfully
```

Delete the Join Index

Use the **-deljidx** option of the ixutil command to delete the join index.

```
> ixutil -deljidx Table1,ID Table2,ID
  -libpath /tmp

SAS Scalable Performance Data Server 4.40(TS M0-WIDE)
Build(Aug  2 2006, 15:45:44)  Index File Utility
Copyright (c) 1996-2006 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, etc. 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.

SAS Scalable Performance Data Server Backup and Restore Utilities

---

SAS SPD Server Backup and Restore Utilities

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 provided by native operating systems are generally inadequate for 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 modifications 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

- **spdsls**
  Lists all the tables in an SPD Server domain. Or, if you specify an SPD Server table component file, lists all other components of that table.

- **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 **spdsbkup** utility.

Using Utilities with SPD Server

SPD Server provides NLS functionality, or National Language Support for multiple languages and character sets in database operations. As a result, all SPD Server utilities require access to the *InstallDir*/**bin64** directory, and you must ensure that the *InstallDir*/**bin64** directory is included in your SPD Server PATH.

Here is an example of statement that specifies the necessary path:

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:InstallDir/bin64

export LD_LIBRARY_PATH
```

Compatibility with Previous Versions

Scalable Performance Data Server backup and restore utilities are not compatible with SPD Server 3.x and earlier releases. Therefore, you cannot restore existing backup files with 4.4 utilities. You must use your 3.x utilities to restore 3.x backup files, and then archive them using the 4.4 utilities.

Privileged Access Protection

Running the SPD Server backup and restores utilities is a privileged operation. For appropriate access to the utilities, one of the following must be true:

- The user ID that starts the SPD Server must be identical to the user ID that is used by the SPD Server backup and restore utilities (by definition the user ID that starts the SPD Server is a privileged user).
- A user must have ACLSPECIAL=YES privileges.

To run the backup and restore utilities remotely using the SAS PROC SPDO *spdscmd* command requires similar access: either identical user IDs or ACLSPECIAL=YES user privileges. See Usage Scenarios for more information on 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.

Spdsls - the Table List Utility

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 two 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 that may be affected.

Usage

```
spdsls -l [-i] [-o] [-a] [-s] [-v] [-v8] [-v6] [-aonly] <libpath> [Table. . .]
spdsls -c [-i] [-o] [-a] [-v] <ComponentPath>
spdsls -info [-o] [-v] [-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 which may 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 which may 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.

-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.

Spdsbkup - the 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 - Full Backup

- When you do a full backup of a table, all 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 to the attributes since the full backup 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 printed. spdsbkup, meanwhile, will continue to back up any specified tables.

Important Details - Incremental Backup

- When you do an incremental backup of a table, only the changes to the rows made since the last full backup are included in the incremental backup. Any changes to the attributes in the table are not backed up. When an incremental backup is restored, the incremental changes to the rows are applied. Any attributes for the table at the time of the restore (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 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 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 Scalable Performance Data Server LIBNAME domain with the "BACKUP=YES" option in the parameter file. To explain the processing of a domain, created with or without the BACKUP=option, here are two example LIBNAME entries from the data's server's libname parameter file, libnames.parm:

Example Entries in libnames.parm

```
libname=nobackup pathname=/usr/foo/test;
libname=canbackup pathname=/usr/foo/test BACKUP=YES;
```

Notice that the entry for the LIBNAME domain *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, it backs up only the eligible tables in *canbackup*. On the client connection with pass-through or libname, the BACKUP=NO libname option can be used to override. Tables created when either of these is in effect will be able to be backed up.

An Incremental Backup 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, then inform spdsbkup of the system's last full backup date.
- Use spdsbkup for the full backup if none has been done before.

After a full backup is done for the table, you can then proceed with an incremental backup. The first incremental backup saves any table changes subsequent to the last full backup date. And, each successive incremental backup saves any changes made subsequent to the previous incremental backup date.

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 full backup, it performs a full backup of the table and sets the last full backup date.
- If the table DOES have a full backup, it performs an incremental backup using the later of the two dates: the last full backup date or the last incremental backup date 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 only necessary 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 will 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 4.4 can only backup SPD Server and SPD Server data sets. Attempting to backup earlier versions of SPD Server data sets will result 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 being updated cannot be backed up. Spdsbkup 4.4 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(s) which 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 it encounters a table that was created before the specified date/time (that is, it is in the last full system backup), its sets the last full backup date for the table to the specified time and performs an incremental backup of the table from 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 tables in the domain to be included in the backup. If no tables are specified, all eligible tables in the domain are backed up.
Note: The tables must be the last option specified.

Backup Return Values

When **spdbkup**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 Filename

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 done. 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 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. (SPD Server file extension is the "0.0.0" portion of the name.) While the extensions for the files will be different, the date/time will be the same on all the files.

For a restore to be successful, the backup file with any extensions must be available.

The 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 created by the same SPD Server operation, the timestamp for the backup file and the table of

contents file are identical. But, 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 by the system's file limit.

The table of contents file contains the following information for each backed up table:

- Columns 1 — 32 contain the table name or ".ACLs" if it is a domain ACL file.
- Columns 33 — 232 contain the backup filename.
- Columns 233 — 250 contain the last full backup date in 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 an "F" if this is a full SPD Server backup or an "I" if this is an incremental backup.
- Columns 270 — 277 contain the number of indexes backed up for a full SPD Server backup. This field contains a 0 if this is an incremental backup.
- Column 278 contains a "T" if this is a domain ACL file and an "F" if it is not (it's a normal table). If "T", also sets columns1-32 to ". ACLs".

The table of contents file is formatted so that it can be used as a SAS backup file table of contents table. 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 acls $1.;
```

After performing each SPD Server backup, you should append the resulting table of contents file to the SAS backup file table of contents table. This is an **important** step because it saves the backup history and assists later in restoring the tables.

If, for example, you need to determine which backup files are necessary to restore a specific table from a given last full backup 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

There are three basic categories of backup user messages:

- 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 *-q* option is specified. The notes include useful information, such as the name of the table backed up, the number of observations backed up, and whether the backup was full or incremental.

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 abort 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 from when the full backup occurred and then all of the rows are added.

Restore Requirements

Before **spdsrstr** restores a table, it must meet the following criteria:

- 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.
- The 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 extensions must be available.

If a table does not meet all of the criteria, spdsrstr prints a warning message to stdout, and it does not restore the table. If spdsrstr is restoring multiple tables, it will restore only those 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 <passwrd>] [-proj <dir>] [table ...]

spdsrstr -v -d <dom> -h <host> {-f <fullfile> | -e <extfile>}
[-s <serv>] [-u <user>] [-p <passwrd>] [-proj <dir>] [table ...]

spdsrstr -t {-f <fullfile> | -e <extfile>} [table...]

spdsrstr -help
```

spdsrstr
Restores all or selected tables from a backup file.

spdsrstr -t
Prints a table of contents for a backup file 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 the domain ACL (Access Control List) files are consistent with the last file restored.

-aonly
Only restore the domain ACL (Access Control List) files, 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()) to restore ALL backup files in the directory with the name "<*extfile*>_BK_*ddmmmyyyy_hhmmss*.0.0.0.spds" will be restored in the order of the oldest to the newest backup file 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

The number of times spdsrstr will retry accessing tables which are not available 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 this 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 tables to restore from the backup file. If no tables are specified, all tables in the file are restored.

Note: The tables **must** be the last option specified.

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

There are three basic categories of User Messages

- For a Successful Restore
- For a Table That Cannot Be Restored
- A Failed Restore

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 restored, the number of rows restored, and whether the restore was full or incremental.

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 abort and print an error message that states the reason for the failure.

Restore Usage Examples

This section presents common examples for the SPD Server backup and restore utilities. In our example environment, the starting date for the backup cycle is Sunday, February 5, 2006. 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.

Example 1: Exclusive SPD Server Full and Incremental Backups

You can use SPD Server backup/restore utilities exclusively to perform full and incremental table backups and restores.

This example outlines the steps required to use the SPD Server utilities to perform a full backup of the domain once a week and incremental backups the rest of the week. The *incremental* backups will also fully back up any newly-created tables.

1. On Sunday, February 5, 2006 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_05Feb2006_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 backed up table.

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 and a full backup of tables that were created after the previous night's backup and will also back up any ACL files 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 you have archived previously and to perform full backups on new data sets that have never been backed up. You also can back up your SPD Server data sets using a system utility from your operating environment. Which one is best to 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 on 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 file is usually backed up to tape.

This example outlines the steps required to use system utilities to perform a full system backup of the domain "test" once a week, then use SPD Server to back up the domain on the remaining nights:
.

1. On Sunday, February 5, 2006 at 23:30, run the SPD Server "list" (spdsls -l) utility to get a listing of the "test" domain tables in preparation for a full backup.

   ```
   spdsls -l -a physical_path_of_domain
   ```

   Run the system backup facility to get a full backup of the domain tables and ACL files, then archive the backup.

2. On Monday, February 6, 2004 at 23:30, run the SPD Server backup utility to set the last full backup date to the previous night for the 'test' tables. The utility then performs an incremental backup of tables that have changed since the last full system backup, and performs a full backup of tables created after the last full system backup date.

```
spdsbkup -d test -h host -s serv -t 02/06/04:23:30:00 -f backup
```

The utility creates the backup data file backup_BK_06Feb2006_233000.0.0.0.spds and a backup table of contents file backup_TC_06Feb2006_233000.

The backup file contains: *incremental* changes for tables that were modified after 02/08/2004 23:30:00, and *full* backups of tables created after 02/06/2006 23:30:00. 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

The following are the steps required to restore a table which was accidentally deleted from the domain "test" on Friday, 02/10/2006.

1. If the table was backed up fully by the system backup utility, use the system restore utility to restore the table. (Restore the table to its last full backup state on February 5, 2006.) If the table was backed up fully by the SPD Server backup utility, skip this step.
2. Run a SAS query on the backup table of contents table bkup_toc.

```
select bk_file from foo.bkup_toc where domain = "test" and table =
    "results" and dttime >= '05Feb2006:23:30:00'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 SPD Server backup file to restore the table in order, that is, from the oldest date to the most recent backup file date. Our example table was backed up fully with the SPD Server backup utility on Sunday, February 5. The table was then backed up incrementally on Tuesday, February 7 and Thursday, February 9:

Thus, the order of the statements required to restore the table are

```
spdsrstr -d test -h host -s serv -f backup_BK_05Feb2006_233000.0.0.0.spds results
spdsrstr -d test -h host-s serv -f backup_BK_7Feb2006_233000.0.0.0.spds results
spdsrstr -d test -h host -s serv -f backup_BK_09Feb2006_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 restoring a single table, you do not want to restore the ACL files, since they were not deleted.

Example 4: Restoring an SPD Server Domain

The following lists steps that are necessary to restore the domain named "test". The domain named "test" was lost due to a system media failure that occurred on Friday, February 10, 2006.

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 5, 2006. If the domain was backed up fully using the SPD Server utility, then skip this step.
2. Use SAS to run a query on the backup table of contents table bkup_toc.

```
select bk_file from foo.bkup_toc where domain = "test" and dttime >=
   '05FEB2006:23:30:00'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 spdscmd to run the SPD Server backup and restore utilities. There is one constraint: you must use it with an SPD Server LIBNAME that has 'special' privileges. To grant 'special', privileges, you must specify the LIBNAME option ACLSPECIAL=YES. (Backup and restore utilities require privileged access.)

When you execute commands using the PROC SPDO spdscmd, the current working directory (default pathname) is the home directory of SPD Server. Output messages from the commands are echoed to the SAS log. In the example that follows, the SPD Server *incremental* backup and restore utilities reside in the SPD Server directory. And, 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. Aforce will fail if there are any active connections to the domain specified with the *-d* option on the restore, if ACLs currently exist. Also, note that the ACLSPECIAL= libname connection will need to be a different domain from the domain where you are attempting to restore the ACLs if the ACLs currently exist, as this will cause the ACL restore to fail.

Steps required to use PROC SPDO to execute SPD Server backup and restore utilities include the following:

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's remote system command capability to issue backup and restore commands on the server. The following example performs a full SPD Server backup of the domain "tstdomn" at 23:30 on Feb 5, 2006.

```
spdscmd 'spdsbkup -a -full -d tstdomn -h sunny -s 5150 -f /spdsadm/bkup/test';
```

The example statement creates the backup file /spdsadm/bkup/test_BK_05Feb2006_233000.0.0.spds and the table of contents file / spdsadm/bkup/test_TC_05Feb2006_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.

```
spdscmd 'spdsrstr -aforce -d tstdomn -h sunny -s 5150 -e /spdsadm/bkup/test
```

Backup and Restore of Table Indexes with SPD Server Full Backups

When you perform an SPD Server full backup of a table, by default the utility saves information to re-create the indexes. This information is subsequently used if the table is fully restored, to be able to re-create the indexes.

The SPD Server full backup utility does not save the index data, only the information necessary to re-create 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 re-create 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 re-create the indexes. In some cases, this may provide better performance at the cost of having to manually re-create the table indexes.

Backup and Restore of SPD Server Table Indexes Using System Full Backups

Restoring indexes from system *full* backup 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:

- restore the index dynamically as the table is being restored
- re-create 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 - Restoring the Index Dynamically

To use this method you must include the table index files in the *full* backup and restore of the table. To determine which index files to include, use spdsls with the *-i* index option. The output lists component files for each table in the domain that is 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 may be required to restore them. (You can restore indexes for a table that has no *incremental* changes after the system full backup with a system *full* restore.)

Method 2 - Re-creating the Index after the Table Is Restored

If you use this method, you do not need to include the index files in the 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 which may 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.

As you can see, the second method trades the resources saved from not fully backing up the index files against the relatively long time it may take to re-create the indexes fully if the table must be restored.

SAS Scalable Performance Data Server Directory Cleanup Utility

---

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 *only* be used 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 a '**?**' characters in the search pattern will wildcard match any single character in a LIBNAME domain name in the libname file.
- The '*****' character terminates the pattern and wildcard matches 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`', etc., 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:

- Options That Define Actions
- Options That Modify Action Behavior.

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 Action Behavior.

-libnamefile *libnameFile*
The *-libnamefile* option runs cleanup on the SPD Server environment specified in *libnameFile,*the LIBNAME parameter file. The cleanup action empties directory resources that are defined in LIBNAME statements in the specified LIBNAME parameter file. Cleaning directory resources removes files and file types that you specify in this action. Spdsclean always deletes residual lock files that were left behind in the domain directory. Residual temporary files in the allocated domain directories are deleted by default as well. You can include ACL files and the LIBNAME state file in the files to be deleted, as well as suppress the default deletion of residual temporary files. Use the *-domains* option with pattern matching to filter the domains that you want to clean in the *libnameFile*. See the description for the *-domains* option below for more information.

-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 *.spdslog* files. When *.spdslog* files are found, SPD Server checks them for aging criteria. You specify the aging criteria, which tells SPD Server how long to keep the log files using the *-logage* option. When spdsclean finds server log files that have a creation date that is older than *-logage* days, spdsclean deletes the files. Files aged less than or equal to the specified threshold are retained. See the *-logage* option for more details and default values.

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 which build 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 7 days.

Spdsclean Examples

For the following examples, assume that the *InstallDir/* for your SPD Server is the directory **`/opt/spds44`**.

Cleaning WORKPATH Files on Your Server

This spdsclean command cleans all of the files in the WORKPATH directory list designated by **`/opt/spds44/site/spdsserv.parm`**.

```
spdsclean -parmfile /opt/spds44/site/spdsserv.parm
```

If you want spdsclean to log the files it deletes, add the *-verbose* option to the command.

```
spdsclean -parmfile /opt/spds44/site/spdsserv.parm -verbose
```

Cleaning Residual Temporary LIBNAME Domain Files

This spdsclean command cleans all of the residual temporary files from all of the LIBNAME domains that are defined in the *-libnamefile* specified.

```
spdsclean -libnamefile /opt/spds44/site/libnames.parm
```

If you want spdsclean to log the files it deletes, just add the *-verbose* option to the command.

```
spdsclean -libnamefile /opt/spds44/site/libnames.parm -verbose
```

Cleaning Specific LIBNAME Domains

This spdsclean command cleans all residual temporary files from the LIBNAME domain TRIAL99.

```
spdsclean -libnamefile /opt/spds44/site/libnames.parm -domains trial99
```

Suppose that you want to add domain UJOE04 to be cleaned also. The following command will do this:

```
spdsclean -libnamefile /opt/spds44/site/libnames.parm -domains trial99, ujoe04
```

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:

```
spdsclean -libnamefile /opt/spds44/site/libnames.parm -domains trial9?, ujoe*
```

To log the domains processed and the files deleted from each, just add the *-verbose* option to any of these spdsclean commands.

Cleaning Other LIBNAME Domain File Classes

This spdsclean 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.

```
spdsclean -libnamefile /opt/spds44/site/libnames.parm +tmp -acl -domains ujoe*
```

To clean domain state files from domains TRIAL9x for the specified *-libnamefile*. submit the following spdsclean command:

```
spdsclean -libnamefile /opt/spds44/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 spdsclean 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*.

```
spdsclean -parmfile /opt/spds44/site/spdsserv.parm
          -libnamefile /opt/spds44/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 spdsclean command cleans the *.spdslog* files from the specified *-logdir* directory that are more than 7 days old.

```
spdsclean -logdir /opt/spds44/log
```

Suppose you want to keep log files older than 10 days from the date of execution. The following spdsclean command will do this:

```
spdsclean -logdir /opt/spds44/log -logage 10
```

If you want to see the files deleted, add the *-verbose* option to the spdsclean command.

Cleaning WORKPATH, LIBNAME Domain and Log Files

This spdsclean 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 7 days from the *-logdir* directory.

```
spdsclean -parmfile /opt/spds44/site/spdsserv.parm
          -libnamefile /opt/spds44/site/libnames.parm
          -logdir /opt/spds44/log -verbose
```

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.parm File

The *libnames.parm* file defines the SPD Server LIBNAME domains for the SPD Server environment. The *libnames.parm* file is a collection of LIBNAME statements. Each LIBNAME statement defines a storage domain that SPD Server uses with clients. You modify the *libnames.parm* file 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.parm File

The spdsserv.parm 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.parm file, use the spdsserv command with the -parmfile option.

System-Specific Temporary Files

SPD Server uses pre-assigned directories (which vary by operating environment) that are designated for temporary files. The pre-assigned directories hold files, logs, and other temporary entities that SPD Server creates while running. SPD Server normally cleans up these temporary files when exiting. If SPD Server terminates abnormally, these temporary files may 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).

SAS Scalable Performance Data Server Debugging Tools

Introduction

SPD Server includes debugging tools that system administrators will find useful. The debugging tools will 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 4.4 running on SAS 9.1.3. The debugging tools are organized into LIBNAME statement options for debugging, and server parameter file options for debugging.

SPD Server 4.4 LIBNAME Statement Debug Options

When you issue a LIBNAME statement in SPD Server 4.4, the following debug options are available:

DEBUG=

Enables or disables launching of a debug image for base SPD Server (spdsbased) and SPD Engine (spdsengd).

Syntax

```
DEBUG= YES | CORE | NO
```

Use the following arguments:

YES
launches debug image (*spdsbased* and *spdsengd*) which will enable a good traceback.

CORE
launches debug images and sets the SPD Server parameter file option COREFILE for that proxy. This ensures a good core file.

NO
disables the debug image for the specified proxy, if one is present.

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=

Enables the use of an alternate binary path that is defined using the ALTBINPATH= option in the **spdsserv.parm** 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 **spdsserv.parm** 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 (e.g., testing a fix) without having to replace the production copy of SPD Server on a user basis.

Syntax

```
ALTPATH= YES | NO
```

Use the following arguments:

YES
: enables use of the alternate binary path that is defined in **spdsserv.parm** as ALTBINPATH=.

NO
: disables the the alternate binary path for the specified proxy, if one is present.

Example

Issue a LIBNAME proxy that uses the alternate binary path that is defined in **spdsserv.parm**:

```
libname mylib sasspds 'spdsdata' user='denettee' altpath=y;
```

SPD Server 4.4 Server Parameter File Debug Options

SPD Server includes new server parameter file options that are designed to be troubleshooting and debugging tools for system administrators :

- ALTBINPATH=
- RECORDFLAGS=

ALTBINPATH=

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. The ALTBINPATH= server parameter file option is enabled when a LIBNAME statement that contains the ALTPATH=Y option is issued.

ALTBINPATH= '*DirPath*'

RECORDFLAGS=

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. The option values 1, 2, and 4 are bit flags. 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).

Syntax

RECORDFLAGS= 1 | 2 | 4

Use the following arguments:

1

launches 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.

Examples

Use the RECORDFLAGS= option to launch a debug image for the SPD Server record locking process, and to produce a core file if an unexpected exception is encountered while running:

```
RECORDFLAGS= 5
```

Use the RECORDFLAGS= option to launch a debug image for the SPD Server record locking process, to load the image specified in ALTBINPATH=, and to produce a core file if an unexpected exception is encountered while running:

```
RECORDFLAGS= 7
```

SPD Server NLS Support

Contents

Overview of NLS

NLS, or National Language Support, deals both with *Internationalization* and *Localization* of SAS software. *Internationalization* is the process of designing an application so that it can be adapted to different languages and regions, without requiring engineering changes. Often the term internationalization is abbreviated as **i18n**, because there are 18 letters between the first "i" and the last "n." *Localization* is the process of adapting software for a particular region or language by adding locale-specific components and translating text. The term localization is often abbreviated as **L10n**, because there are 10 letters between the "L" and the "n." Translation of user interface, messages, and documentation is a large part (but not all) of localization. Localizers also verify that the formatting of dates, numbers, currencies etc. conforms to local requirements.

SAS 9 contains built-in support for NLS character set encoding and locale choices. Users access the NLS encoding and locale choices through various SAS, LIBNAME, and data set options. SPD Server 4.4 and SAS 9.1.3 together offer an experimental level of NLS support. This document describes the basic entities of NLS support and how they are implemented in SPD Server 4.4

Character Encoding Overview

All input to a computer is represented internally as numbers. The computer assigns a number to each character -- technically, the number is a binary number (base 2 numbering system, consisting of 0s and 1s).

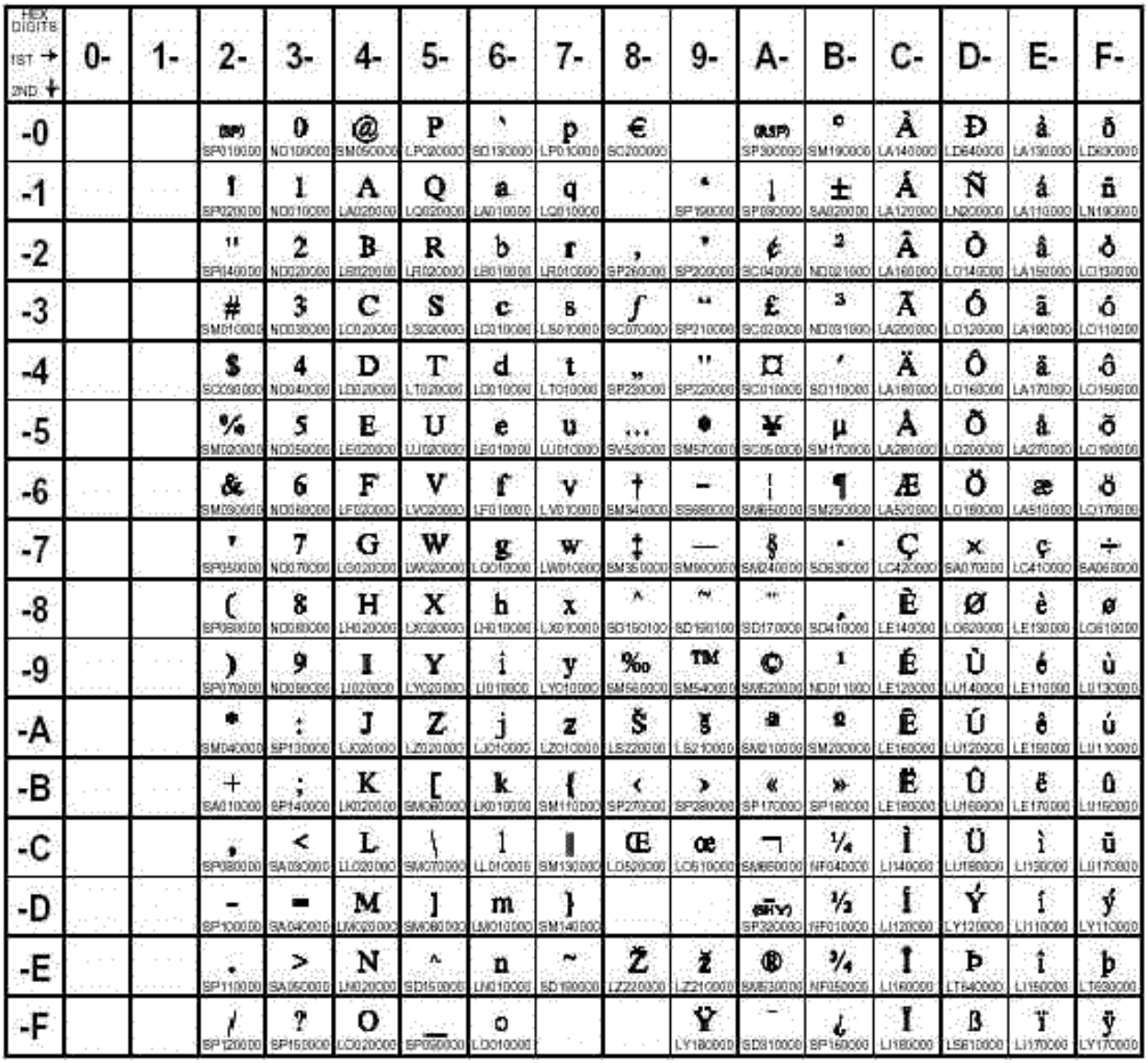
Because most of us don't want to think in binary numbers, computers provide hexadecimal (base 16 numbering system) representation as a shorthand for binary representation. For example, for the decimal number 167, it's easier to understand the hexadecimal number A7 than the equivalent binary number 10100111. Therefore, you can think of the computer's internal numeric representation of all data as a hexadecimal number.

What is Character Encoding?

All data that is stored, transmitted, or processed by a computer is in an encoding. An *encoding* maps each character to a unique numeric representation. For example:

1. You press a key on a keyboard, like the uppercase letter A.
2. The computer assigns the internal numeric representation, that is, a unique hexadecimal number.
3. To display or print the character, the computer uses the font (graphical representation) that matches the numeric representation, that is, the uppercase letter A.

To assign the numeric representation to a character, an encoding uses a *code page*, which is an ordered set of characters in which a numeric index (code point value) is associated with each character. The position of a character on the code page determines its two-digit hexadecimal number. The first digit of the hexadecimal number is determined by the column, and the second digit by the row. For example, the following is the code page for the Windows Latin1 encoding. The numeric representation for the uppercase A is the hexadecimal number 41, and the numeric representation for the equal sign (=) is the hexadecimal number 3D.



Encoding is the combination of a character set with an encoding method:

- A *character set* is the repertoire of characters and symbols that are used by a language or group of languages. A character set includes national characters (which are characters specific to a particular nation or group of nations), special characters (such as punctuation marks), the unaccented Latin characters A-Z, the digits 0-9, and control characters that are needed by the computer.

- An *encoding method* is the set of rules that are used to assign the numbers to the set of characters that are in an encoding. These rules govern such things as the size of the encoding (number of bits used to store the numeric representation of the character) and the ranges in the code page where characters are allowed to appear.

When the rules of the encoding method are followed, and numbers are assigned to the characters, the result is called an encoding.

An individual character can have different positions in code pages for different encodings, which result in different hexadecimal numbers. For example, the position of the uppercase

letter A in the Wlatin1 code page (shown above) results in the hexadecimal number 41, while in the following Danish EBCDIC code page, the position of the uppercase letter A results in the hexadecimal number C1.

HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM050000	- SP100000	| SM650000	@ SM080000	° SM100000	µ SM170000	¢ SC040000	æ LA610000	å LA270000	\ SM070000	0 ND100000
-1	(RSP) SP800000	é LE110000	/ SP120000	É LE120000	a LA010000	j LJ010000	ü LU170000	£ SC020000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC060000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ä LA170000	ë LE170000	Ä LA180000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	à LA130000	è LE130000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	§ SM240000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ã LA190000	î LI150000	Ã LA200000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	} SM140000	ï LI170000	$ SC030000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ç LC410000	ì LI130000	Ç LC420000	Ì LI140000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	ß LS610000	Ñ LN200000	` SD130000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	# SM010000	€ SC200000	ø LO610000	: SP130000	« SP170000	ª SM210000	¡ SP030000	¬ SM660000	(SHY) SP920000	¹ NO011000	² ND021000	³ ND031000
-B	. SP110000	Å LA280000	, SP080000	Æ LA620000	» SP180000	º SM200000	¿ SP160000	‖ SM130000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA080000	* SM040000	% SM020000	Ø LO620000	ô LO630000	{ SM110000	Ð LO620000	¯ SM150000	ö LO170000	~ SD100000	Ö LO180000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ý LY110000	‚ SD410000	Ý LY120000	" SD170000	ò LO130000	ù LU130000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	þ LT630000	[SM060000	Þ LT640000	´ SD110000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	! SP020000	^ SD150000	? SP150000	" SP040000	± SA020000	] SM080000	® SM590000	× SA070000	õ LO190000	ÿ LY170000	Õ LO200000	(EO)

Common Encodings

There are many encodings that address the requirements of different languages. Very few languages use only the 26 characters A through Z of the Latin alphabet. In addition, there are different encodings to address different operating system standards.

An encoding that represents each character in one byte is a single-byte character set (SBCS). A single-byte character set can be either 7 bits (providing up to 128 characters) or 8 bits (providing up to 256 characters). An example of an 8-bit SBCS is the Latin1 encoding (represents the characters of Western Europe). (Note that the term *octet*, for the international community, is an 8-bit byte. Since a byte is not 8 bits in all computer systems, octet provides an unambiguous term.)

A multiple-byte character set (MBCS) is a mixed-width encoding in which some characters consist of more than one byte. For example, the Japanese, Korean, Simplified Chinese, and Traditional Chinese are MBCS encodings. A double-byte character set (DBCS) is a specific type of a MBCS encoding that includes characters that consist of two bytes.

The following are common encodings:

ASCII (American Standard Code for Information Interchange)
: is a 7-bit encoding for the United States that provides 128 character combinations. The encoding contains characters for uppercase and lowercase English, American English punctuation, base 10 numbers, and a few control characters. The set of 128 characters is the one common denominator that is contained in most encodings, excluding EBCDIC-based encodings. ASCII is used by personal computers.

ISO (International Organization for Standardization) 646 family
: is a 7-bit encoding that is an international standard and provides 128 character combinations. The ISO 646 family of encodings is like ASCII except for 12 code points for national variants. The 12 national variants represent specific characters needed for a particular language.

EBCDIC (Extended Binary Coded Decimal Interchange Code) family
: is an 8-bit encoding that provides 256 character combinations. There are multiple EBCDIC-based encodings. EBCDIC is used on IBM mainframes and most IBM midrange computers. EBCDIC follows ISO 646 conventions to facilitate translations between itself and 7-bit ASCII-based encodings. Characters A-Z and 0-9 are mapped to the same code points on all EBCDIC code pages, while the rest of the code points may be used for special characters and national characters, depending on the encoding.

ISO 8859 family and Windows family
: is an 8-bit extension of ASCII that supports all of the ASCII code points and adds 12 more, providing 256 character combinations. Latin1, which is officially named ISO-8859-1, is the most frequently used member of the ISO 8859 family of encodings. In addition to the ASCII characters, Latin1 contains accented characters, other letters needed for languages of Western Europe, and some special characters.

Unicode
: uses two bytes for each character rather than one and provides up to 65,536 character combinations. Unicode can handle the scripts of basically all of the world's languages. For example, the Japanese language, which has thousands of characters, uses a 16-bit, multiple-byte character set. There are various forms of Unicode, including UTF-8, UTF-16, and UTF-32.

Moving Data across Environments with Different Encodings

Transcoding

Although it's easy to move data across environments that use the same encoding, it can be more difficult to move data across environments that use different encodings. When the encoding of a file is incompatible with the computer environment's encoding, transcoding occurs.

Transcoding is the process of mapping data from one encoding to another, for example, from an ASCII-based encoding to an EBCDIC-based encoding. Transcoding is not translating from one language to another; transcoding is *remapping of characters*.

For example, consider a file that was created on a UNIX platform that uses the Latin1 encoding, then moved to an IBM mainframe that uses the Danish EBCDIC encoding. When the file is processed on the IBM mainframe, the data is remapped from the Latin1 encoding to the Danish EBCDIC encoding. If the data contains a dollar sign ($), the hexadecimal number is converted from 24 to 67.

Transcoding can occur in the following situations:

- when you move a SAS file from one platform to another and the file's encoding is incompatible with the current session encoding, for example, from an z/OS operating environment with an EBCDIC-based encoding to a Windows operating environment with an ASCII-based encoding.
- when you share data between two SAS sessions (like in a client/server environment) that have incompatible session encodings.
- when you read and write an external file.

How Base SAS Transcodes Data

Base SAS provides transcoding when you move data and applications from one environment to another. To transcode one encoding to another, SAS uses translation tables, like the one that maps Wlatin2 (Windows) to ISO Latin2 (UNIX).

For example, when you

- use the CPORT and CIMPORT procedures to create a transport file, SAS automatically uses translation tables to transcode one encoding to another and back again. First, the data is converted from the source encoding to transport format, then the data is converted from the transport format to the target encoding.
- process a SAS data set that has an encoding that is different from the current session encoding, SAS automatically uses CEDA (cross environment data access) software to transcode data. (CEDA is the same software in SAS that converts a SAS file to the correct data representation when you move a file from one platform to another.)

Base SAS Encoding Behavior

For base SAS files (not SPD Server), the encoding support depends on the version of SAS that created the file:

- Data sets created in SAS 9 automatically have an encoding attribute, which is stamped in the descriptor portion of the file.
- Data sets created in SAS 8 do not have an encoding value stamped on the file; they are assumed to be in the session encoding of the host environment.

The NLS features in SPD Server only support encoding from SAS 9.

SAS 9 Output Processing

For SAS 9 data sets (not SPD Server), encoding is determined as follows:

- For a new output file, the data is written to the file using the current session encoding.
- For a new output file that is created with the OUTREP= option, which specifies a data representation different from the current session, the data is written to the file using the default session encoding for the operating system that is based on the specified OUTREP= value.
- For output processing that replaces an existing file, the new file inherits the encoding of the existing file.
- For output processing that replaces an existing file that is from another platform or if the existing file has no encoding stamped on it, then the current session encoding is used.

SAS 9 Input Processing

For input (read) processing in SAS 9 (not SPD Server), encoding behavior is as follows:

- If the session encoding and the encoding that is stamped on the file are incompatible, the data is transcoded to the session encoding. For example, if the current session encoding is ASCII and the encoding that is stamped on the file is EBCDIC, SAS transcodes the data from EBCDIC to ASCII.
- If a file does not have an encoding stamped on it, SAS transcodes the data only if the file's data representation is different from the current session.

Reading and Writing External Files

SAS reads and writes external files using the current session encoding. SAS assumes that

the external file is in the same encoding as the session encoding. For example, if you are creating a new SAS data set by reading an external file, SAS assumes that the file's encoding is the same as the session encoding. If it is not, the data could be written to the new SAS data set incorrectly.

Setting the Encoding for Base SAS Sessions

When SAS 9 is installed, the base SAS (not SPD Server) default encoding is host dependent and is determined by the default settings for several SAS system options. Here are three system options that you should be familiar with:

ENCODING=

establishes the session encoding, which is the encoding that SAS uses to process syntax, process SAS data sets, and read and write external files. The default value is host dependent; all are SBCS encodings:

Host	Value	Description
OpenVMS Alpha	Latin1	Western (ISO)
z/OS	OPEN_ED_1047	OpenEdition EBCDIC cp1047-Latin1
UNIX	Latin1	Western (ISO)
Windows	WLatin1	Western (Windows)

LOCALE=

specifies the locale of the SAS session. The locale reflects the local language, conventions, and culture for a particular geographical region. A locale's conventions may include the formatting of dates, times, and numbers, and printer preferences like paper size. Specifying a locale also automatically sets the default encoding that establishes the session encoding; a locale has a common encoding that is used most often for a particular operating environment. The default locale is English, and the common encodings for English are the defaults above for ENCODING=.

NONLSCOMPATMODE | NLSCOMPATMODE

provides national language compatibility for non-English data processing using native characters. For SAS 9, the default is NONLSCOMPATMODE, which provides consistency for running SAS on multiple systems. NONLSCOMPATMODE specifies that data is to be processed in the encoding that is set by the ENCODING= or LOCALE= system option.

Changing the Encoding for Base SAS Sessions

You can change the session encoding by using the LOCALE= system option, the ENCODING= system option, or both. Note that valid values for both options are host dependent.

Here's how you can set the base SAS (not SPD Server) session encoding when NONLSCOMPATMODE is specified:

- You can specify the LOCALE= system option in a configuration file, at SAS invocation, in an OPTIONS statement, or in the SAS System Options window. In SAS 9, several NLS-related system options are automatically set, based on the value of LOCALE=. Most customers will implicitly set encoding with the LOCALE= system option.
- You can specify the ENCODING= system option in a configuration file or at SAS invocation.
- Here is how LOCALE= and ENCODING= interact:
    - If a value is not specified for ENCODING= (that is, the installation default is set), then specifying a value for LOCALE= sets the encoding based on the LOCALE= value. In addition, values for the following system options are set based on the LOCALE= value: DFLANG=, TRANTAB=, DATESTYLE=, and PAPERSIZE=.
    - If a value is specified for ENCODING=, that value sets the session encoding and overrides LOCALE=.
    - If the value specified for LOCALE= is not compatible with the value specified for ENCODING=, then the value for LOCALE= is used. A warning message is provided if ENCODING= and LOCALE= conflict.
- If the DBCS system option is set, which specifies that SAS process DBCS encodings, the values for DBCSLANG= and DBCSTYPE= system options determine the session encoding and the locale. These options are used for Asian languages or for English with DBCS extensions.

Here is an example of implicitly setting the base SAS (not SPD Server) session encoding based on the specified locale when you invoke SAS:

```
sas9 -explorer -locale spanish
```

Here is an example of explicitly setting the base SAS (not SPD Server) session encoding with the OPTIONS statement:

```
options encoding=wlatin2;
```

Tip: Changing encoding for a SAS session does not affect SAS keywords, which remain in English, or SAS log output, which also remains in English.

NLS Support in SPD Server 4.4

SPD Server contains support for a subset of the SAS 9 NLS functions documented above. SPD Server utilizes encoding and locale currently only on SAS software.

In future releases of SPD Server, the locale identifier information will be used for locale-sensitive case folding and linguistic collation. Case-folding is defined as "a process applied to a sequence of

characters, in which those identified as non-uppercase are replaced by their uppercase equivalents". Linguistic collation is performing linguistic sorts based on linguistic sort keys. However, those functions have yet to be implemented in SPD Server production code.

All tables that are produced by SPD Server and SAS inherit the SAS session's default encoding and locale settings. By default, SPD Server code expects new tables to follow the current SAS session's encoding and locale. Table updates that append rows or update existing rows will perform transcoding to ensure that appended and updated table rows match the existing table encoding.

Wire transfer is in the character set encoding of the SAS session for transfers to and from the SPD Server host, unless SPD Server transcoding has been disabled. SPD Server transcoding is enabled or disabled by inserting a [NO]NLSTRANSCODE statement in the SPD Server spdsserv.parm parameter file.

SPD Server NLS Limitations

Affected Data

SPD Server hosts are restricted in the way they handle NLS character strings. SPD Server hosts are restricted to data that is contained in character columns in data sets and some metadata structures. The NLS support for SPD Server is functional for only table labels and variable labels.

Column names, index names, table names, and catalog names are *not* supported in the SPD Server NLS support. Column names, index names, table names, and catalog names are still dependent on ASCII support. SPD Server SQL is subject to the NLS same restrictions.

Pass-Through SQL

SPD Server pass-through SQL does not support any NLS functions. Pass-through SQL operates in the encoding and locale of the SAS session that initiates the CONNECT to SASSPDS.

Case Folding and Sort Sequences

SPD Server NLS code supports very limited English Latin1 and Polish Latin2 case folding for SBCS encodings. UTF8 case folding is limited to the ASCII range of UTF8 encoding. NLS Sort sequences for SPD Server 4.4 are restricted to lexical sorts for all combinations. Linguistic sorting is a subject for future SPD Server releases.

Indexes and Ordering

Indexes in SPD Server are created in the table's encoding, and only support lexical

ordering. If the client's encoding and locale settings match the SPD Server host table's encoding and locale settings, index use is unrestricted. Otherwise, index usage is restricted to certain predicates in WHERE clauses that can be safely interpreted according to the table's encoding and locale settings. When the client and host table encoding and locale settings differ, the EVAL2 strategy is used to filter predicates that require use of order.

Date and Time Representations

SPD Server server-side functions and formats that produce or accept textual date, time, and date/time representations are *not* locale-sensitive.

Suppressing Transcoding

You can suppress transcoding in the SPD Server environment by entering the following into the spdsserv.parm options:

```
NONLSTRANSCODE;
```

If you add the NONLSTRANSCODE option to your spdsserv.parm file, character transcoding between the SPD Server host and connected clients is disabled. Disabling character transcoding restricts the kinds of operations that the SPD Server host performs to operations it can safely perform, where host and client tables share the same encoding. Disabling SPD Server host transcoding assumes that the *client* will perform any needed transcoding on the data streams that it sends and receives to match the encoding of referenced tables. The SPD Server host setting for NONLSTRANSCODE does not perform any actions to deny client access to a host table that has mismatched encoding.

LIBNAME Option Restrictions:

The following options are not implemented in the SPD Server NLS functions:

The LIBNAME option

OUTENCODING=<*client-server encoding*>

is not supported and will produce a WARNING message if submitted to SASSPDS.

In addition, the related data set option

ENCODING=<*client-server encoding*>

is supported by the SAS LIBNAME engine for OUTPUT data sets only. Character data is assumed to be in the encoding of the session that initiates the CONNECT to SASSPDS and is normally stored using that encoding. ENCODING= will cause SPD Server to transcode from the SAS session encoding to the specified encoding for storing data. If you

specify ENCODING= for a non OUTPUT data set open, and if the encoding value that you specify doesn't match the data set's encoding, the data set open will produce a warning:

```
ENCODING= specified on table open fails to match table
encoding. Option ignored.
```

The LIBNAME option

```
TRUNCWARN=YES
```

Suppresses hard failure on NLS transcoding overflow and character mapping errors. When using the TRUNCWARN=YES LIBNAME option, data integrity may be compromised because significant characters can be lost in this configuration. The default setting is NO, which causes hard read/write stops when transcode overflow or mapping errors are encountered. When TRUNCWARN=YES, and an overflow or character mapping error occurs, a warning is posted to the SAS log at data set close time if overflow occurs, but the data overflow is lost.

Your Turn

We want your feedback.

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