

SAS[®] 9.3 Intelligence Platform Application Server Administration Guide

The correct bibliographic citation for this manual is as follows: SAS Institute Inc 2011. *SAS® SAS 9.3 Intelligence Platform: Application Server Administration Guide*. Cary, NC: SAS Institute Inc.

SAS® SAS 9.3 Intelligence Platform: Application Server Administration Guide

Copyright © 2011, SAS Institute Inc., Cary, NC, USA.

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

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

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

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

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

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

1st electronic book, July 2011

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

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

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

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

Contents

<i>What's New in Application Server Administration for the SAS 9.3 Intelligence Platform</i>	<i>vii</i>
<i>Recommended Reading</i>	<i>xi</i>

PART 1 Getting Started 1

Chapter 1 • Before You Begin	3
Introduction to This Guide	3
Accessibility Features in the SAS Intelligence Platform Products	4
Chapter 2 • Understanding the SAS Application Server	5
Overview of SAS Application Servers	5
The Structure of a SAS Application Server	6

PART 2 Server Concepts 9

Chapter 3 • Understanding Workspace Servers and Stored Process Servers	11
Overview of Workspace Servers and Stored Process Servers	11
SAS Object Spawners	13
Chapter 4 • Understanding SAS/CONNECT Servers	15
Overview of SAS/CONNECT and the SAS Intelligence Platform	15
Introduction to SAS/CONNECT	16
The Uses of SAS/CONNECT in the SAS Intelligence Platform	17
Initial Configuration of the SAS/CONNECT Server	19
Chapter 5 • Understanding the Batch Servers	23
Overview of SAS Batch Servers	23
The SAS DATA Step Batch Server	24
The SAS Java Batch Server	25
Additional Information	26
Chapter 6 • Understanding SAS Grid Servers	27
Overview of SAS Grid Servers	27
Overview of Grid Monitoring Servers	28
The Role of the SAS Grid Server in the SAS Intelligence Platform	28
The Initial Configuration of the SAS Grid Server	28

PART 3 Load Balancing and Pooling 31

Chapter 7 • Understanding Server Load Balancing	33
Overview of Load Balancing	34

Planning a Load-Balancing Cluster	39
Creating Metadata for Load-Balancing Clusters	40
Installing and Configuring Software for Load-Balancing Servers	41
Stopping and Restarting Load-Balancing Servers	44
Adding or Deleting Load-Balancing Servers	44
Understanding the Load-Balancing Algorithms	44
Chapter 8 • Understanding Server Pooling	55
Overview of Pooling	55
How Server-side Pooling Works	56
Understanding the Server-side Pooling Connection Process	56
How Client-side Pooling Works	57
Understanding the Client-side Pooling Connection Process	58
Chapter 9 • Configuring Client-side Pooling	61
Client-side Pooling Concepts and Overview	61
Configuring Client-side Pooling	63
Configure Client-side Pooling across Multiple Machines	69
Configuring a Client-side Pooling Workspace Server to Enforce Row-Level Security	72
 PART 4 Server Administration 79 	
Chapter 10 • Managing SAS Application Servers	81
Defining Multiple Application Servers	81
Add a New Logical Server in an Existing SAS Application Server	84
Adding a New Server in an Existing Logical Server	85
Modify a Server Definition	88
Remove Logical Servers	89
Chapter 11 • Managing Workspace Servers and Stored Process Servers	91
Managing Data and Catalogs for Servers on Multiple Machines	91
Adding or Modifying E-Mail Settings for SAS Application Servers	93
Moving Workspace Servers and Stored Process Servers	94
Encoding and Locale Information	97
Adding Environment Variables to Server Invocations	98
Run SAS Code at Server Session Boundaries	98
Workspace Server Configuration Tasks	99
Chapter 12 • Managing the Object Spawner	105
Object Spawner Configuration Tasks	105
Configuring and Starting the Object Spawner on z/OS	113
Spawner Invocation Options	117
Chapter 13 • Administering SAS OLAP Servers	125
Administrative Overview for SAS OLAP Servers	126
Migrating OLAP Cubes	127
Installing and Configuring SAS OLAP Servers	127
Connecting to SAS OLAP Servers	128
Starting SAS OLAP Servers	128
Stopping, Pausing, and Resuming SAS OLAP Servers	129
Disabling and Enabling Cubes	129
Building Cubes: Overview for Administrators	129
Updating Cubes: Overview for Administrators	130

Coalescing Cubes	131
Deleting Cubes	132
Authorizing Access to SAS OLAP Servers	132
Authorizing Access to OLAP Cubes and Cube Data	133
Monitoring SAS OLAP Servers	133
Managing OLAP Sessions and Queries	134
Logging SAS OLAP Servers	134
Tuning SAS OLAP Servers with Advanced Server Options	134
Refreshing Cube Metadata for Calculated Members and Named Sets	138
Administering OLAP Schemas	138
Chapter 14 • System Options for SAS Application Server Components	139
Overview of System Options for SAS Application Server Components	139
Dictionary	139
Chapter 15 • IOMOPERATE Procedure	153
Overview: IOMOPERATE Procedure	154
Concepts: IOMOPERATE Procedure	154
Syntax: IOMOPERATE Procedure	154
Using: IOMOPERATE Procedure	175
Examples: IOMOPERATE Procedure	175
PART 5 Appendixes 183	
Appendix 1 • Object Spawner and SAS OLAP Server Messages	185
Object Spawner Messages	185
Load Balancing Error Messages	199
Glossary	205
Index	213

What's New in Application Server Administration for the SAS 9.3 Intelligence Platform

Overview

The *SAS Intelligence Platform: Application Server Administration Guide* explains how to administer a SAS Application Server.

This document contains the following enhancements and changes to the SAS Intelligence Platform:

- “Added Object Spawner Command Option for Load-balancing Peer without a Peer Refresh” on page viii
- “Enhanced Support for Running SAS Code at Server Session Boundaries” on page viii
- “Enhanced Support for Running SAS Code at Server Boundaries” on page viii
- “Added Single Sign-on Support, Based on Kerberos, for UNIX” on page ix
- “Added Object Spawner Support for FIPS” on page ix
- “Changed Server Credentials for Load Balancing” on page ix
- “Added Support for Grid Algorithm” on page ix
- “Changed Object Spawner Refresh” on page ix

- [“Added New SAS Procedure: PROC IOMOPERATE”](#) on page ix

Added Object Spawner Command Option for Load-balancing Peer without a Peer Refresh

A new Object Spawner command option (-lbaddtocluster) enables you to add a new host to an existing load balancing peer object without requiring a peer refresh. This feature is required for cloud computing and software as a service models.

Enhanced Support for Running SAS Code at Server Session Boundaries

In addition to the stored process server, the workspace server and pooled workspace server now support running SAS code at server session start up and shutdown. For more information, see [“Run SAS Code at Server Session Boundaries”](#) on page 98.

Enhanced Support for Running SAS Code at Server Boundaries

In SAS 9.3, the IOM servers also support running SAS code at server start up and shutdown. For more information, see [“INITSTMT= System Option”](#) in *SAS System*

Options: Reference and “TERMSTMT= System Option” in *SAS System Options: Reference*.

Added Single Sign-on Support, Based on Kerberos, for UNIX

Single sign-on support based on Kerberos has been added for SAS servers running on UNIX. For more information, see [“SSPI System Option” on page 151](#).

Added Object Spawner Support for FIPS

The object spawner supports the Federal Information Processing Standards (FIPS) compliance mode that is provided by SAS/SECURE software in its implementation of the FIPS 140-2 specification. For more information, see [“-encryptfips” on page 119](#).

Changed Server Credentials for Load Balancing

Server load balancing generates its own credentials internally in SAS 9.3. Logical server credentials are no longer needed. For more information, see [Chapter 7, “Understanding Server Load Balancing,” on page 33](#).

Added Support for Grid Algorithm

The load-balancing grid algorithm is now supported for the OLAP, stored process, and pooled workspace servers. For more information, see [“Choose a Load-Balancing Algorithm” on page 39](#).

Changed Object Spawner Refresh

In SAS 9.3, when you refresh the object spawner, the spawner now quiesces any servers that it has started. The servers shut down when their clients have completed their work. For more information, see [“Refresh the Object Spawner” on page 110](#).

Added New SAS Procedure: PROC IOMOPERATE

There is a new SAS 9.3 procedure, PROC IOMOPERATE. It administers SAS servers that support the SAS IOM infrastructure. For more information [Chapter 15, “IOMOPERATE Procedure,” on page 154](#).

x *Application Server Administration*

Recommended Reading

- *Grid Computing in SAS*
- *SAS Guide to BI Row-Level Permissions*
- *SAS Intelligence Platform: Data Administration Guide*
- *SAS Intelligence Platform: Desktop Application Administration Guide*
- *SAS Intelligence Platform: Installation and Configuration Guide*
- *SAS Intelligence Platform: Middle-Tier Administration Guide*
- *SAS Intelligence Platform: Migration Guide*
- *SAS Intelligence Platform: Overview*
- *SAS Intelligence Platform: Security Administration Guide*
- *SAS Intelligence Platform: System Administration Guide*
- *SAS Intelligence Platform: Web Application Administration Guide*
- *Scheduling in SAS*
- SAS offers instructor-led training and self-paced e-learning courses to help you administer the SAS Intelligence Platform. For more information about the courses available, see support.sas.com/admintraining.

For a complete list of SAS publications, go to support.sas.com/bookstore. If you have questions about which titles you need, please contact a SAS Publishing Sales Representative:

SAS Publishing Sales
SAS Campus Drive
Cary, NC 27513-2414
Phone: 1-800-727-3228
Fax: 1-919-677-8166
E-mail: sasbook@sas.com
Web address: support.sas.com/bookstore

Part 1

Getting Started

<i>Chapter 1</i>	
Before You Begin	3
<i>Chapter 2</i>	
Understanding the SAS Application Server	5

Chapter 1

Before You Begin

Introduction to This Guide	3
Accessibility Features in the SAS Intelligence Platform Products	4

Introduction to This Guide

This guide covers the administration of the SAS Application Server, which is a logical entity that represents the SAS server tier in the SAS Intelligence Platform. This application server contains a set of actual servers. For example, a SAS Application Server usually contains the following servers:

Workspace Server

enables client applications to submit SAS code to a SAS session by using an application programming interface (API). For example, when you use SAS Data Integration Studio to submit an extract, transform, and load (ETL) job for processing, the application generates the SAS code necessary to perform the processing and submits it to a workspace server. You can run as many instances of workspace servers as are needed to support your workload.

Pooled Workspace Server

enables client applications to submit SAS code to a SAS session by using an application programming interface (API). Pooled workspace servers are workspace servers in every respect except that these servers automatically use pooling and load balancing.

Stored Process Server

interacts with SAS by submitting stored processes, which are SAS programs that are stored and can be executed by client applications. You can use stored processes to perform complex tasks such as analyzing data and creating reports, and then return the results to the client or publish the results to a channel or repository.

OLAP server

delivers pre-summarized, multidimensional data (cubes) to business intelligence applications. The data is queried using the multidimensional expressions language (MDX).

In addition, a SAS Application Server might contain one or more of the following servers:

SAS/CONNECT Server

enables clients to execute code on a remote host, or to move data between client and server machines.

SAS batch server

stores information in metadata about how to execute a SAS command in batch mode. A batch server is required if you are using the SAS scheduling system. There are three batch servers:

- DATA Step Batch Server
- Java Batch Server
- Generic Batch Server

SAS Grid Server

enables Platform LSF to start SAS/CONNECT servers on a SAS compute grid in order to execute grid-enabled SAS programs or grid-enabled jobs that are created in SAS Data Integration Studio and SAS Enterprise Miner.

This guide explains how to administer all of these server components and the SAS Application Server as a whole.

This guide assumes that you are familiar with the concepts and terminology that are introduced in the *SAS Intelligence Platform: Overview* document. For a list of all of the documents that SAS publishes to support administration of the SAS Intelligence Platform, see <http://support.sas.com/93administration>.

Accessibility Features in the SAS Intelligence Platform Products

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

Chapter 2

Understanding the SAS Application Server

Overview of SAS Application Servers	5
What are SAS Application Servers?	5
A Collection of Server Components	5
A Server Context	6
The Structure of a SAS Application Server	6
The SAS Application Server's Server Components	6
The SASMeta Application Server	8
The Hierarchy of Metadata Objects Used to Define a SAS Application Server	8

Overview of SAS Application Servers

What are SAS Application Servers?

When the SAS Intelligence Platform was installed at your site, a metadata object that represents the SAS server tier in your environment was defined. In the SAS Management Console interface, this type of object is called a SAS Application Server. By default, the application server object is named SASApp.

Note: In SAS deployments prior to SAS 9.2, the default SAS application server is named SASMain.

You can view the properties of this object by using the Server Manager plug-in to SAS Management Console. Expand the Server Manager tree node. Then right-click the **SASApp** node, and select **Properties** from the pop-up menu. You can also see the server components that make up the application server by completely expanding the SASApp node in the Server Manager tree.

In addition to this metadata object, a **SASApp** directory was created on each machine that hosts a SAS server (under the SAS configuration directory). This directory contains important files that you will use in the management of your SAS Application Server. In particular, it contains a file called **sasv9.cfg**, a configuration file that is used in the start-up of most SAS servers.

A Collection of Server Components

A SAS Application Server is not an actual server that can execute SAS code submitted by clients. Rather, it is a logical container for a set of application server components, which do execute code. Typically, these components execute SAS code, although some

components can execute Java code or MDX queries. For example, a SAS Application Server might contain a standard workspace server and a pooled workspace server, which can execute SAS code that is generated by clients such as SAS Data Integration Studio or SAS Web Report Studio. A SAS Application Server might also contain a stored process server, which executes SAS Stored Processes, and an OLAP server which executes and processes multidimensional expressions language (MDX) code to query a cube. If SAS runs on multiple machines, the SAS application might contain a SAS/CONNECT Server, which can upload or download data and execute SAS code submitted from a remote machine.

For a complete list of application server components, see [“The SAS Application Server's Server Components”](#) on page 6.

A Server Context

A SAS Application Server knows its server context (the context in which it is being used) and makes decisions based on that knowledge. For example, a client such as SAS Data Integration Studio is assigned a default SAS Application Server, and when the client generates code, it submits the code to that application server. The application server determines what type of code is being submitted and directs it to the correct server. That is, if the code is typical SAS code that could be run in the SAS Display Manager, the code is executed by the application server's workspace server.

In addition, data-related objects such as SAS libraries, database libraries, and OLAP schemas can be assigned to a SAS application server. Once this assignment is made, if a client needs to access data in a particular library or OLAP schema, it uses a server component belonging to the application server to which the library or schema has been assigned.

The Structure of a SAS Application Server

The SAS Application Server's Server Components

As mentioned in [“A Collection of Server Components”](#) on page 5, a SAS Application Server is a logical entity that encompasses a set of actual servers. Several types of servers might belong to a SAS Application Server, as shown in the following list:

- **SAS Workspace and SAS Pooled Workspace Servers** - These servers are provided with SAS Integration Technologies and are accessed through the Integrated Object Model (IOM) workspace interface. This interface provides access to Foundation SAS features such as the SAS language, SAS libraries, the server file system, results content, and formatting services. A SAS workspace represents a session with the SAS system and is functionally equivalent to the execution of the SAS System as a batch job. A pooled workspace server is a workspace server in every respect except that it automatically uses mechanisms—server-side pooling and load-balancing—to improve performance on larger SAS deployments. (For standard workspace servers, you must set up load balancing or client-side pooling manually.)
- **Stored Process Server** - The Stored Process Server is also part of SAS Integration Technologies. It retrieves SAS Stored Processes from a repository and executes them.

A stored process is a SAS program that is stored on a server and can be executed as required by requesting applications. You can use stored processes for Web reporting, analytics, building Web applications, delivering packages to clients or to the middle

tier, and publishing results to channels or repositories. Stored processes can also access any SAS data source or external file and create new data sets, files, or other data targets that are supported by SAS.

- SAS OLAP Server - Similar to the way in which a database management system (DBMS) can read an SQL query and return data from a database, the SAS OLAP Server processes MDX queries and returns data from OLAP cubes.

An OLAP server has a close relationship with a workspace server, and the two generally run on the same machine. The workspace server is used to build OLAP cubes, and the OLAP server is used to query the cubes.

- SAS/CONNECT Server - The SAS/CONNECT server has several general capabilities:
 - SAS/CONNECT provides compute services. A SAS/CONNECT client running on one machine can submit code to one or more remote SAS/CONNECT servers, which execute the code.
 - SAS/CONNECT provides Remote Library Services (RLS). These services enable SAS code to read, write, and update remote data as if it were resident on the client host. RLS can be used to access SAS data sets across machines that have different architectures.
 - SAS/CONNECT provides a set of Data Transfer Services (DTS). A SAS/CONNECT client can download data from a remote host where a SAS/CONNECT server is running, or the client can upload data to the server host. The client and server host do not need to be running the same operating system.

In addition, the SAS/CONNECT server has an important role in several features that are unique to the SAS Intelligence Platform:

- SAS/CONNECT servers run on all of the nodes in the compute grid. Together, the servers execute SAS Data Integration Studio and SAS Enterprise Miner jobs that use parallel algorithms.
- SAS Data Integration Studio can also use a SAS/CONNECT server for regular jobs. The application can generate code that uses the SAS/CONNECT server to upload data to a remote machine, download data from a remote machine, or execute the code for one or more transformations.
- batch servers - A batch server is actually a metadata object that stores a SAS command that is run in batch mode to execute SAS or Java code. The batch server contains an association between the stored command and the host on which it runs and possibly a log file. These SAS commands are actually scheduled jobs that are created with programs such as SAS Data Integration Studio and SAS Web Report Studio.

Different types of batch servers are available for different types of code: a SAS DATA Step Batch Server, a SAS Java Batch Server, and a SAS Generic Batch Server. For more information about these subtypes of the batch server, see [“Understanding the Batch Servers” on page 23](#).

- SAS Grid Server - The SAS Grid Server is similar to the batch server in that it stores a command. In this case, the server stores the command that Platform LSF will use to start SAS/CONNECT sessions on the nodes in the grid. For more information about the architecture of a system that supports grid computing, see [Grid Computing in SAS](#).

The SASMeta Application Server

The SAS Deployment Wizard creates a second SAS application server called SASMeta by default. The SASMeta application server is used for certain metadata functions such as backup and restore that depend on a SAS Workspace Server and (for some functions) a SAS/CONNECT Server. For more information about these metadata utilities, see “About Backups and Restores” in Chapter 11 of *SAS Intelligence Platform: System Administration Guide*.

In addition, the SASMeta server context also contains the SAS Metadata Server, although the metadata server is technically not an application server component.

The Hierarchy of Metadata Objects Used to Define a SAS Application Server

When your system was first installed, an application server was created when the first server—perhaps a workspace server—was defined. Defining the application server involved creating three objects:

- an application server
- a logical server (for example, a logical workspace server)
- a server (for example, a workspace server)

The Server Manager plug-in to SAS Management Console has a tree structure similar to the one shown in the following display:



The SASApp tree node represents the SAS Application Server. You assign resources such as libraries and OLAP schemas to this object. The result is that when an application such as SAS Web Report Studio needs to access a particular resource, it will use a server, such as a workspace server, that belongs to this application server.

The object named **SASApp - Logical Workspace Server** is a logical server. An application server such as SASApp can contain at most one logical server for each type of server that is listed in the section “[The SAS Application Server's Server Components](#)” on page 6. Generally, each logical server can contain one or more servers of the appropriate type. However, logical Grid Servers can contain only a single server.

The logical server level in the hierarchy enables you not only to group related servers together, but to control the behavior of the set of servers that belongs to the logical server. For example, if you have two workspace servers in a logical workspace server, you use the logical workspace server to indicate that you want to balance the workload that goes to these two servers. A logical server also gives you a place at which to use metadata access controls to secure all servers of a particular type in the same way.

The object named **SASApp - Workspace Server** represents the server that executes SAS code. In the case of a workspace server, this object contains information about the machine that the server runs on, the command that is used to start it, and the port on which it listens for requests.

Part 2

Server Concepts

<i>Chapter 3</i>	
<i>Understanding Workspace Servers and Stored Process Servers . . .</i>	<i>11</i>
<i>Chapter 4</i>	
<i>Understanding SAS/CONNECT Servers</i>	<i>15</i>
<i>Chapter 5</i>	
<i>Understanding the Batch Servers</i>	<i>23</i>
<i>Chapter 6</i>	
<i>Understanding SAS Grid Servers</i>	<i>27</i>

Chapter 3

Understanding Workspace Servers and Stored Process Servers

Overview of Workspace Servers and Stored Process Servers	11
What are Stored Process Servers and the Workspace Servers?	11
SAS Stored Process Servers	11
SAS Workspace Servers	12
SAS Pooled Workspace Servers	12
The Default Stored Process Server and the Workspace Servers	12
SAS Object Spawners	13
Overview of SAS Object Spawners	13
Configuration File for Metadata Connection	13
Spawner Tasks	14

Overview of Workspace Servers and Stored Process Servers

What are Stored Process Servers and the Workspace Servers?

Stored process servers and the workspace servers are crucial elements of the SAS Intelligence Platform that enable clients to perform SAS processing and to access Foundation SAS resources.

SAS Stored Process Servers

SAS Stored Process Servers interact with SAS by submitting stored processes, which are SAS programs that are stored and can be submitted by SAS client applications. You can use stored processes to perform complex tasks such as analyzing data and creating reports, and then returning the results to the client or publishing the results to a channel or repository.

Each stored process server process handles multiple users, and by default each server uses multiple server processes or instances. A load-balancing algorithm distributes client requests between the server processes. For more information about load balancing, see [“Overview of Load Balancing” on page 34](#).

If the job load for your stored process server is high, you might want to add additional server processes to your server definition. Each server process is defined as a MultiBridge connection in SAS Management Console.

SAS Workspace Servers

SAS Workspace Servers interact with SAS by creating a server process for each client connection. The workspace server process is owned by the client user who made the server request. Each workspace server process enables client programs to access SAS libraries, perform tasks by using the SAS language, and retrieve the results.

In the default configuration, SAS presents you with a workspace server that is not pooled (standard workspace server) and a workspace server that is pooled (pooled workspace server).

For the standard, non-pooled workspace server, SAS creates a new server process each time that a client requests a connection. For simple configurations and for sites that do not place heavy loads on a workspace server, a non-pooled server might be adequate. In situations where the demands on the workspace server are greater, you should (depending on which SAS application you are using) consider configuring the workspace server for either pooling (Web applications) or load balancing (desktop applications).

In a pooling configuration, a set of server processes are reused to avoid the processing time that is associated with starting a new process for each connection. SAS offers two types of pooling: client-based and server-based. A pooling configuration can also be shared across multiple machines. Pooling is recommended if your server supports SAS Web Report Studio and other Web applications. For more information, see [“Overview of Pooling” on page 55](#).

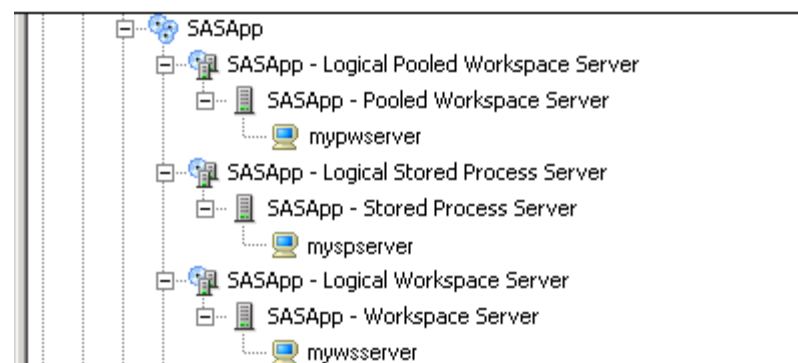
In a load-balancing configuration, your workspace server processes are distributed between multiple machines. Load balancing is recommended if your server supports applications that submit large jobs, such as SAS Data Integration Studio. For more information, see [“Overview of Load Balancing” on page 34](#).

SAS Pooled Workspace Servers

SAS Pooled Workspace Servers are workspace servers in every respect except that these servers automatically use pooling and load balancing. Like a standard workspace server, each pooled workspace server enables client programs to access SAS libraries, perform tasks by using the SAS language, and retrieve the results. For more information, see [“How Server-side Pooling Works” on page 56](#).

The Default Stored Process Server and the Workspace Servers

When the installer at your site runs the SAS Deployment Wizard, that person defines metadata for a SAS application server.



Usually, your application server contains a stored process server, a workspace server, and a pooled workspace server.

The initial stored process server is configured as a load-balancing server named by default **SASApp - Stored Process Server**. By default, the stored process server definition includes three MultiBridge connections.

The initial workspace server is configured as a standard workspace server named **SASApp - Workspace Server**.

The initial pooled workspace server is configured as a load-balanced, server-based pooled workspace server named by default **SASApp - Pooled Workspace Server**.

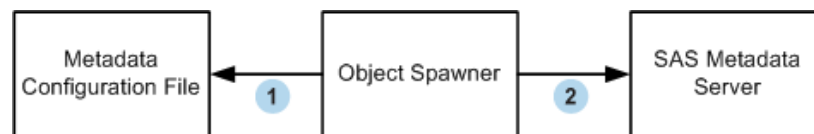
SAS Object Spawners

Overview of SAS Object Spawners

Workspace servers and stored process servers are initialized by the SAS Object Spawner. An object spawner runs on each machine where you want to run a workspace server or stored process server, listens for requests, and launches the servers as necessary.

The object spawner uses a configuration file that contains information for accessing the metadata server. When you invoke the spawner, the spawner works as shown in the following figure.

Figure 3.1 How the Spawner Obtains Metadata



- 1 The object spawner accesses a configuration file that contains information for accessing the SAS Metadata Server.
- 2 The object spawner connects to the SAS Metadata Server for configuration information.

(The spawner uses a proprietary SAS bridge protocol to communicate with the metadata server, and uses TCP for communication.)

The spawner can then listen for requests for various spawner tasks. See [“Spawner Tasks” on page 14](#).

Configuration File for Metadata Connection

A metadata configuration file contains information for accessing a metadata server. The spawner uses the information that is contained in the configuration file to connect to a metadata server and read the appropriate server definitions.

The default metadata configuration file is **metadataConfig.xml**. This file is located in the ObjectSpawner subdirectory of your SAS configuration directory.

Spawner Tasks

When a request is received, the spawner accepts the connection and performs the action that is associated with the port or service on which the connection was made. A connection to a spawner can do the following:

- request a server

When a connection is made on a port or service that is associated with a Server object, the spawner authenticates the client connection against either the metadata server (default for stored process and pooled workspace servers) or the host authentication provider for the server's machine (default for the standard workspace server). The spawner then launches a server for use by the connecting client.

When you define a server in SAS Management Console, you must specify a command that the spawner uses to start the server. When the object spawner starts, it reads all server definitions for which it is associated. Therefore, if you change or add a server start-up command, you must refresh (or restart) the object spawner for any modified commands to be used to launch a server. For details about the server command, see [“Add System Options to the Workspace Server Launch Command” on page 101](#). For SAS Stored Process Servers and SAS Pooled Workspace Servers, on the server definition, you must also configure credentials for the spawner to use to start a multi-user server. Every connection to the server is authenticated. Here are the authentication methods used for each type of spawned server:

- standard workspace servers

These servers start under the client user's credentials when set up for host authentication. (This is the default.) If metadata authentication is used (SAS Token Authentication), then standard workspace servers run under a configured user ID, referred to as launch credentials.

- stored process servers and pooled workspace servers

These servers start under a configured user ID, referred to as launch credentials. Typically, this user ID is the SAS Spawned Servers user (sassrv, by default).

- initiate the operator interface

When a connection is made on the port or service that is identified as the operator port or operator service in the spawner definition, the spawner initiates the administration interface. Only one administrator can be active at a given time.

- balance the server workload between server processes by using load balancing

See [“Overview of Load Balancing” on page 34](#).

Chapter 4

Understanding SAS/CONNECT Servers

Overview of SAS/CONNECT and the SAS Intelligence Platform	15
Introduction to SAS/CONNECT	16
Overview of Services	16
Compute Services	16
Data Transfer Services	16
Remote Library Services	17
The Uses of SAS/CONNECT in the SAS Intelligence Platform	17
Overview of the Uses of SAS/CONNECT in the SAS Intelligence Platform	17
SAS/CONNECT, SAS Data Integration Studio, and SAS Enterprise Miner	17
SAS/CONNECT and Grid Computing	18
Initial Configuration of the SAS/CONNECT Server	19
Overview of the Initial Configuration of the SAS/CONNECT Server	19
SAS/CONNECT Metadata Objects	19
SAS/CONNECT Configuration Files	21
Changing the Logging Level of the SAS/CONNECT Spawner	21

Overview of SAS/CONNECT and the SAS Intelligence Platform

SAS/CONNECT software provides the essential tools for sharing data and processing power across multiple computing environments. SAS code uses these tools to perform tasks such as the following:

- dividing time-consuming tasks into multiple units of work and executing these units in parallel
- moving data from a client machine to a server machine, or vice versa, so that the data is on the same machine as the code processing it

Any code that your company writes for use with the SAS Intelligence Platform, such as stored processes, can use these SAS/CONNECT features. For more information about the general capabilities of SAS/CONNECT, see [“Introduction to SAS/CONNECT” on page 16](#) or the *SAS/CONNECT User's Guide*.

In addition, SAS/CONNECT plays some special roles in the SAS Intelligence Platform. For example, in a properly configured environment, some of the platform clients, such as SAS Data Integration Studio and SAS Enterprise Miner, can generate code that includes SAS/CONNECT statements. These statements enable the generated code to perform tasks such as those mentioned in the previous paragraph. For more information about

these special roles, see [“Overview of the Uses of SAS/CONNECT in the SAS Intelligence Platform”](#) on page 17.

If your environment contains SAS/CONNECT, the SAS Deployment Wizard might have configured the product when your system was installed. If this configuration did take place, it is important for you to know what metadata objects and files were created during installation so that you can manage your environment effectively. For more information about how the wizard configures SAS/CONNECT, see [“Initial Configuration of the SAS/CONNECT Server”](#) on page 19.

Introduction to SAS/CONNECT

Overview of Services

SAS/CONNECT provides applications with three types of services:

- **Compute Services.** Use the Compute Services to synchronously or asynchronously direct the execution of SAS programs to one or more server sessions.
- **Data Transfer Servers.** Use the Data Transfer Services to move a copy of your data from one machine to another in order to translate data between machine architectures and SAS versions, as necessary.
- **Remote Library Services.** Use the Remote Library Services to transparently access SAS data that resides in server libraries on machines across the network.

The following sections describe these services briefly. For detailed information, see Chapter 1, “SAS/CONNECT: Definitions and Services,” in *SAS/CONNECT User's Guide*.

Compute Services

The compute services of SAS/CONNECT enable the use of multiple local processors and remote computing resources including hardware, software, and data to most efficiently execute an application.

You can move any or all portions of an application's processing to other processors (either local or remote) in order to use hardware resources, use software in remote environments, interface with legacy systems, and execute code against a remote copy of the data. The results of the remote processing can be returned to the local machine. This is useful when the remote machine has hardware or software that can perform a particular task most efficiently.

Compute services are also helpful if the amount of data to be processed is too large to move to the local machine or if the data is updated too frequently for a local static copy of be useful.

Data Transfer Services

The data transfer services of SAS/CONNECT provide a method for moving a copy of data from one machine to another where a physical copy is then created. Subsequent local processing takes place against the local copy of the data without generating further network traffic until you decide to update the original copy with another transfer. Data transfer services automatically perform any conversion or translation necessary to move data, such as from one SAS release to another or from one machine representation to

another. These services can move data stored in SAS data sets, external databases, and external files.

Remote Library Services

The remote library services of SAS/CONNECT provide access to remote data libraries as if they were stored locally. The data moves through the network only as the local program requests it. A copy of the data is not written to the local files system, and the data must pass through the network on subsequent use by the local processor. This enables you to maintain a single copy of your data and build applications that provide seemingly identical access to local and remote data without requiring the user to know where the data resides.

The Uses of SAS/CONNECT in the SAS Intelligence Platform

Overview of the Uses of SAS/CONNECT in the SAS Intelligence Platform

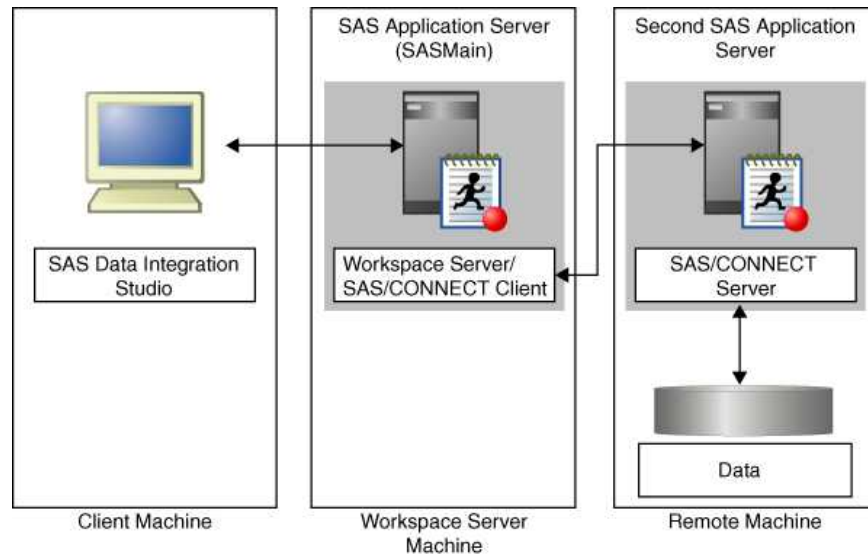
[“Introduction to SAS/CONNECT” on page 16](#) explains the general capabilities that SAS/CONNECT gives SAS programmers. This section explains how some of the SAS Intelligence Platform clients and some of the SAS macros that are provided with the platform use SAS/CONNECT. You must install SAS/CONNECT on the correct machines (and possibly configure a SAS/CONNECT spawner and server) to use the following features:

- the ability of SAS Data Integration Studio and SAS Enterprise Miner to generate SAS/CONNECT code
- the ability of SAS Data Integration Studio and SAS Enterprise Miner to use grid computing

For more information about providing the infrastructure for these features, see the following sections.

SAS/CONNECT, SAS Data Integration Studio, and SAS Enterprise Miner

If you have installed and configured SAS/CONNECT, then both SAS Data Integration Studio and SAS Enterprise Miner can generate SAS/CONNECT code as part of a job. For example, suppose that you are creating a SAS Data Integration Studio job that processes a large amount of remote data. Instead of sending requests for data across the network and results being returned across the network to you, the job can use SAS/CONNECT compute services to submit the code that does the processing to a SAS/CONNECT server that is running on the remote machine, as shown in the following figure.

Figure 4.1 Using a SAS/CONNECT Server to Access Remote Data

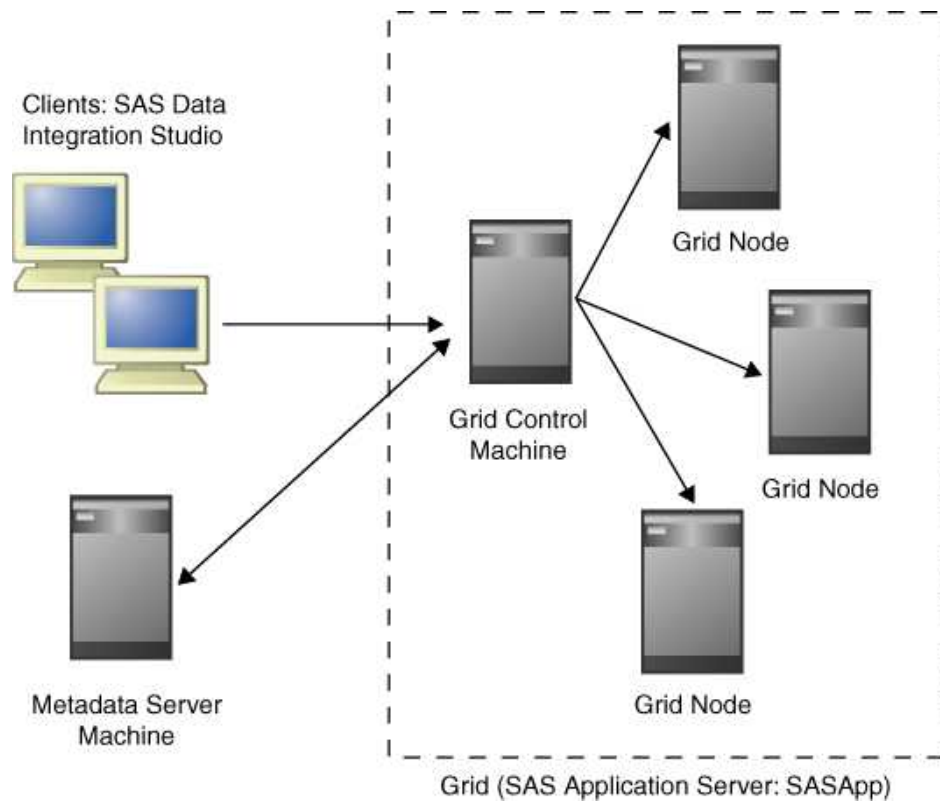
Because the code is now executing on the machine where the data resides, the job can execute much faster.

Alternatively, the job can use the data transfer service in SAS/CONNECT to download the data from the remote host so that the job can run on the local processor. If necessary, the job can also upload revised data.

For complete details about how to set up your environment in order to enable this functionality, see “Setting Up Multi-Tier Environments” in Chapter 5 of *SAS Intelligence Platform: Desktop Application Administration Guide*. For information about creating data integration and data mining jobs, see the user's guides and online Help for SAS Data Integration Studio and SAS Enterprise Miner.

SAS/CONNECT and Grid Computing

If you have invested in the SAS Grid Manager software application, then users of SAS Data Integration Studio and SAS Enterprise Miner can use SAS/CONNECT not only to submit code to a remote host, but to submit code to a grid of processors. The following figure illustrates the role that SAS/CONNECT plays in grid computing.

Figure 4.2 The Role of SAS/CONNECT in Grid Computing

The illustration is designed to give you an idea of how this type of system is set up. For more information, see Chapter 1, “What Is SAS Grid Computing?,” in *Grid Computing in SAS*.

Initial Configuration of the SAS/CONNECT Server

Overview of the Initial Configuration of the SAS/CONNECT Server

“[Overview of the Uses of SAS/CONNECT in the SAS Intelligence Platform](#)” on page 17 looks at several scenarios in which SAS/CONNECT plays an important role in the SAS Intelligence Platform. In the first two scenarios, SAS/CONNECT must be configured on at least one machine in the system. This means that not only was SAS/CONNECT installed on the machine, but the SAS Deployment Wizard configured a SAS/CONNECT spawner and a SAS/CONNECT server on that machine.

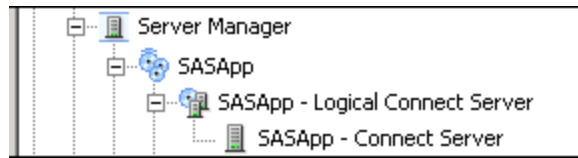
If SAS/CONNECT was configured, then the SAS Deployment Wizard will have created metadata objects and files that are used in the management of the spawner and server. The next two sections explain what metadata objects and files were created so that you can understand how things are currently set up and where you might need to make changes.

SAS/CONNECT Metadata Objects

When the SAS Deployment Wizard configures SAS/CONNECT, it creates at least three metadata objects:

- one representing a SAS/CONNECT spawner
- one representing a SAS/CONNECT server
- one representing a connection to the SAS/CONNECT server

You can see all of these objects in SAS Management Console. (In order to see the connection, you must select the server.)



You can view the properties of each object by right-clicking its icon and selecting **Properties** from the pop-up menu. A properties dialog box will be displayed.

The metadata definition for the SAS/CONNECT spawner is very simple. The main pieces of information that it contains are the following items:

- name of the machine on which the spawner will run.
- name of the metadata definition for the SAS/CONNECT server that the spawner can start. This will be set to *SAS-application-server - Connect Server*.

This metadata definition also specifies whether sign-on scripts are allowed. By default, this value is set to **Yes**.

The SAS Deployment Wizard creates the metadata for the SAS/CONNECT server (and the associated connection) and sets values for the following items:

- name of the CONNECT server, logical server, and spawner
- name of the CONNECT spawner log file
- name, display name, and description of the CONNECT spawner Windows service

The following table lists the SAS/CONNECT properties and their default values as set by the SAS Deployment Wizard:

Table 4.1 Default CONNECT Server, Spawner, and Connection Properties

Setting	Default
SAS invocation options	-dmr -noterminal -nosyntaxcheck
Authentication domain	DefaultAuth
Host name	localhost
Port	7551
SASCMD options	<i>SAS-configuration-directory</i> \Levn\SASApp\ConnectServer\ConnectServer.bat
Requires encryption	No
Encryption algorithm names	(none)
Encryption algorithm key size	0

Setting	Default
Communication protocol	TCP
SIGNON type	Scriptless
Prompt for user ID and password	No
Execute remote submits synchronously	Yes
Display signon window	Yes
Signon wait	Yes

For more information about these settings, see the *SAS/CONNECT User's Guide*.

SAS/CONNECT Configuration Files

The SAS Deployment Wizard creates several directories and files in support of SAS/CONNECT.

The wizard creates two directories under *SAS-config-dir\Levn*: **ConnectSpawner** and **ConnectSpawner\Logs**. The **Logs** directory holds the SAS/CONNECT spawner log. The **ConnectSpawner** directory contains configuration files and scripts related to the operation of the SAS/CONNECT spawner. One of those files, **metadataConfig.xml**, contains the information that the SAS/CONNECT spawner needs in order to connect to the metadata server so that it can read the information that it needs to start a SAS/CONNECT server. The file contains the following types of information:

- name of the machine on which the metadata server is running
- TCP/IP port on which the metadata server is listening
- authentication domain to which the metadata server belongs
- credentials that can be authenticated by the metadata server (those of the SAS Trusted User)
- name of the metadata repository that contains the definition of the SAS/CONNECT server

The wizard also creates a **ConnectServer** and a **ConnectServer\Logs** directory under: *SAS-config-dir\Levn\SASApp*. These directories contain various configuration files for the SAS/CONNECT server, the SAS/CONNECT server log, and a script that the SAS/CONNECT spawner uses to start a SAS/CONNECT server process.

Changing the Logging Level of the SAS/CONNECT Spawner

By default, the SAS/CONNECT spawner does not write much information to its log file. To debug SIGNON problems, you might need to change the logging mode to verbose. For information about how to perform this task, see Chapter 9, “Administering Logging for SAS Servers,” in *SAS Intelligence Platform: System Administration Guide*.

Chapter 5

Understanding the Batch Servers

Overview of SAS Batch Servers	23
The SAS DATA Step Batch Server	24
The SAS Java Batch Server	25
Additional Information	26

Overview of SAS Batch Servers

Batch servers are a required part of the SAS Intelligence Platform's scheduling system. They are metadata objects that store information about how to execute a SAS command in batch mode. For example, when a SAS Data Integration Studio job is scheduled, the following things happen:

- The name of the SAS program that represents the job is read from a deployment directory.
- Information about the program that will execute the job is read from a SAS DATA Step Batch Server.

This information is passed to the scheduling server, which can then run the job at the appropriate time.

There are three types of batch servers:

- SAS DATA Step Batch Server
- SAS Java Batch Server
- SAS Generic Batch Server

A SAS DATA Step Batch Server is used to locate a script that executes SAS programs in batch mode. Generally, these programs are jobs that are created in and deployed from SAS Data Integration Studio. If your deployment plan contained a scheduling component, then a SAS DATA Step Batch Server object will have been defined during the initial installation and configuration of your system. For more information about SAS DATA Step Batch Servers, see [“The SAS DATA Step Batch Server” on page 24](#).

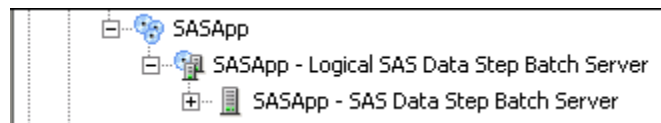
A SAS Java Batch Server points to a Java application that SAS supplies. Each Java Batch server has a subtype that refers to a particular application. If your deployment plan contains a product that uses reporting, such as SAS Web Report Studio, then the SAS Deployment Wizard automatically creates a SAS Java Batch Server for you. For more information about SAS Java Batch Servers, see [“The SAS Java Batch Server ” on page 25](#).

A SAS Generic Batch Server is not used frequently. It enables you to store the path to a stand-alone command or executable that is supplied by SAS, for use cases that do not pertain to the DATA step or Java batch servers.

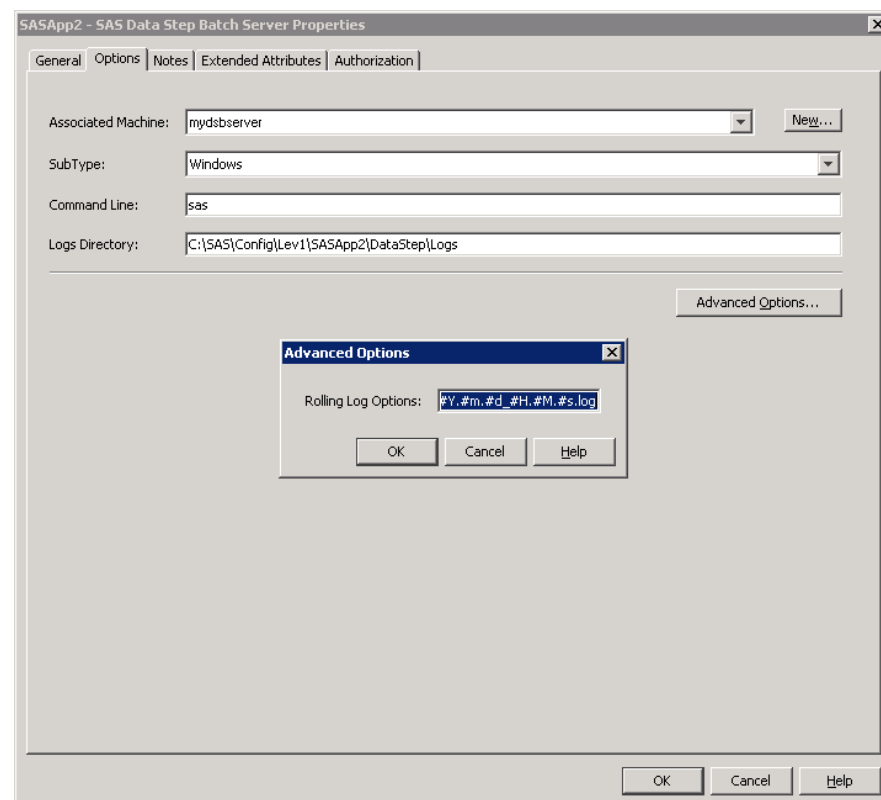
Note: For a complete discussion of scheduling, see *Scheduling in SAS*.

The SAS DATA Step Batch Server

If your environment includes a SAS scheduling component, such as Platform Process Manager, then the SAS Deployment Wizard will have defined a Logical SAS DATA Step Batch Server and a SAS DATA Step Batch Server as part of your SAS Application Server.



The properties of the SAS DATA Step Batch Server are shown in the following display:



These properties are explained in the following list:

Associated Machine

specifies the machine on which the SAS DATA Step Batch Server was configured. This is also the machine on which the deployed job is presumed to reside. It is not necessarily the machine on which the scheduled job will actually execute. The job will execute on the machine where the scheduling server is running. Ensure that any

commands that you store in the batch server are capable of running on the scheduling server.

SubType

specifies the operating system of the machine on which the job will execute.

Command Line

specifies the script that will start SAS in batch mode so that it can execute the job. This script must be able to execute on the scheduling server host.

Logs Directory

specifies the log directory to which SAS can write a log file. This field is required when defining a DATA step batch server. To specify the current directory, enter `./` or an equivalent specification for your host environment.

Rolling Log Options

specifies the format for the name of the log file and indicates when a new log file should be started.

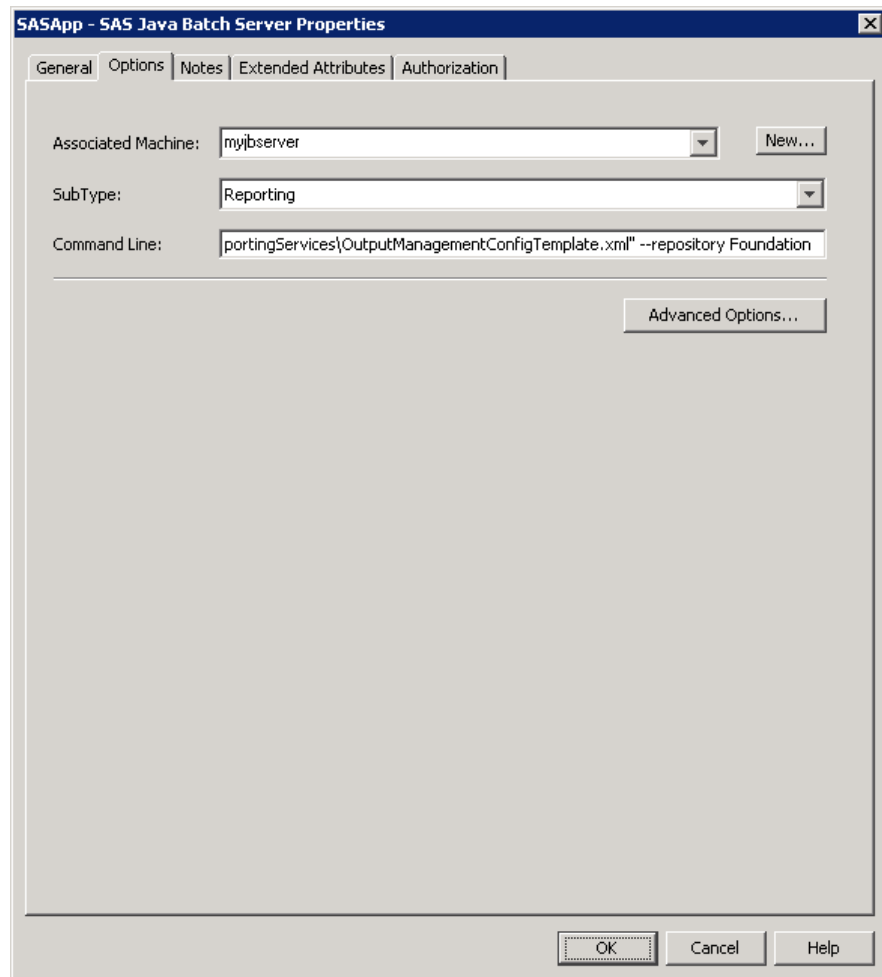
The server is configured to start a new log file automatically when the value of any directive in the name changes. By default, the log file will roll over every second (%s). For information about the directives used in the log filename, see the documentation for the SAS system option LOG=.

Note: If you want to use the SAS 9.3 logging facility instead of the traditional logging feature (prior to SAS 9.2), enter a hyphen in this field, and append this option to the sas command: `-logconfiglocscript-path/logconfig.xml`, where *script-path* is the absolute path to the sasbatch script. For more information, see “Enabling Server Logging” in Chapter 9 of *SAS Intelligence Platform: System Administration Guide*.

The SAS Java Batch Server

If your environment includes a SAS product that relies on reporting, such as SAS Web Report Studio, then the SAS Deployment Wizard will have defined a Logical SAS Java Batch Server and a SAS Java Batch Server as part of your SAS Application Server. The type of Java Batch Server that the wizard creates depends on the product configuration and the subtype that it requires. For example, SAS Web Report Studio requires a **Reporting** subtype that will execute the `outputgen` command.

The SAS Java Batch Server has the properties that are shown in the following display.



The properties are explained in the following list:

Associated Machine

specifies the machine on which the SAS Java Batch Server is configured. This is also the machine on which the reports that are being scheduled must reside. It is not necessarily the machine on which the scheduled job actually executes. The job executes on the machine where the scheduling server is running. Ensure that any commands that you store in the batch server are capable of running on the scheduling server.

SubType

specifies the type of Java batch server that is being started (such as Marketing Automation or Reporting)

Command Line

specifies the command to start a SAS program or the command to run the batch job.

Additional Information

The batch servers are just one part of the SAS Intelligence Platform scheduling system. For information about the larger picture, see *Scheduling in SAS*.

Chapter 6

Understanding SAS Grid Servers

Overview of SAS Grid Servers	27
Overview of Grid Monitoring Servers	28
The Role of the SAS Grid Server in the SAS Intelligence Platform	28
The Initial Configuration of the SAS Grid Server	28
Overview of the Initial Configuration of the SAS Grid Server	28
The Logical Grid Server Metadata Object	29
Logical Grid Server Configuration Files	29

Overview of SAS Grid Servers

The SAS Grid Server serves as a bridge between SAS applications and the grid environment. It enables an application to recognize the grid and submit jobs to it.

The grid server is actually a logical server, which is a component under a SAS application server. A grid consists of the following nodes:

- a grid control server a machine that distributes jobs to machines on the grid. A grid control server can also do work allocated to the grid.
- one or more grid nodes a machine or machines that run a portion of the work allocated to the grid.

A logical grid server is required on both types of nodes.

The grid control server and the grid node include the following components:

- grid middleware provider
- workspace server and spawner (grid control server only)
- DATA step batch server
- Base SAS
- SAS/CONNECT server and spawner

The logical grid server definition specifies the command that is used by the middleware provider to start a SAS/CONNECT session.

To configure grid nodes, you first design the grid and designate a machine to be the grid control server. Next, you set up the logical grid server definitions on all of the grid nodes.

After a grid is configured, you can add grid nodes to increase grid capacity, create the required logical server definitions on a machine, and install the required software on each machine. Specific information about setting up and configuring a grid is on the SAS Scalability and Performance focus area: <http://support.sas.com/rnd/scalability/grid/griddocs.html>. If you are setting up a grid using middleware other than Platform Suite for SAS (such as United Devices GridMP or DataSynapse GridServer), specific values for the fields in a grid server definition are also on the SAS Scalability and Performance focus area.

Overview of Grid Monitoring Servers

A Grid Monitoring Server is also required in order to use grid computing. It is not a component of a SAS application server nor a SAS Management Console plug-in. The grid monitoring server obtains grid usage information from the grid middleware provider (Platform LSF or other). The information is then available for use by the Grid Manager plug-in that is available through the SAS Management Console. During configuration, the user has to specify the middleware provider and the program module that is used to provide the usage information.

Specific configuration information (especially for non-LSF middleware providers) is available on the SAS Scalability and Performance focus area: <http://support.sas.com/rnd/scalability/grid/gridinstall.html>.

The Role of the SAS Grid Server in the SAS Intelligence Platform

You will use the SAS Grid Server only if your company has purchased the SAS Grid Manager software application. This software enables you to build a compute grid that can execute grid-enabled jobs that are created in SAS Data Integration Studio or to execute SAS Enterprise Miner or grid-enabled programs.

Either SAS Data Integration Studio or SAS Enterprise Miner submits a grid-enabled job to a workspace server (through its object spawner) that is running on a grid-control machine. SAS/CONNECT and Platform LSF (or another middleware provider) are also installed on this grid-control machine. Platform LSF starts SAS/CONNECT servers on the appropriate grid nodes. Then SAS/CONNECT statements in the job cause portions of the job to be submitted to the remote SAS/CONNECT servers for execution.

The SAS Grid Server enables Platform LSF to start SAS/CONNECT servers on the grid node. It provides Platform LSF with the command to start the servers, and it provides the script that the command submits.

The Initial Configuration of the SAS Grid Server

Overview of the Initial Configuration of the SAS Grid Server

When you build a SAS compute grid, you configure a logical grid server on the grid-control machine and on each grid node. When you configure a logical grid server on the grid-control machine, two actions occur:

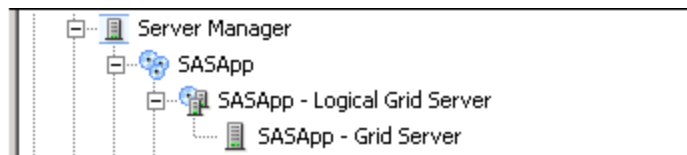
- A metadata object is created to represent the logical grid server. It is a server component that belongs to your SAS Application Server. For more information about this object, see “[The Logical Grid Server Metadata Object](#)” on page 29.
- A **GridServer** directory is created in the configuration directory on that machine. This directory contains files and directories that are used in the operation and management of the grid. For more information about the **GridServer** directory, see “[Logical Grid Server Configuration Files](#)” on page 29.

When you configure a logical grid server on a grid node, only the **GridServer** directory and its contents are created. Only one metadata object is needed per environment. However, the files in the **GridServer** directory are needed on each grid node.

For more detailed information about how to set up a grid, see Chapter 2, “Planning and Configuring a Grid Environment,” in *Grid Computing in SAS*.

The Logical Grid Server Metadata Object

When you configure a logical grid server on a grid node, metadata objects are created that represent the grid server and the associated connection. The grid server belongs to a logical grid server, which in turn belongs to your application server. To display the icon for the grid server in SAS Management Console navigation tree, expand the **Server Manager** icon. Then, expand the application server icon, and expand the **Logical Grid Server** icon.



You can display the properties of the grid server by right-clicking its icon and selecting **Properties** from the pop-up menu. To see the connection object (in the right pane), select the grid server icon. You can view the properties of the connection by opening its properties dialog box.

The SAS Deployment Wizard populates the logical grid server properties with user input that is supplied during the SAS 9.3 installation. For more information about these properties, see *Grid Computing in SAS*.

Logical Grid Server Configuration Files

When you configure a logical grid server on a grid node, a **GridServer** directory is created on the machine. This directory contains the following two items:

- the script **sasgrid.extension**. This is the script that Platform LSF calls to start a SAS/CONNECT server on that machine.
- a **logs** directory. By default, no program writes a log file to this directory. However, you can change the command that is used to start the SAS/CONNECT server so that the server writes a log file to this directory.

Part 3

Load Balancing and Pooling

<i>Chapter 7</i>	
Understanding Server Load Balancing	33
<i>Chapter 8</i>	
Understanding Server Pooling	55
<i>Chapter 9</i>	
Configuring Client-side Pooling	61

Chapter 7

Understanding Server Load Balancing

Overview of Load Balancing	34
What Is Load Balancing?	34
Overview of Planning, Installation, and Configuration	36
MultiBridge Connections (SAS Stored Process Servers Only)	36
Overview of the Initial Load Balancing Setup for Stored Process Servers	37
Security	38
Load-Balancing Algorithms	38
Log Files	39
Planning a Load-Balancing Cluster	39
Overview	39
Select Hosts and Port Numbers	39
Choose a Load-Balancing Algorithm	39
Creating Metadata for Load-Balancing Clusters	40
Overview	40
Convert the Logical Server to Load Balancing	40
Edit Metadata and Convert the Server Instance to Load Balancing	41
Create Metadata for the Other Load-Balancing Servers	41
Installing and Configuring Software for Load-Balancing Servers	41
Overview	41
Install Server and Spawner Software	41
Edit sasv9_usermods.cfg	42
Edit logconfig.xml	42
Edit the .bat File	43
Edit shortcuts.ini	43
Edit OLAPServerSSCU.ini and Install the Service	43
Edits for the UNIX and z/OS Operating Environments	43
Start or Refresh the Object Spawners or OLAP Servers	43
Stopping and Restarting Load-Balancing Servers	44
Adding or Deleting Load-Balancing Servers	44
Understanding the Load-Balancing Algorithms	44
Overview	44
Cost Algorithm: Overview	45
Cost Algorithm: Parameters	46
Cost Algorithm: SAS Workspace Server Example	46
Cost Algorithm: SAS Stored Process Server Example	48
Response Time Algorithm	50
Grid Algorithm	50
Most Recently Used (MRU) Algorithm	51

Least Recently Used (LRU) Algorithm	51
MRU and LRU Algorithms: Pooled Workspace Server Examples	51

Overview of Load Balancing

What Is Load Balancing?

Load balancing distributes SAS sessions across a cluster of servers. You can add or subtract servers to accommodate changes in peak demand.

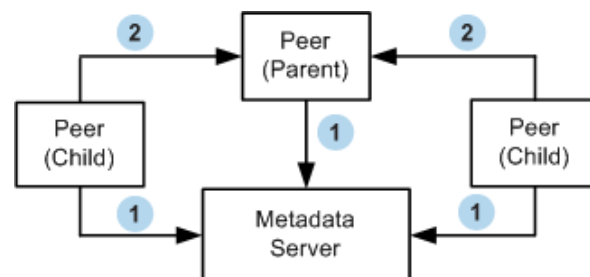
You can create load-balancing clusters of SAS Workspace Servers, SAS Stored Process Servers, SAS Pooled Workspace Servers, and SAS OLAP Servers. For all clusters except OLAP, load balancing is handled by the object spawners that are associated with each server. SAS OLAP Servers do not use object spawners, so OLAP servers handle load balancing directly.

Pooled workspace servers are deployed in load-balancing clusters when you install them with the SAS Deployment Wizard. You can create clusters of the other server types at initial deployment, or at any time thereafter, without having to restart your SAS Metadata Server.

Load-balancing clusters use a peer-to-peer connection. One of the peers in the cluster acts as the parent for the cluster.

The following diagram depicts how peers read cluster metadata at initialization.

Figure 7.1 Cluster Initialization



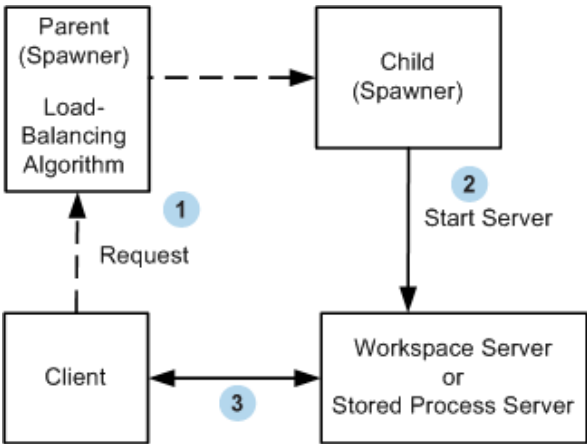
- 1 Peers connect to and read configuration from the metadata server.
- 2 Peers attempt to connect to each other. One peer is designated to accept connections only from the other peers. This peer is known as the parent peer. Other peers are known as child peers. (Peers can be either object spawners or OLAP servers.)

The parent peer runs a load-balancing algorithm to determine the server that is best suited to accept another client. When the parent receives a client request, the parent either begins the session itself or redirects the client to a specific child peer.

If a child peer receives a connection request directly from a client, the child peer routes the request to the parent peer for assignment to the server with the least load.

The following diagram depicts how clients are assigned to a server in the cluster for workspace servers and stored process servers.

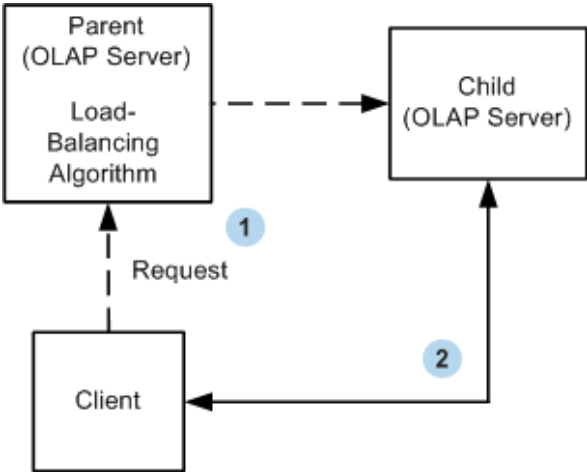
Figure 7.2 Client Assignment to Server (Workspace and Stored Process Servers)



- 1 The parent receives a client request, applies it to its load-balancing algorithm, and redirects the client to connect to a specific peer.
- 2 In the case of a workspace server or a stored process server, the object spawner creates new server instance.
- 3 The client connects to that peer server.

The following diagram depicts how clients are assigned to a server in the cluster for OLAP servers.

Figure 7.3 Client Assignment to Server (OLAP Servers)



- 1 The parent receives a client request, applies it to its load-balancing algorithm, and redirects the client to connect to a specific peer.
- 2 The client connects to that peer server.

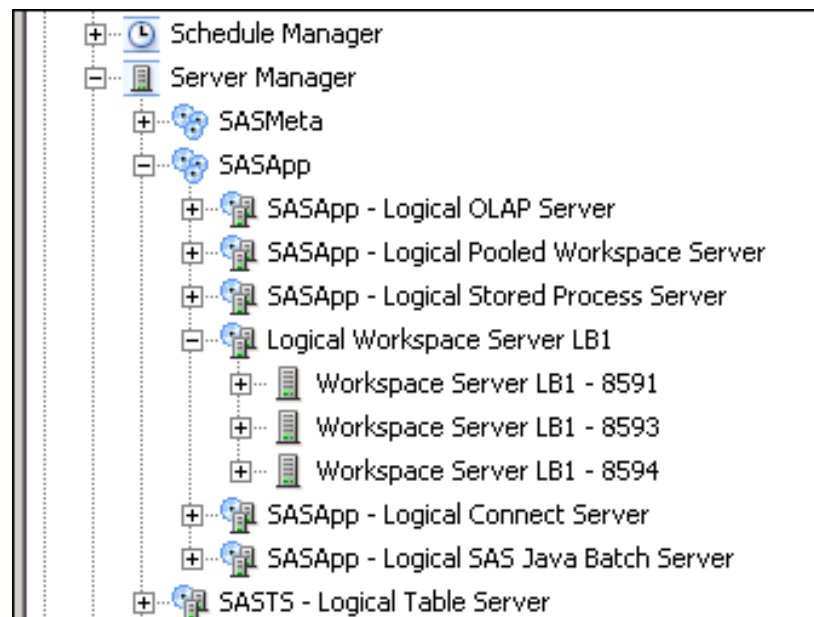
During operation, if the parent peer terminates, one of the child peers assumes the role of parent peer with no interruption of service. When you restart the former parent peer, it restarts as a child peer instead of the parent peer. The new parent peer remains the parent until that peer is restarted.

If a child peer terminates, load balancing continues across the remaining peers. When a child peer is restarted, the parent peer includes that peer in its routing of client requests with no interruption of service.

When you add or remove servers from a cluster, restart all of the peers (object spawners and OLAP servers) in the cluster to ensure that all servers read the latest metadata.

The following display shows how load balancing clusters are defined under a single application server and logical server in SAS Management Console.

Display 7.1 Clustered Workspace Servers in SAS Management Console



Overview of Planning, Installation, and Configuration

To create a load-balancing cluster, you need to plan the configuration, create server metadata, install server and spawner software, configure the server software, and start the servers on their respective hosts.

In the planning stage, you select an application server, a logical server, separate server hosts, unique port numbers, a load-balancing algorithm, and server login credentials.

In the metadata creation stage, you use SAS Management Console to create metadata objects for each server and spawner in the cluster. You then convert the servers and spawners to load balancing.

In the server installation and configuration stage, you use the SAS Deployment Wizard in Install Only mode to install server software (without configuration files) on host computers. For clusters of workspace servers, stored process servers, and pooled workspace servers, you then use the SAS Deployment Wizard to install a spawner on each host. (SAS OLAP Servers do not use spawners.)

After software installation, you copy server configuration files from the cluster's logical server to the new servers that reside on different hosts. You then edit the configuration files and refresh the spawners or servers.

MultiBridge Connections (SAS Stored Process Servers Only)

When you configure load balancing for SAS Stored Process Servers, you must define at least one MultiBridge connection for each server in the cluster. A MultiBridge connection is a specialized bridge connection that is used for stored process servers.

Each MultiBridge connection represents a separate server process and runs on a specific port.

The bridge connection for a stored process server is used only for the initial server request. After the spawner determines which server process has the least load, the client is redirected to the appropriate MultiBridge connection. That is, a client requests the bridge connection for a stored process server, and then the spawner redirects the client to the appropriate MultiBridge connection.

Overview of the Initial Load Balancing Setup for Stored Process Servers

In the initial load balancing SAS Stored Process Server configuration, three MultiBridge connections are set up for the stored process server so that the object spawner can start up to three stored process server processes. The object spawner balances the work load across these processes. The object spawner runs on the server host, listens for client requests, and redirects clients to the appropriate server process.

The metadata server's foundation repository contains the spawner, server, and security metadata for the load balancing stored process server configuration. The object spawner must connect to the metadata server, and the metadata must be configured appropriately, in order for the spawner to start the load balancing stored process server processes.

Note: On Windows, all user IDs are host or domain qualified, for example (*domain-name\sastrust*).

The object spawner obtains the metadata that it needs to start a load balancing stored process server as follows:

1. When the spawner is started, it reads a metadata configuration file named **metadataConfig.xml** that contains information that is required to access the metadata server. This metadata configuration file specifies the following information:
 - the location of the metadata server
 - the user ID that the spawner will use to connect to the metadata server

By default, the **metadataConfig.xml** file contains the user ID **sastrust**, which is owned by the SAS Trusted User (in the metadata).

2. The object spawner connects to the metadata server with the user ID that is specified in **metadataConfig.xml**. This user's credentials are authenticated by the metadata server's authentication provider.
3. On the metadata server, the connection from the object spawner is associated with the user that owns the **sastrust** user ID, SAS Trusted User. The spawner, as the SAS Trusted User, reads the metadata for the server and spawner configuration.

Note: The SAS Trusted User can view the stored process server's multi-user login credentials (**sassrv**) because the SAS Trusted User is a member of the SAS General Servers group. The SAS General Servers group owns the server's multi-user login credentials.

At this point, the object spawner has the necessary metadata to launch a stored process server.

When a client requests a server, the client connects to the spawner and is authenticated using the token that the client received when it connected to the metadata server.

If the object spawner needs to launch a new stored process server, then the object spawner uses the server's multi-user login credentials (**sassrv**) to launch the load-balancing stored process server.

Note: Because the stored process server runs under the multi-user login credentials that are specified in the stored process server definition, each client can access information that only **sassrv** has permission to access.

In your initial load balancing stored process server configuration, you must ensure that the following security is set up properly:

- Ensure that the SAS Trusted User's credentials are specified in the metadata configuration file **metadataConfig.xml**.
- Using the Authentication Manager in SAS Management Console, ensure that, in the foundation metadata repository, the SAS Trusted User is a member of the SAS General Servers group.
- Ensure that the SAS Trusted User has access to the metadata definitions for the object spawner and any servers that it manages.
- Ensure that, in the foundation metadata repository, the group login that is owned by the SAS General Servers group is specified in the stored process server definition. The server login credentials are provided on the **Options** tab of the server's Properties window.
- Ensure that the user ID and password of the group login for the SAS General Servers group match the credentials in a user account that is defined in the stored process server's host authentication provider.

Security

With load balancing, every connection to the server is authenticated with the credentials of the client. Also, every client must have ReadMetadata permission on the server.

The credentials that the server runs under depend on the type of the server:

- SAS Workspace Servers run under the credentials of the client.
- SAS Stored Process Servers and SAS Pooled Workspace Servers run under the multi-user login credentials that are specified in the stored process server definition.

Note: Because the load-balancing stored process server runs under the multi-user login credentials, the operating system account for these credentials must have access to any operating system resources used by stored processes that are hosted on the stored process server.

- SAS OLAP Servers run under credentials of whoever launches it. If it is launched as a service on Windows, this is the local system account by default.

Load-Balancing Algorithms

The parent peer in a load-balancing cluster runs a load-balancing algorithm to evaluate server load and select child peers for new SAS sessions. Five algorithms are available: Cost, Response Time, Grid, Most Recently Used, and Least Recently Used. SAS OLAP Servers are required to use the Least Recently Used or the Grid algorithms. Other types of load-balancing clusters enable you to select between two or more available algorithms.

Log Files

At installation, clustered servers are configured to generate the same error log files that are generated by default on servers that are not clustered, as described in the chapter “Administering Logging for SAS Servers” in the SAS Intelligence Platform: System Administration Guide.

Clusters of SAS OLAP Servers, SAS DATA Step Batch Servers, and SAS/CONNECT Servers generate separate ARM log files in addition to their error log files. You should enable ARM logs only when you are tuning and testing your clusters.

CAUTION:

To ensure the accuracy of your ARM log files, be sure to configure your cluster so that each server writes to a separate ARM log file.

Log files are configured as described in [“Edit logconfig.xml” on page 42](#).

Planning a Load-Balancing Cluster

Overview

To plan for a new load-balancing cluster, you choose hosts, port numbers, a load-balancing algorithm, and logical server login credentials (for OLAP clusters).

Select Hosts and Port Numbers

To select the hosts that will make up your load-balancing cluster, follow these steps:

1. Determine the number of hosts that you will need in your cluster based on your average number of concurrent users and your maximum number of concurrent users.
2. Decide on the logical server that will contain your cluster of servers. You can use an existing logical server or create a new one using the SAS Deployment Wizard. In either case, you will have one configured server available under that logical server, as shown in SAS Management Console. Note that the additional processing load on the parent peer is not significant, so the parent peer does not have to be the highest performing host in your cluster.
3. Select and configure the hosts that will comprise your cluster.
4. Select and write down unique server names and unique port numbers for each host. (Unique ports are required when configuring load balancing on a single machine.) You will enter the server names when you copy and update the server configuration files between hosts. In the server names, you might want to identify the servers as load balancing and include the port number for quick reference.

Choose a Load-Balancing Algorithm

Select a load-balancing algorithm based on the choices that are available for your server type. Clusters of SAS OLAP Servers are required to use either the Least Recently Used or Grid algorithm. The Cost algorithm is recommended for clusters of SAS Workspace Servers and SAS Stored Process Servers.

The following table depicts the algorithms that are available for each server type.

Table 7.1 Available Load-Balancing Algorithms

SAS Server Cluster	Load Balancing Algorithms				
	Cost	Response Time	Grid	Most Recently Used	Least Recently Used
Workspace Server	X		X		
Stored Process Server	X	X	X		
Pooled Workspace Server			X	X	X
OLAP Server			X		X

For more information about the load-balancing algorithms, see [“Understanding the Load-Balancing Algorithms”](#) on page 44.

Creating Metadata for Load-Balancing Clusters

Overview

At this stage in the process of creating a load-balancing cluster, you have identified hosts, chosen server names and port numbers, and selected server login credentials. In the metadata creation stage, you will establish metadata identities for the servers in your cluster and set load-balancing parameters. After this stage, you will install and configure the server software on your hosts.

If you need to install a new SAS Application Server and SAS Logical Server on a new host, do so now using the SAS Deployment Wizard in Configuration Mode. Return to this section afterwards. For information about creating new servers, see [“Add an Additional SAS Application Server”](#) on page 81.

Convert the Logical Server to Load Balancing

To convert your logical server to load balancing (workspace and OLAP servers only), follow these steps:

1. Open SAS Management Console and click **Server Manager** to display the available servers.
2. To display the SAS Logical Server that will contain your cluster, double-click the appropriate SAS Application Server.
3. Right-click the logical server and select **Convert To** ⇨ **Load Balancing**.
4. In the Load Balancing Options window, choose a load-balancing algorithm and choose values for the other available options. For information about the load-balancing algorithms, see [“Understanding the Load-Balancing Algorithms”](#) on page 44. Click **Help** to learn about the load-balancing options.

5. Provide data for the remaining fields in the wizard.

Edit Metadata and Convert the Server Instance to Load Balancing

At this point, your logical server should contain one server instance and one associated spawner process (for all cluster types except OLAP). To edit metadata for the first server and convert the server to load balancing, follow these steps:

1. Open the properties window of the initial server instance beneath the load-balancing logical server.
2. On the **General** tab, rename the server according to the naming convention of your cluster. In the **Description** field, mention the purpose of your load-balancing cluster. Then click **OK** to close the properties window.

Note: This step is optional. Renaming the server helps to identify that it is configured as load balanced in the SAS Management Console Server Manager and in log files.

3. For an OLAP server only, move to the **Connections** tab and open the properties window of the server connection. On the **Options** tab, change the port number if necessary, and then click **OK** to close the properties window.

Create Metadata for the Other Load-Balancing Servers

To create metadata for the new servers in your cluster, follow these steps:

- Right-click the logical server that has been converted to load balancing and select **Add Server**.
- In the New Server Wizard, select the server type that you want to add to your cluster and fill out the rest of the information that is requested. Be sure to enter the appropriate host name, server name, and port number.
- Repeat these steps for the rest of the new servers in your cluster.

Move to the server software installation and configuration stage.

Installing and Configuring Software for Load-Balancing Servers

Overview

At this point, you have planned your cluster and created metadata for your servers and spawners. In this last stage, you will install and configure the server and spawner software on the hosts of your cluster, copy and edit server configuration files, and then start or refresh your spawners or OLAP servers.

Install Server and Spawner Software

The last stage in the process of creating a load-balancing cluster involves the installation of server and spawner software on the hosts of your cluster, and the copying and editing of server configuration files onto your hosts.

To install server software and configuration files on your hosts, follow these steps:

1. Use the SAS Deployment Wizard in Install Only mode to install the server software on the first new host in your cluster. Use Install Only mode so that you do not install server configuration, start-up, or log files. Instead of installing those files automatically, you will add and edit the configuration files manually, later in this procedure. For now, to install server software, see “Adding a New Server in an Existing Application Server” in the chapter “Managing SAS Application Servers.”
2. For the same host, and for all server types other than OLAP (workspace, pooled workspace, or stored process), use the SAS Deployment Wizard to install an object spawner on the same host. Use Configuration Mode to automatically install spawner configuration files.
3. Copy all of the server start-up files from the logical server to the host that received the new server. Be sure to include the subdirectories named logs and sasuser, along with all of their contents. The path to the source files on the logical server should be of the form: *SAS-config-dir/Levn/SASApp1/OLAPServerLB1/*.**. The path to the target on the new host should be of the form: *SAS-config-dir/Levn/SASApp1/OLAPServerLB1/OLAPLB1_2249*, where OLAPServerLB1_2249 is the name of the new server on the new host.
4. Edit the server files as described in the remainder of this section, according to the operating environment of each host, so that the files reflect the name and port of the new server.
5. Repeat these steps for the other hosts, servers, and spawners in your cluster.

Edit sasv9_usermods.cfg

SAS asks that you not edit sasv9.cfg. Instead, edit **sasv9_usermods.cfg**. The user modifications file serves the same purpose as **sasv9.cfg**, because the contents of the .cfg files are hierarchical and inherited.

To edit the file that was originally copied from the cluster's logical server, on each new host in your cluster, follow these steps:

1. In the **-OBJECTSERVERPARMS** option, change **SERVER=** to point to the new server.
2. Add the **PORT=** option to specify the unique port number for the new server.
3. In the Windows operating environment, add the **PROTOCOL=BRIDGE** option.
4. Change the **-LOGCONFIGLOC** option to point to the logconfig.xml file on the new server.
5. Change the **-SASUSER** option to point to the subdirectory of the same name on the new server.
6. Change the **-CONFIG** and **-AUTOEXEC** search paths. Give thought to the configuration settings and **AUTOEXEC** actions that you will share among the servers in your cluster. Add your settings and actions to the appropriate level of the search paths.

Edit logconfig.xml

To edit the log configuration file **logconfig.xml**, follow these steps:

1. For the Rolling File Appender, change the value of the `fileNamePattern` parameter to specify the location of the `.log` file on the new host.
2. For the ARM File Appender, change the value of `fileNamePattern` parameter to specify the location of the `.arm` file on the new host.

All logging behavior can be configured, as described in the SAS Intelligence Platform: System Administration Guide.

CAUTION:

For clusters of SAS OLAP Servers, SAS DATA Step Batch Servers, and SAS/CONNECT Servers, make sure that each server in your cluster writes to a separate ARM log file. Sharing log files results in inaccurate log information.

Edit the .bat File

On Windows, to edit the `.bat` file for your new server, follow these steps:

1. Change the value of `CONFIGDIR=` to include the name of the new server. Be sure not to change the application server context, which is commonly specified by `%APPSERVER_ROOT%`. The new server needs to point to the application server of the first server in the cluster, which runs on a different host.
2. Change the start command to include the name of the new server.

Edit shortcuts.ini

Files with names such as `OLAPServer_shortcuts.ini` might contain user-defined shortcuts. Update the server name in the paths that appear in this file.

Edit OLAPServerSSCU.ini and Install the Service

In the Windows operating environment, edit the server name that appears in various option values in this file. When the edits are complete, run the `.bat` file in install mode. For example, you might execute the following command:

```
C:\SAS\Config\Lev1\SASApp1\OLAP_LB1_2249\OLAPServer.bat install
```

Edits for the UNIX and z/OS Operating Environments

Edit the following configuration files in the UNIX operating environment, or in UNIX System Services for the z/OS operating environment: `rexx.cfg`, `tkmvseenv.fg` and the server shell file (with a name such as `OLAPserver.sh`). In all of these files, you need to change the name of the server.

Start or Refresh the Object Spawners or OLAP Servers

You should now start or refresh the object spawners or OLAP servers in the load-balancing cluster. Refreshing a spawner reinitializes it. The spawners or OLAP servers reread their configuration out of the metadata. As part of this refresh, the spawner shuts down any servers that it currently has started. If changes are made to the server or spawner configurations, the spawner can be refreshed in order to pick up and apply these

new changes. For more information, see “Refresh the Object Spawner” in the chapter “Managing SAS Application Servers”.

At this point, your load balancing cluster should be operational. Check the log files as needed to confirm operation.

Stopping and Restarting Load-Balancing Servers

You must restart the object spawner and the OLAP server. (Any other servers that are servers associated to the object spawner are started by demand.) For information about how to restart these servers, see Chapter 5, “Operating Your Servers,” in *SAS Intelligence Platform: System Administration Guide*.

Be sure to restart all of the servers in a cluster after you add or delete servers.

When you stop a child peer, or if it fails, the parent peer continues to distribute SAS sessions to the remaining servers in the cluster, with no loss of service. You can restart the child peer at any time. In that case the parent peer simply adds the new child peer into its distribution of SAS sessions.

If the parent peer stops, the next server in the list becomes the parent peer. The new parent peer continues the distribution of SAS sessions without interruption.

When you restart the former parent peer, it restarts as a child peer. The new parent peer remains in that role until you restart that server.

Adding or Deleting Load-Balancing Servers

After you create a load-balancing cluster, you might need to add or delete servers in response to accommodate changes in peak demand.

To add or delete servers from a cluster, follow these steps:

1. To add a server, follow the instructions in “[Creating Metadata for Load-Balancing Clusters](#)” on page 40 and “[Installing and Configuring Software for Load-Balancing Servers](#)” on page 41.
2. To delete a server, open the Server Manager in SAS Management Console, right-click the server, and select **Delete**. Remove server software as necessary.
3. Restart all of the peers (object spawners and OLAP servers) in the cluster to ensure that all servers read the latest metadata.

Understanding the Load-Balancing Algorithms

Overview

The following algorithms support load balancing on SAS workspace server, stored process servers, pooled workspace servers, and OLAP servers:

- [Cost](#) on page 45

(SAS Workspace Servers and SAS Stored Process Servers only) The cost algorithm assigns a cost value (determined by the administrator) to each client that connects to the server. The algorithm can also assign cost values to servers that have not started yet. When a new client requests a connection, load balancing redirects the client to the connection with the lowest cost on the machine with the lowest total cost. This is the default algorithm for stored process and standard workspace servers.

- [Response Time on page 50](#)

(SAS Stored Process Servers only) Each spawner's load balancer maintains an ordered list of machines and their response times. Load balancing updates this list periodically at an interval that is specified by the administrator. When a new client requests a connection, load balancing redirects the client request to the machine at the top of the list.

- [Grid on page 50](#)

The Grid algorithm communicates with a SAS Grid Manager to allow load balancing access to grid-related load information. This information is used by the object spawner to find the least loaded server machine that will accept the client request. (This algorithm is available only when the SAS Grid Server has been deployed.)

- [Most Recently Used \(MRU\) on page 51](#)

(SAS Pooled Workspace Servers only) The Most Recently Used algorithm emphasizes reusing workspace servers. This algorithm attempts to send clients into running servers before starting new servers. The goal of this algorithm is to reduce the overhead of starting new servers by using servers that are already running. This is the default algorithm for pooled workspace servers.

- [Least Recently Used \(LRU\) on page 51](#)

(SAS OLAP Servers and SAS Pooled Workspace Servers only) The Least Recently Used algorithm attempts to use the least recently used server. This algorithm provides more of a breadth-first approach to balancing the client load.

Cost Algorithm: Overview

The Cost algorithm uses a cost value to represent the work load that is assigned to each server (or server process) in the load-balancing cluster. Each time a client connects or a stored process is executed, the spawner updates the cost value for the appropriate server. When a client requests a connection to the load-balancing cluster, the spawner examines the cost values for all of the servers in the cluster, and then redirects the client to the server that has the lowest cost value.

The Cost algorithm supports SAS Workspace Servers and SAS Stored Process Servers only. This algorithm works differently depending on the server type:

- SAS Workspace Servers. The cost algorithm uses the server with the lowest cost on the host with the lowest cost.

When a new client requests a connection, the load-balancing spawner redirects the client to the server that has the lowest cost value. When the client connects to the designated server, the spawner increments that server's cost by a specified value (cost per client). When that client disconnects, the spawner decrements that server's cost by the same value (cost per client).

- SAS Stored Process Servers. The cost algorithm uses the server process with the lowest cost on the host with the lowest cost.

When a new client requests a connection, the load-balancing spawner redirects the client to the server process that has the lowest cost value on the machine with the lowest total cost. When the client connects to the designated server process, the spawner increments the cost for that process by the same value (cost per client).

The stored process server cost is determined by three values: the cost per client is used when the client connects; session cost; and server context cost. The default values for session and context costs are 1 and 100 respectively. To change the default values, use the **Options** tab in the properties window of the logical stored process server.

Because stored process servers are reused, there might be processes already running that can be reused if their cost is lowest. Workspace servers are not reused, so SAS is always starting a new server process for workspace servers.

See also:

- [“Cost Algorithm: Parameters” on page 46](#)
- [“Cost Algorithm: SAS Workspace Server Example” on page 46](#)
- [“Cost Algorithm: SAS Stored Process Server Example” on page 48](#)

Cost Algorithm: Parameters

The Cost algorithm uses the following cost parameters, which are treated as weighted values:

Cost per client

(field on the load-balancing logical server definition) specifies the default amount of weight (cost) that each client adds (when it connects) or subtracts (when it disconnects) to the total cost of the server.

Startup cost

(field on the server definition) specifies the start-up cost of the server. When a request is made to the load-balancing spawner, the spawner assigns this start-up cost value to inactive servers. A new server is not started unless it is determined that its cost (the start-up cost) is less than that of the rest of the servers in the cluster. This field enables the administrator to control the order in which servers are started. After a server is started, the cost value is 0. When a client connects to the server, the server's cost value is increased.

Maximum cost

(field on the load-balancing logical server definition) specifies the maximum cost value that each server can have. After a server reaches maximum cost, the load-balancing spawner will not redirect any more clients to the server until its cost value decreases.

See also:

- [“Cost Algorithm: Overview” on page 45](#)
- [“Cost Algorithm: SAS Workspace Server Example” on page 46](#)
- [“Cost Algorithm: SAS Stored Process Server Example” on page 48](#)

Cost Algorithm: SAS Workspace Server Example

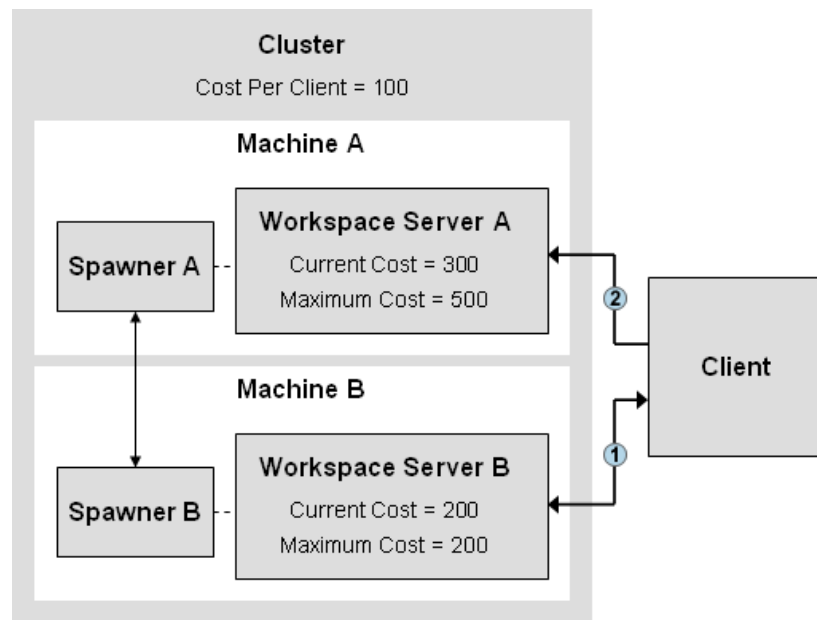
A load balancing cluster contains two workspace servers on two different machines, Machine A and Machine B. The following table displays the initial status of the cluster:

Table 7.2 Initial Cluster Status

Parameters	Workspace Server A	Workspace Server B
Clients	3	2
Maximum Cost	500	200
Cost Per Client	100	100
Cost to Connect	300	200

At the start of the example, five clients have connected to the cluster and the client connections are balanced between the two servers. Workspace Server A has three clients and Workspace Server B has two clients. The following figure illustrates what happens when an additional client requests a connection:

Figure 7.4 New Client Connection



- 1 The client requests a connection to Workspace Server B. The spawner on Machine B examines the cost values of all of the servers in the cluster. Workspace Server B has the least cost, but it has reached its Maximum Cost value and cannot accept any more clients. The spawner redirects the client to Workspace Server A.
- 2 The client requests a connection to Workspace Server A. The spawner on Machine A creates a server connection for the client, and then increments the cost value for Workspace Server A by the cluster's Cost Per Client value (100).

Table 7.3 Final Cluster Status

Parameters	Workspace Server A	Workspace Server B
Clients	4	2

Parameters	Workspace Server A	Workspace Server B
Maximum Cost	500	200
Cost Per Client	100	100
Cost to Connect	400	200

At the end of the example, the cost to connect to Workspace Server A is 400, because there are four clients and the Cost Per Client value is 100.

See also:

- [“Cost Algorithm: Overview” on page 45](#)
- [“Cost Algorithm: Parameters” on page 46](#)
- [“Cost Algorithm: SAS Stored Process Server Example” on page 48](#)

Cost Algorithm: SAS Stored Process Server Example

A load-balancing cluster contains one stored process server with two server processes (MultiBridge connections), Server Process A and Server Process B. The following table displays the initial status of the cluster:

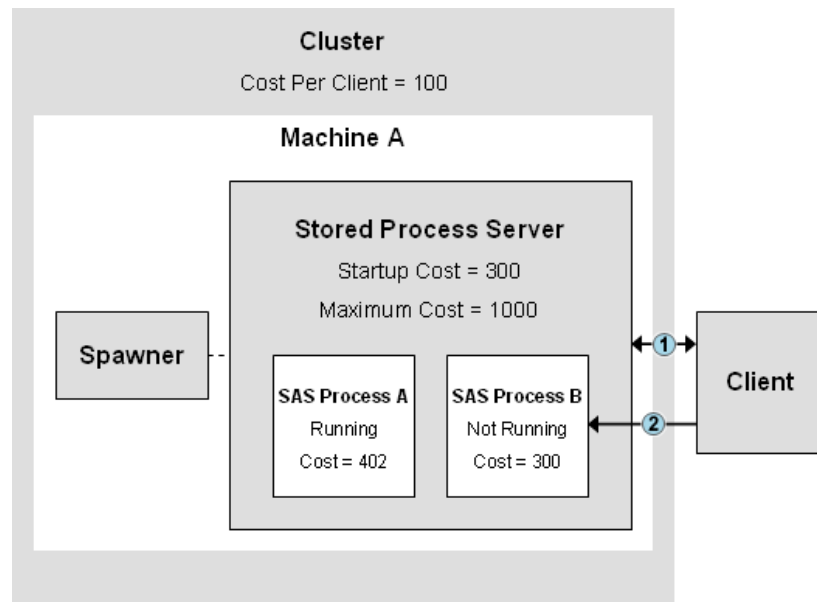
Table 7.4 Initial Cluster Status

Parameters	Server Process A	Server Process B
Status	Running	Not Running
Clients	2	0
Stored Processes	2	0
Startup Cost	300	300
Cost to Connect	402	300
Cost Per Client	100	100

At the start of the example, Server Process A is running and has two clients. Each client on Server Process A is running one stored process, so the cost to connect for Server A is 402 (2 clients * 100 + 2 stored processes running * 101). 100 represents the cost per client. 101 represents the context cost (100) and the session cost (1).

Server Process B has not started yet, so the cost to connect to Server Process B is the Startup Cost (300). The following figure illustrates what happens when an additional client connects:

Figure 7.5 New Client Connection



- 1 The client requests a connection to the stored process server. The load-balancing spawner examines the cost values of all of the servers in the cluster and determines that Server Process B has the lowest cost. The spawner redirects the client to Server Process B.
- 2 The client requests a connection to Server Process B. The spawner starts the server process and then provides a connection to the client. The spawner increments the cost value for Server Process B by the cluster's Cost Per Client value (100).

The following table displays the final status of the cluster:

Table 7.5 Final Cluster Status

Parameters	Server Process A	Server Process B
Status	Running	Running
Clients	2	1
Stored Processes	2	0
Startup Cost	300	300
Cost to Connect	402	100
Cost Per Client	100	100

At the end of the example, the cost for Server Process B is 100, because there is one client and the Cost Per Client value is 100. There are no stored processes running, and the Startup Cost value does not apply because the server process has been started. If the client submits a stored process, the cost will increase by 101 (the standard cost per stored process).

See also:

- “Cost Algorithm: Overview” on page 45
- “Cost Algorithm: Parameters” on page 46
- “Cost Algorithm: SAS Workspace Server Example” on page 46

Response Time Algorithm

The Response Time algorithm uses a list of server response times in order to determine which server process has the least load. For each server process in the load-balancing cluster, the load-balancing spawner maintains an ordered list of servers and their average response times. Each time the spawner receives a client request, it redirects the client to the server process at the top of the list. The spawner updates the server response times periodically. You can specify the update frequency for the response time (response refresh time) in the metadata for the load-balancing cluster.

The Response Time algorithm supports Stored Process Servers only.

The Response Time algorithm uses the following parameters:

Refresh rate

(field on the load-balancing logical server definition) specifies the length of the period in milliseconds that the load-balancing spawner will use the current response times. At the end of this period the spawner updates the response times for all of the servers in the cluster and then reorders the list of servers.

Note: If this field is set to 0, the load-balancing spawner does not use the response time list to redirect clients to servers. Instead, the spawner redirects clients to servers sequentially, in the order in which the servers are defined in the metadata.

Maximum clients

(field on the server definition) specifies the maximum number of clients that a server can have. After a server reaches its maximum number of clients, the spawner will not redirect any more clients to the server until a client disconnects.

Grid Algorithm

If you have a SAS grid installed and configured, then you can leverage the functionality of the SAS Grid Manager to identify the SAS IOM server best suited to handle a SAS client's request in your cluster of workspace servers. The Grid algorithm communicates with a SAS Grid Manager to allow load-balancing access to grid-related load information. This information is used by the object spawner or the OLAP server to find the least loaded server machine that will accept the client request.

The Grid algorithm uses the following parameters:

Grid server

(field on the load-balancing logical server definition) specifies the name of the SAS Grid Server with which the object spawner gathers grid-related load information.

Grid server credentials

(field on the load-balancing logical server definition) specifies valid credentials that the object spawner uses to authenticate with the grid server.

Grid server connect timeouts

(field on the load-balancing logical server definition) specifies the amount of time (in seconds) to wait for a connection to the grid server.

Most Recently Used (MRU) Algorithm

The Most Recently Used algorithm emphasizes reusing workspace servers. This algorithm attempts to send clients into running servers before starting new servers. The goal of this algorithm is to reduce the overhead of starting new servers by using servers that are already running.

The MRU algorithm supports Pooled Workspace Servers only.

The Most Recently Used algorithm uses the following parameters:

Server process maximum

(field on the server definition) specifies the maximum number of server processes that can be created for this server definition.

Server process minimum

(field on the server definition) as server pools grow and shrink on demand, **Server process minimum** specifies smallest the pool can shrink to.

See also, “[MRU and LRU Algorithms: Pooled Workspace Server Examples](#)” on page 51.

Least Recently Used (LRU) Algorithm

The Least Recently Used (LRU) algorithm attempts to use the least recently used server. This algorithm balances provides more of a breadth-first approach to balancing the client load.

The LRU algorithm supports the Pooled Workspace and SAS OLAP Servers only.

The Least Recently Used algorithm uses the following parameters:

Server process maximum

(field on the server definition) specifies the maximum number of server processes that can be created for this server definition.

Server process minimum

(field on the server definition) as server pools grow and shrink on demand, **Server process minimum** specifies smallest the pool can shrink to.

See also, “[MRU and LRU Algorithms: Pooled Workspace Server Examples](#)” on page 51.

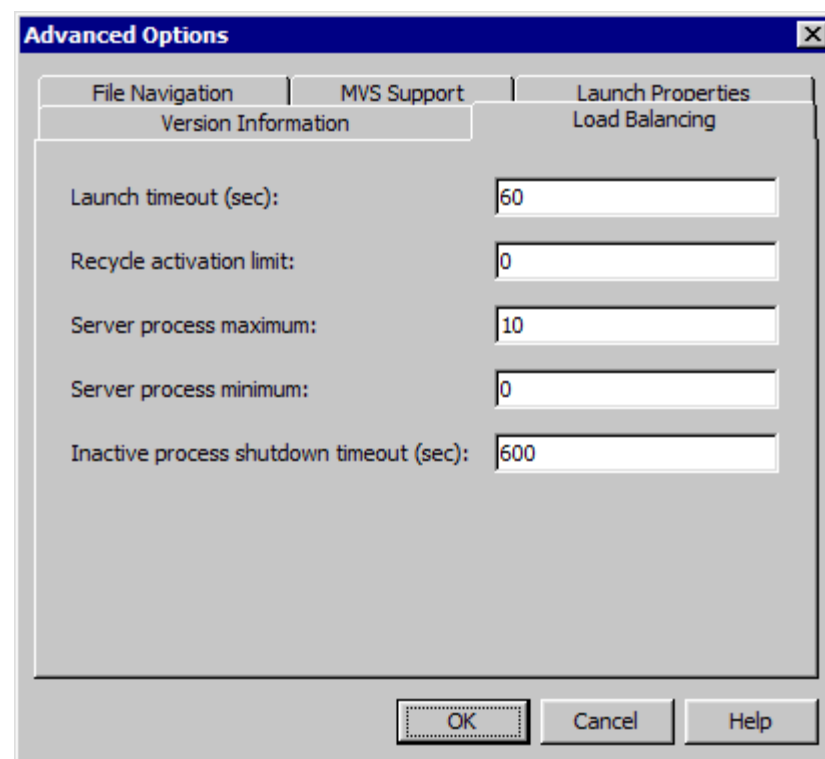
MRU and LRU Algorithms: Pooled Workspace Server Examples

Overview of MRU and LRU Algorithm Examples

The topics in this section examine three examples of load balancing a pooled workspace server using the Most Recently Used (MRU) and Least Recently Used (LRU) load-balancing algorithms:

- “[Example 1: Default LRU and MRU Settings](#)” on page 52
- “[Example 2: Server Process Minimum Increased](#)” on page 53
- “[Example 3: Recycle Activation Limit Set](#)” on page 53

You specify load balancing settings in the SAS Management Console on the **Servers** tab: **Pooled Workspace server** ⇒ **Properties** ⇒ **Options** ⇒ **Advanced Options** ⇒ **Load Balancing**.

Figure 7.6 Advanced Options Dialog Box in SAS Management Console**Example 1: Default LRU and MRU Settings**

The following table displays the default load balancing settings for a pooled workspace server:

Table 7.6 Example 1: Load Balancing Settings (Advanced Options dialog box)

Field Definition	Value
Server Process Maximum	10
Server Process Minimum	0
Inactive process shutdown timeout	600

This pool manages a maximum of 10 clients concurrently. Each server process accepts only one client at a time. The **Server Process Maximum** property determines how many processes can be started. If a client disconnects from one of the servers in the pool, the process continues running and is returned to the pool. The reuse of the server from the pool is determined by the load-balancing algorithm.

The LRU algorithm attempts to use the least recently used server. LRU pools tend to grow faster because the processes that are not running are at the top of the list when looking at least recently used processes.

The MRU algorithm attempts to use a server that was recently used. MRU pools try to reuse processes rather than start new ones, but they will start new processes as needed, up to the maximum. Server processes in the pool that are inactive shut down after 10

minutes (600 seconds). There is no minimum set on this pool, so all the processes in this pool can time out if they are inactive.

Example 2: Server Process Minimum Increased

In example 2, clients are managed in the same manner as in example 1. However, here the **Server Process Minimum** is set to three. When server processes begin to time out, the pool does not shrink below three servers.

Table 7.7 Example 2: Load Balancing Settings (Advanced Options Dialog Box)

Field Definition	Value
Server Process Maximum	10
Server Process Minimum	3
Inactive process shutdown timeout	600

Example 3: Recycle Activation Limit Set

In example 3, clients are managed in the same manner as in example 1. However, here, the **Recycle Activation Limit** is set to 100. This setting allows processes in the pool to be reused for a maximum of 100 times. After the 100th client returns the server to the pool, the pool is restarted.

Table 7.8 Load Balancing Settings (Advanced Options Dialog Box)

Field Definition	Value
Server Process Maximum	10
Server Process Minimum	0
Recycle Activation Limit	100
Inactive process shutdown timeout	600

Chapter 8

Understanding Server Pooling

Overview of Pooling	55
How Server-side Pooling Works	56
Understanding the Server-side Pooling Connection Process	56
How Client-side Pooling Works	57
Understanding the Client-side Pooling Connection Process	58

Overview of Pooling

A collection of reusable workspace server and stored process server processes is referred to as a pool. By reusing server processes, pooling avoids the cost that is associated with creating a new process for each connection. If your client application uses frequent, short-duration connections to SAS, pooling might greatly improve your server performance.

SAS versions, after 9.1.3 support the following two types of pooling:

- server-side pooling

Server-side pooling (introduced after SAS 9.1.3) is the process by which the SAS Object Spawner maintains a collection of workspace servers that are available for clients. The usage of servers in this pool is governed by the authorization rules that are set on the servers in the SAS metadata. For more information, see [“How Server-side Pooling Works” on page 56](#).

- client-side pooling

Client-side pooling is the process by which the client application maintains a collection of reusable workspace server processes. For more information, see [“How Client-side Pooling Works” on page 57](#).

For a discussion on the strengths, weaknesses, and scope for each type of pooling, see [“Choices in Workspace Server Pooling”](#) in Chapter 12 of *SAS Intelligence Platform: Security Administration Guide*.

How Server-side Pooling Works

The usage of the servers in a server-side pool is governed by the authorization rules that are set on the servers in the SAS metadata. Server-side pools use the ReadMetadata permission on the server component metadata objects in the pool to control access. Server-side pooling allows more flexibility than client-side pooling, which limits access to a single group.

Another benefit of server-side pooling is that workspace server load balancing functionality is automatically built in. When a user connects to a pool, the spawner determines the server to send the user to based on the user's access rights and on current server loads.

By using access controls on servers in metadata, the server that actually runs a user's job can be controlled by the user's identity in metadata. (A server defined in metadata, can be one or more processes that are running on one or more machines.) Because different servers can have different command lines, you can use server-side pooling to ensure that a user is assigned to a server process that was launched with a specific command line. One useful implementation of this feature is to launch server-side pooled workspace servers that use an operating system's run priority level.

For example, on UNIX, an administrator might manage server performance by creating the server with the UNIX `nice` command. Lower priority users might use a server that was launched with a higher system value such as `5`. Higher priority users might use a server that was launched with a lower system value such as `-2`. On Windows, the administrator might manage server performance using the `/<priority class>` option on the `start` command. (A user with a lower priority might run on a pooled workspace server started with a start-up command that is similar to: `start/low sas.exe`. Higher priority users might run on a pooled workspace server with a start-up command that is similar to: `start/high sas.exe`.) For more information, refer to your documentation for the appropriate operating system.

Understanding the Server-side Pooling Connection Process

Server-side pooling is the process by which the object spawner maintains a pool of servers available for a client. (The object spawner maintains a port bank to facilitate client connections to this server pool.) The usage of servers in this pool is governed by the authorization rules that are set on the servers in the metadata. The following process describes how an application retrieves and uses a pooled workspace server to handle a user request:

1. A user accesses a SAS client application, and the application requests a connection to a workspace server.
2. The spawner receives the application connection request (on behalf of the user) and decides which server to send the request to based on two criteria:
 - the application user's identity (On which servers does the user have authorization?)
 - the load-balancing algorithm

3. The spawner receives the user connection request and passes the user's credentials to the SAS Metadata Server to determine what servers in the pool the user is authorized to access.
4. The spawner decides which server to send the user to based on the selected load-balancing algorithm, the user's identity, and which servers that the user is authorized to use.
5. The spawner performs one of the following actions based on server availability:
 - If there are servers available, then the spawner redirects the user to an available server.
 - If there are no servers available in the pool and one can be started, then the spawner starts a new server under the launch credentials, adds this new server to the pool, and redirects the user to this server.
 - If the pool has reached its maximum size, then the spawner cannot add any more servers to the pool. The user must wait until there is an available server in the pool or fail after an amount of time. (The maximum size of the pool is determined by the **Server Process maximum** property for each server. The application wait time is determined by the **Availability timeout** property that is found on the Logical Pooled Workspace server.)
6. The workspace server is available to the application to run SAS code. Because the pooled workspace server runs under an administrator-defined host identity, any physical file access that is done by the workspace server process is done under that host identity. This includes access to SAS data sets.
7. When the user has finished using the pooled workspace server, the server is returned to the pool, and it can be reused by other users.

How Client-side Pooling Works

The server processes within a pool are divided into one or more puddles. A puddle is a group of server processes that are accessible to a specific user group and that connect to SAS by using a single set of credentials called the puddle login.

The metadata administrator might choose to create several puddles to control the data that users are authorized to access. Because the SAS server uses the puddle login both to connect to the metadata and to run the server process, this authorization (access control) can be applied in the metadata or on the physical data (by using file system authorization). For example, the metadata administrator might give one puddle Read and Write access to a table on an IOM server, although giving another puddle only Read access.

Another reason for using multiple puddles is to control how available the server processes are for different users. If you are a member of only one puddle access group, then you can access only those connections. If you are a member of multiple puddle access groups, then you have access to more connections.

Note: By default, the SAS Deployment Wizard deploys a server-side pooled workspace server. To completely eliminate use of server-side pooling, delete this logical pooled workspace server. For more information, see [“Remove Logical Servers” on page 89](#).

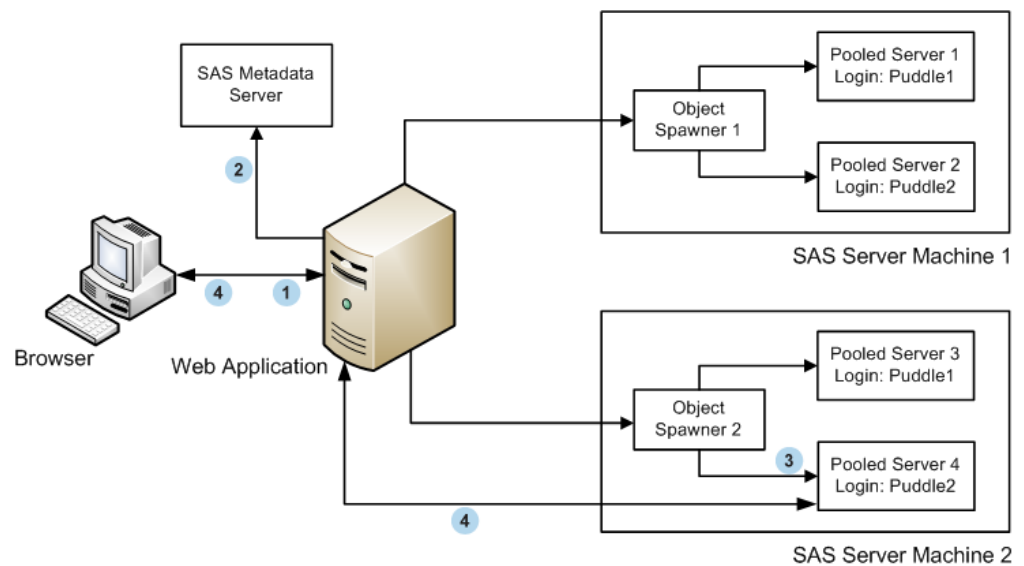
Understanding the Client-side Pooling Connection Process

Consider an example of a client-side pool configuration that consists of:

- two puddles: Puddle1 and Puddle2
- two groups: Puddle1Access and Puddle2Access
- two SAS server machines with an object spawner on each

The following figure shows a connection to a pooled workspace server:

Figure 8.1 The Connection Process for a Client-side Pooled Server



The following process describes how a user retrieves and uses a client-side pooled connection:

- 1 A user, User B, accesses a SAS client application, and the client requests a connection to SAS for the user.
- 2 The client application uses a special account called the pool administrator to connect to the SAS Metadata Server and to read the pool metadata. The pool administrator must be able to view the metadata for all the logins (puddle logins) that are used to make connections for the pool. The pool administrator also needs to be able to view the membership of the puddle access groups. The SAS Trusted User is the default pool administrator.

Note: The pool administrator does not need to be able to view the login definition for the requesting user ID.

- 3 For each puddle, if the minimum number of servers and minimum available servers are not met, the client application uses the appropriate puddle login credentials to launch new server processes.

The pool determines which puddle the requesting user ID can access. The pool selects a puddle where one of the following is true:

- The requesting user ID is a member of the group that is granted access to the puddle.
- The requesting user ID matches the puddle login's user ID, or is owned by the same user or group that owns the puddle login's user ID. (An example for this usage is rare: a single-user application wants to set up a personal connection pool for benefits such as reconnecting after an abnormal disconnect.)

In this example, User B is a member of the Puddle2Access group. The Puddle2Access group has access to the Puddle2 puddle, therefore, User B will access Puddle2.

- 4 The pool manager returns a server connection from the selected puddle as follows:
 - If a server process is available, then the pool returns a connection to the requesting user.
 - If there are no available server processes, and the maximum number of server processes has not been met, then the pool uses the puddle login to create a new server process and returns a connection to the requesting user.
 - If there are no available server processes and the maximum number of server processes has been met, then the requesting user must wait for a server process to become available. When a process becomes available, the pool returns a connection to the requesting user.

Note: For Java client applications, the pool balances the number of connections for each puddle among the server machines. For Windows client applications, all of the connections are assigned to the first server machine, until the maximum number of connections for that machine is met.

The user accesses resources and services on the SAS server. Authorization for content on the SAS server is performed by using the puddle login.

When the user has finished using the server connection, the server process is returned to the pool, and it can be reused by other users.

Chapter 9

Configuring Client-side Pooling

Client-side Pooling Concepts and Overview	61
Is Client-side Pooling Right for My Site?	61
Understanding Client-side Pooling Concepts	62
Overview of Configuring Client-side Pooling	62
Configuring Client-side Pooling	63
Plan the Metadata Identities and Logins for Puddle Access	63
Plan the Puddles for the Client-side Logical Pooled Server	63
Convert a Logical Workspace Server to Client-side Pooling	64
Configure Client-side Pooling Properties for Each Server	65
Setting Client-side Pooling Application Properties	67
Verify That Client-side Connection Pooling Is Working for SAS Web Report Studio	69
Configure Client-side Pooling across Multiple Machines	69
Configuring a Client-side Pooling Workspace Server to Enforce Row-Level Security	72
About Row-Level Permissions Configuration	72
Defining the Necessary Users and Groups	72
Create a Restricted Workspace Server Client-side Pool	74
Assign Libraries to the New Server	76
Create a Second SAS Web Report Studio Deployment	77
Secure Sensitive Data Resources	77

Client-side Pooling Concepts and Overview

Is Client-side Pooling Right for My Site?

Prior to SAS 9.2, client-side pooling was the only option for configuring SAS Workspace servers in a pool. SAS 9.3 continues to support client-side pooling and server-side pooling. For more information about the relative strengths of each method, see these topics:

- [“Overview of Pooling” on page 55](#)
- “Choices in Workspace Server Pooling” in Chapter 12 of *SAS Intelligence Platform: Security Administration Guide*.

Understanding Client-side Pooling Concepts

Before performing this configuration, it is important that you understand a couple of concepts. For example, what does it mean to set up a client-side pooling workspace server? If a workspace server has not been converted to pooling, then each time a SAS Web Report Studio or SAS Information Delivery Portal user starts a session, a workspace server process must be created and the user must establish a connection to this process. This can be a time-consuming sequence of events. When you set up a client-side pooling workspace server, a pool (or set) of connections to workspace servers are opened when SAS Web Report Studio or SAS Information Delivery Portal makes its first request for a workspace server. A user can then obtain a preexisting connection from the pool instead of having to establish the connection.

Another important concept is that of a puddle. A puddle is a subset of the connections in a client-side pool. Setting up puddles enables you to associate a different set of users with different puddles. The Pool Administrator is the user who reads metadata regarding the entire pool. This ID defaults to `sastrust@saspw`. It is not used for the actual workspace server, just to process the pool. Each puddle has a login that is specific to the puddle. Typically, the reason for setting up different groups of users is to give the different groups different levels of access.

For more information, see “Choices in Workspace Server Pooling” in Chapter 12 of *SAS Intelligence Platform: Security Administration Guide*.

Overview of Configuring Client-side Pooling

This topic summarizes the steps required for setting up client-side pooling for your workspace server for use with SAS Web Report Studio and SAS Information Delivery Portal. To contrast client-side pooling with server-side pooling, see “[Overview of Pooling](#)” on page 55. The result of this configuration has better performance than connecting to a standard workspace server.

To set up a client-side pool, you must do the following:

1. “[Plan the Metadata Identities and Logins for Puddle Access](#)” on page 63
2. “[Plan the Puddles for the Client-side Logical Pooled Server](#)” on page 63
3. “[Convert a Logical Workspace Server to Client-side Pooling](#)” on page 64
4. “[Configure Client-side Pooling Properties for Each Server](#)” on page 65
5. “[Verify That Client-side Connection Pooling Is Working for SAS Web Report Studio](#)” on page 69

Note: By default, the SAS Deployment Wizard deploys a server-side pooled workspace server. To completely eliminate use of server-side pooling, delete this logical pooled workspace server. For more information, see “[Remove Logical Servers](#)” on page 89.

If your deployment requires more than one host for client-side pooled workspace servers, then you can configure client-side pooling across multiple machines. For more information, see “[Configure Client-side Pooling across Multiple Machines](#)” on page 69.

Configuring Client-side Pooling

Plan the Metadata Identities and Logins for Puddle Access

To plan the client-side pooling security, you must determine the user metadata identities, and the logins for the group metadata identities, that can access the puddles in the pool. Your user, group, and login definitions for the users and groups that will access the pool should already be set up when SAS was installed. However, before you determine which identities and logins to use, do the following:

1. Review the access controls on the server. For more information, see Chapter 7, “Permissions on Servers,” in *SAS Intelligence Platform: Security Administration Guide*.
2. Review access control for data on the server. The puddle login requires operating system access to any data sets that the server will retrieve.

For puddle access to the pool, there are two types of logins that you can define:

- a login that is used to launch the workspace server for the puddle users. All users of the puddle use this login when connecting to the SAS server. This login must be accessible to client-side pool administrators. (Pool users are not required to have access to this login).
- logins for the client-side pool administrator in the metadata configuration file that is used with the Windows Object Manager. *Important Note: Do not specify an unrestricted user for the user ID of the client-side pool administrator.*

Plan the Puddles for the Client-side Logical Pooled Server

To plan a client-side pooled logical server, you need to determine how many puddles you want to use and which groups will be used to access each of the puddles. When you convert the logical server to a client-side pooled logical server, you can then divide the pool into one or more puddles that associate the appropriate login definition and group metadata identity to use for access to the puddle. The login for each puddle is used to launch the server.

Note: Multiple puddles are not required. It is very common to have only a single puddle.

Determine the following parameters for each puddle that is associated with the client-side pooled logical server definition. Retain this information because you will need it later in “Convert a Logical Workspace Server to Client-side Pooling” on page 64.

Table 9.1 New Puddle Information

Field	Explanation
Name	Specifies the name of the puddle (for example, Puddle1).

Field	Explanation
Minimum available servers	Specifies the minimum number of servers that should be idle and available in the pool at any time. The number of running servers never exceeds the Maximum Clients value specified on the server. If the number of available servers falls below the value specified in this field, additional server connections are created.
Minimum number of servers	Specifies the minimum number of servers (both idle and active) that should be present in the pool at any time. This value also specifies how many servers should be started when the pool is first created.
Login	Specifies the user ID under which connections to the server are made. The SAS server only sees this login when a connection is made, rather than the user ID of the user who is actually using the pool. The menu on this field lets you select from the three most recently used logins or the More Logins option. Selecting More Logins displays the Login Search window, which you can use to search for a defined login.
Grant access to group	<p>Specifies the logon group whose members are allowed to access the puddle. Users who are not members of the group are not allowed access. When a user requests a connection to a server pool, the software searches for a puddle whose access group contains that user.</p> <p>If you want to allow all users to access the system, you should change this value to PUBLIC.</p> <p>Note: If you do not want members of PUBLIC to be able to use SAS Web Report Studio, <i>and</i> you want to present such users with a clear and user-friendly error message, follow these directions. First, set the value of the Grant Access To Group field to SASUSERS (to control access to the pool). Then, edit the properties file WEB-INF\WebReportStudioProperties.xml, and change the value of the <code>wrs.pfs.allowPublicUsers</code> element to false (to provide useful application-level feedback to members of PUBLIC).</p>

Convert a Logical Workspace Server to Client-side Pooling

To convert your workspace server to client-side pooling and to define a puddle, follow these steps:

Note: You are setting up client-side pooling for a specified client and certain types of requests. This logical workspace server also supports standard workspace servers for other clients.

1. Log on to SAS Management Console as the SAS Administrator (**sasadm@saspw**).
2. In SAS Management Console, expand the Server Manager tree node and the node for the SASApp application server. One of the tree nodes under SASApp is SASApp - Logical Workspace Server.

3. Right-click the icon for the logical workspace server, and select **Convert To** ⇨ **Pooling**. The Information dialog box asks whether you want to continue with the conversion. Click **Yes**. The Pooling Options dialog box appears.
4. In the Pooling Options dialog box, click **New** to indicate that you want to define a puddle.

The New Puddle dialog box appears.

5. Enter the values that you chose in “Plan the Puddles for the Client-side Logical Pooled Server” on page 63 .

Note: If you are not logged on to SAS Management Console as the SAS Administrator (`sasadm@saspw`), you might not see `sassrv` in the **Login** drop-down list box. In this case, click **Cancel** in the New Puddle dialog box. Then, reconnect to the metadata server by using the metadata profile for the SAS Administrator.

6. Click **OK** in the New Puddle dialog box.
7. Click **OK** in the Pooling Options dialog box.

Configure Client-side Pooling Properties for Each Server

To set the configuration options for the client-side pool, follow these steps:

1. In SAS Management Console, expand the SASApp - Logical Workspace Server to reveal the icon for the workspace server.
2. Right-click the workspace server icon, and select **Properties** from the pop-up menu. A Workspace Server Properties dialog box appears.
3. In the Workspace Server Properties dialog box, select the **Options** tab.

Note: If the workspace server is converted to use client-side pooling, information about this tab might not accurately reflect behavior for connections that use this server as a standard workspace server. For a *logical* workspace server that offers client-side pooling, the **Options** tab is disabled and always indicates that credential-based host authentication is configured. However, if you selected some other setting before configuring client-side pooling, that selection is persisted in metadata and is honored when the server is used as a standard workspace server. To avoid confusion, do not use a workspace server for both client-side pooling and standard use. Instead, if you choose to use client-side pooling, create a dedicated workspace server for that purpose, in a separate server context.

4. On the **Options** tab, click **Advanced Options**. The Advanced Options dialog box appears.

- In the Advanced Options dialog box, select the **Performance** tab.

The screenshot shows the 'Advanced Options' dialog box with the 'Performance' tab selected. The fields are as follows:

Field	Value
Maximum cost:	500
Startup cost:	200
Launch timeout (sec):	10
Maximum clients:	8
Recycle activation limit:	50
Inactivity timeout (mins):	0

- Fill the fields in the dialog box as shown in the preceding display. The following table explains the meaning of each value.

Table 9.2 Pooling Properties

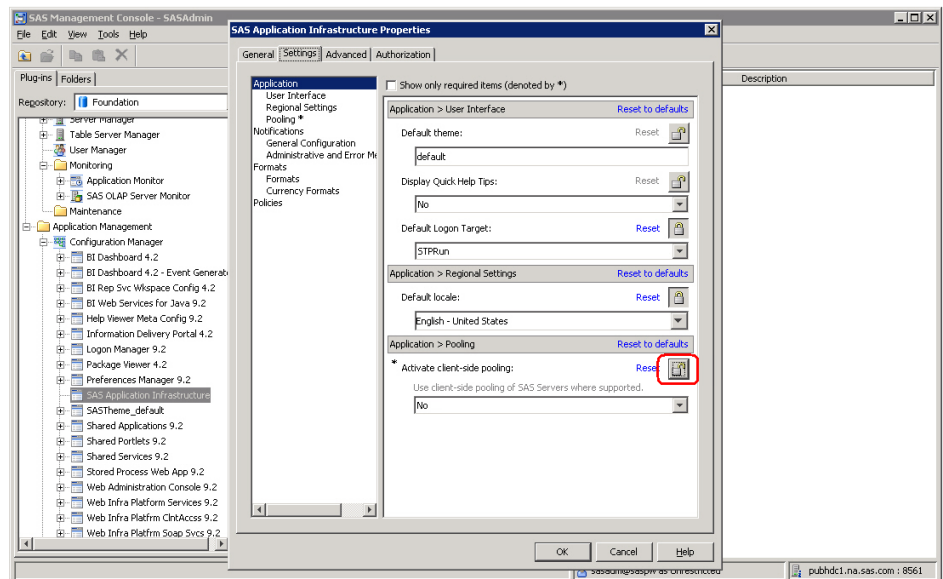
Field	Explanation
Maximum clients	Each client requires a workspace server process on the workspace server host. These processes constitute the pool that will be available to SAS Web Report Studio and SAS Information Delivery Portal. Each process requires approximately 150 MB for efficient operation. As a guide, we recommend two processes per CPU. If the server is <i>not</i> also acting as the metadata server, you can add one or two to this maximum. If you have long-running queries, you can add one or two servers. This will make the system seem more responsive to short-running queries, at the expense of total throughput. A typical setting is between 8 and 10. Applications should not share the same pooled workspace configuration as this creates duplicate pools, one for each application, and an unexpectedly high workload for the server host.
Recycle activation limit	Places a limit on how often workspace server processes are reused to satisfy puddle connections.
Inactivity timeout	A workspace server process can have an inactivity time-out. Having a short time-out reduces the workload on the server host, but might reduce response time for client connection requests.

- Click **OK** in the Advanced Options dialog box.
- Click **OK** in the Workspace Server Properties dialog box.

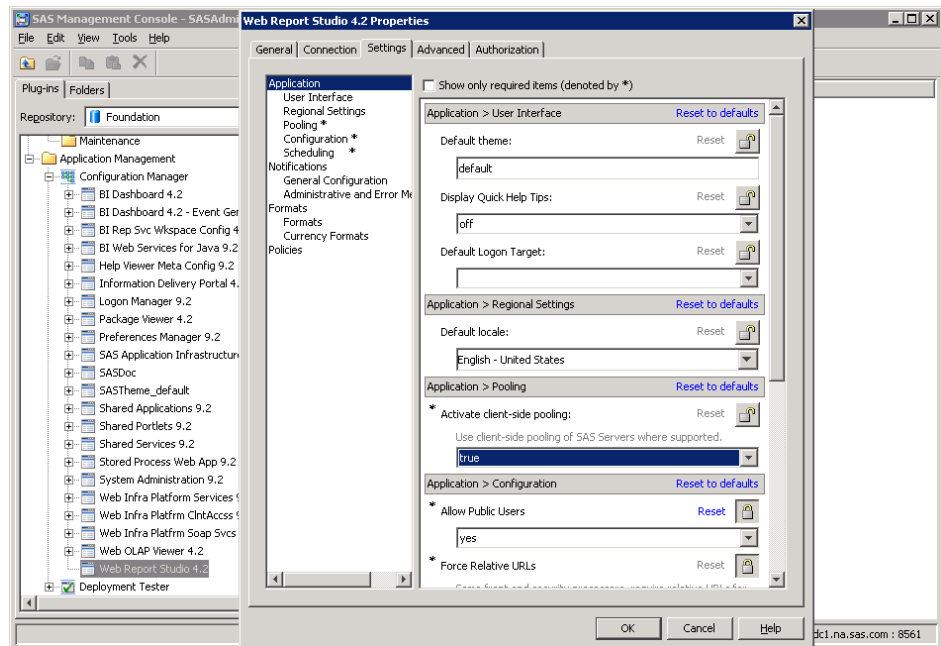
Setting Client-side Pooling Application Properties

To turn on client-side pooling properties for SAS applications such as SAS Web Report Studio, the SAS Information Delivery Portal, and SAS BI Dashboard, follow these steps:

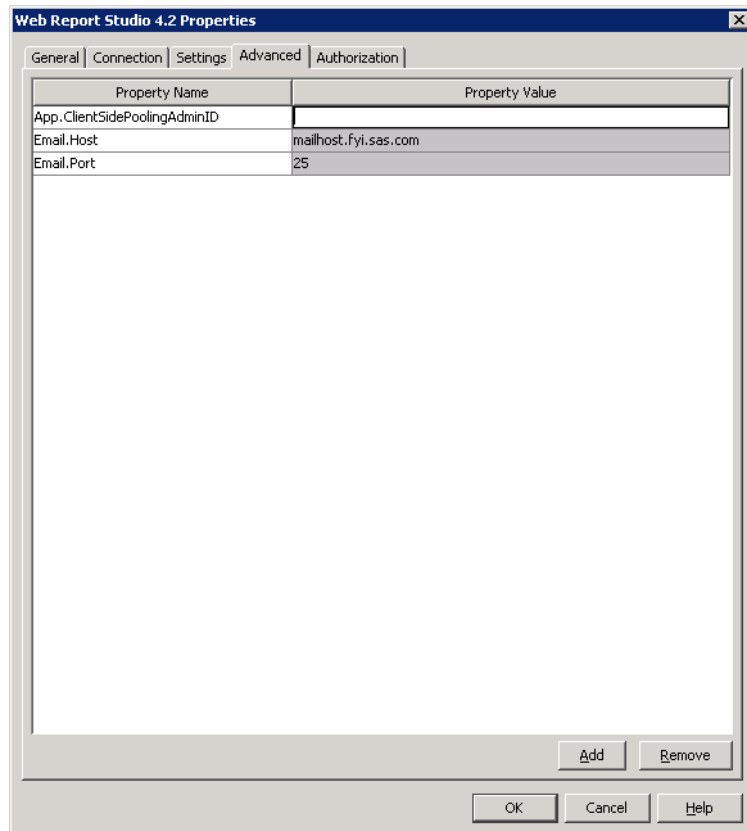
1. Using SAS Management Console, access the SAS Configuration Manager plug-in (**Plug-ins** ⇒ **Application Management** ⇒ **Configuration Manager**).
2. Make sure that client-side pooling is *not* locked at the application infrastructure level, by following these steps:
 - a. Expand the Configuration Manager icon, right-click on **Application Infrastructure**, and in the pop-up menu choose **Properties**.
 - b. In the Properties dialog box, choose the **Settings** tab.
 - c. In the right pane of the **Settings** tab, under **Application > Pooling**, click the lock icon. It should toggle to an unlocked state, and then click **OK**.



3. In the left pane of the SAS Management Console under Configuration Manager, right-click on **Web Report Studio 4.2**, and in the pop-up menu choose **Properties**.
4. In the Properties dialog box, choose the **Settings** tab.
5. In the right pane of the **Settings** tab, under **Application > Pooling**, choose **true** from the drop-down list box.



- Click the **Advanced** tab, and in the **Property Value** column for **App.ClientSidePoolingAdminID**, enter the login for the client-side pooling administrator, and then click **OK**.



- Restart your Web application server for these changes to take effect.

For more information, see “Starting and Stopping Web Application Servers” in Chapter 5 of *SAS Intelligence Platform: System Administration Guide*.

Verify That Client-side Connection Pooling Is Working for SAS Web Report Studio

If you have not configured client-side connection pooling correctly, SAS Web Report Studio will continue to work. However, it will not be able to take advantage of a connection pool. Therefore, it is important that you verify that your system is configured correctly. You can verify this by temporarily changing the logging level for connections, viewing a SAS Web Report Studio report, and then checking the contents of the SAS Web Report Studio log file.

To verify that your system is configured correctly, follow these steps:

1. Using SAS Management Console, add a new logging context (com.sas.services.connections) and set its logging level to DEBUG.

For more information, see “Configuring Logging for SAS Web Report Studio ” in Chapter 4 of *SAS Intelligence Platform: Web Application Administration Guide*.

2. Start SAS Web Report Studio and log on.
3. View a report that accesses relational data (for example, a SAS table).
4. View the SAS Web Report Studio log file (*path-to-config-dir\Web\Logs\WebReportStudio.log*). If pooling is working, you will see information that is similar to this message about the connection to the workspace server:

```
privileged user name: D9588\sastrust@saspw
pd#0: putting cx#8 on the available queue
rq#0 routed to pd#0
```

If pooling is not working correctly, you will see a message similar to this message:

```
request served by unshared connection #9
```

After you have confirmed that connection pooling is working, you can undo the changes that you made in step 1.

Configure Client-side Pooling across Multiple Machines

If your deployment requires more than one host for client-side pooled workspace servers, then you can configure client-side pooling across multiple machines.

An important aspect of multi-machine workspace client-side pooling involves the puddle login ID that you supplied in the topic “[Convert a Logical Workspace Server to Client-side Pooling](#)” on page 64. In order to implement multi-machine, client-side pooling, this login must be valid for all machines in the cluster.

This topic describes a basic configuration where the client-side pool has only one puddle. For this basic implementation, the account that you use for the login ID must be able to authenticate on every machine in the cluster. This means that the login ID must be a network account, and it must be associated with a single authentication domain in metadata (for example, NTDomain).

If you require a workspace client-side pool that consists of host servers in different authentication domains (for example, Windows and UNIX), then you can configure a pool with a separate puddle for each host. Each puddle has its own login ID as

appropriate for the host's authentication domain (for example, NTDomain or UnixAuth). Each login ID must be able to authenticate on every machine in its respective cluster. With this configuration, the pool manager software can identify which machine is used for a given process, based on the puddles that are defined for the pool, and it uses the appropriate login to make a connection.

The following instructions explain how to configure workspace client-side pooling across two machines in the same authentication domain, both supporting the same puddle. Repeat these instructions as appropriate for each additional machine that you want to configure for that same puddle.

Note: These instructions assume that you have already set up and verified client-side pooling on a single machine, as explained in the previous sections.

To configure client-side pooling across two machines, follow these steps:

1. Log on to SAS Management Console as the SAS Administrator (`sasadm@saspw`).
2. If the puddle login ID is not accessible to all machines in the client-side pooling cluster, then you must create a new user. Follow these steps:
 - a. On the host, identify or create a network account. For example, on Windows the user name might be `NTDomain\myaccount`. The user must authenticate against each machine in this cluster.

If creating the account on Windows, grant the "Log on as a batch job" user right for the account on each Windows machine in the cluster.

- b. In the SAS Metadata Console User Manager plug-in, add a login to the SAS General Servers group with the user ID and password of the user that you just created.

This login should be associated with the server's authentication domain (for example, DefaultAuth or ServerAuth), even if you are using middle-tier trusted authentication. The login is used to start processes on the server machine rather than to authenticate against the metadata server.

- c. Change the login ID that is used for the puddle to this newly established login ID. Make this change in the Edit Puddle dialog box, which is accessed from the Logical Workspace Server Properties dialog box under the **Server Manager** plug-in. For details, see [“Convert a Logical Workspace Server to Client-side Pooling”](#) on page 64 .
3. In the SAS Metadata Console Server Manager plug-in, add a pooled workspace server definition for the machine that you are adding to the client-side pool. Follow these steps:
 - a. Expand the SASApp application server, right-click Logical Workspace Server, and then select **Add Server** from the pop-up menu. The New Server Wizard appears.
 - b. Specify the name for the new server that is being added to the client-side pool and click **Next**.
 - c. Click **New**. In the Add New Machine dialog box, enter the name of the new machine where the server resides, and click **OK**.
 - d. In the Available list, select the new machine.
 - e. Click **Advanced Options**, and then select the **Performance** tab.
 - f. On the **Performance** tab, set the client-side pooling options for this machine and then click **OK**. The maximum number of clients can vary for each server that is included in the pool, and should be configured based on the capacity of each

physical server that is being used. For details, see [“Configure Client-side Pooling Properties for Each Server”](#) on page 65 .

4. Click **Next**.
5. Click **Next** to accept the default authentication domain and port.
6. Review the information in the Summary dialog box. If no changes are required, click **Finish**.
7. To define a new object spawner, in the Server Manager plug-in, follow these steps:
 - a. Right-click the Server Manager and select **New Server** from the pop-up menu. The New Server Wizard appears.
 - b. In the wizard, choose **Object Spawner** as the type of server, and then click **Next**.
 - c. Provide information in the wizard as appropriate, by using these guidelines:
 - Select the host name that you are adding to the pool.
 - Select the newly created pooled workspace server as the server to be spawned by this object spawner.
 - Accept defaults for operator connection, ports, and authentication, unless you have implemented these differently.
8. Under SASApp, expand the logical pooled workspace server, and verify that both physical workspace servers appear under the logical client-side pooled workspace server.
9. Install the necessary SAS software on the machine that is being added to the pool. At a minimum, you should install Base SAS, the SAS Workspace Server, and the SAS Object Spawner. When configuring the object spawner, make sure that the object spawner can connect to the metadata server (running on another machine). If the spawner cannot access the metadata server, then the object spawner cannot be started.
10. Verify that pooling works for SAS Web Report Studio.
 - a. Start the metadata server.
 - b. Start the object spawner on the machine that is being added to the pool.
 - c. Start or restart the middle-tier components.

Note: Because the client-side pool administrator did not change, there is no need to change the middle-tier configuration. Recall that the pool administrator (typically `sastrust@saspw`) is used by the middle-tier application to process client-side pooling requests.

Depending on the puddle configuration and the maximum number of clients that are allowed for each server under the logical client-side pooled workspace server, the number of pooled workspace server processes that show up on each machine will vary.

For more information, see [“Verify That Client-side Connection Pooling Is Working for SAS Web Report Studio”](#) on page 69.

As mentioned earlier in this section, if you require a workspace client-side pool that consists of host servers in different authentication domains, then you can add more puddles to the pool. For each puddle, provide a login ID that is able to authenticate against all machines in the cluster.

The data sets that are accessed by the client-side pooled workspace server processes must either be replicated on each machine that will execute the pooled workspace server processes or accessed via a shared location that uses a common shared path (for example, a UNC or NFS path). For more information about putting the data in a shared location, see [“Overview of Managing Data and Catalogs for Servers on Multiple Machines” on page 91](#). Alternatively, the workspace server processes can access data by using some other sharing mechanism, such as a SAS/CONNECT server. All **LIBNAME** assignments use the same path information, regardless of which physical machine the process runs on.

The object spawner process on the machine that has been added to the client-side pool cannot be started until the metadata server is running. If the object spawner is configured to run as a service that starts automatically, then this dependency cannot be enforced via the Windows services configuration.

Configuring a Client-side Pooling Workspace Server to Enforce Row-Level Security

About Row-Level Permissions Configuration

After the initial installation and configuration of the SAS Intelligence Platform, most sites have a single workspace server, which is part of the default SAS Application Server, SASApp. By default, this workspace server is a standard workspace server, which means that workspace server processes are spawned on an as-needed basis. If you need comprehensive security, set up the high-security configuration of SAS Web Report Studio. This configuration prevents regular users from circumventing row-level filters by accessing the target tables directly (without going through the information map that enforces the filters).

This section explains how to create a second client-side pooling workspace server that you can use as part of an environment in which row-level permissions are enforced. For more information about this environment, see Chapter 1, “Overview of BI Row-Level Permissions,” in *SAS Guide to BI Row-Level Permissions*.

Defining the Necessary Users and Groups

Overview of Defining the Necessary Users and Groups

The first step in setting up the new client-side pooling workspace server is to define two accounts for users who must be authenticated by the operating system on the workspace server host, and several user and group metadata objects.

Create User Accounts

Create user accounts that will enable the operating system on the workspace server host to authenticate the following users. On Windows systems, these accounts can be domain accounts or local accounts:

- **rpooladm** - This account is for the client-side pool administrator, the user who handles requests for processes in the workspace server client-side pool. The password for this account should be unique.

- `rpoolsrv` - This is the puddle login account. Each SAS Web Report Studio user who has access to the client-side pool must belong to a group that has a login that contains this user ID and the associated password.

On Windows systems, grant both of these users the **Log on as a batch job** right. If you created a SAS Server Users group when you first installed the SAS Intelligence Platform, you can give these users this right by adding them to that group.

Once you have created these user accounts, you should create the metadata objects described in the next section.

Create User and Group Objects

In SAS Management Console, use the User Manager to create one user and two groups:

- User: Restricted Client-side Pool Administrator
- Group: Restricted Client-side Pool Puddle Login
- Group: Restricted Client-side Pool Puddle Access

Define the Restricted Client-side Pool Administrator

To define the Restricted Client-side Pool Administrator, follow these steps:

1. Right-click **User Manager**, and select **New** ⇒ **User** from the pop-up menu.
2. In the New User Properties dialog box, on the **General** tab, enter the name **Restricted Client-side Pool Administrator** in the **Name** box.
3. In the same dialog box, select the **Accounts** tab.
4. Add a login to the user object by selecting **New** and entering the appropriate information in the New Login Properties dialog box. In the **User ID** box, enter the name `rpooladm`. If the operating system account for this user is a Windows account, qualify the name with a domain or machine name and a backslash. Then select the authentication domain for your workspace server from the **Authentication Domain** drop-down list. It is preferable, for security reasons, not to put the password in the metadata. Click **OK** at the bottom of the dialog box.
5. Click **OK** in the New User Properties dialog box.

Define the Restricted Pool Puddle Login Group

To define the Restricted Pool Puddle Login group, follow these steps:

1. Right-click **User Manager**, and select **New** ⇒ **Group** from the pop-up menu. A New Group Properties dialog box appears.
2. On the **General** tab, enter the name **Restricted Client-side Pool Puddle Login** in the **Name** box.
3. On the **Members** tab, select the **Restricted Client-side Pool Administrator**, and click the right-arrow button to move the user to the **Current Members** list.
4. On the **Accounts** tab, create a new login that contains the credentials for `rpoolsrv` and the authentication domain of the workspace server. (See step 4 in the [“Define the Restricted Client-side Pool Administrator”](#) on page 73 for details about how to create this login.)
5. Click **OK** in the New Group Properties dialog box.

Create a second group named Restricted Client-side Pool Puddle Access. Add any users or groups that you want to be able to use the restricted client-side pool as members of this group. No logins are necessary.

Create a Restricted Workspace Server Client-side Pool

To create the restricted client-side pooling workspace server, follow these steps:

1. In the directory *SAS-configuration-directory*\SASApp—or SASMain—create a directory called **RestrictedPool**. Then, in the **RestrictedPool** directory, create a **logs** directory.
2. In the directory *SAS-configuration-directory*\SASApp**RestrictedPool**, create a configuration file that will be used when the restricted workspace server is started. The way in which you perform this step depends on whether the workspace server will run on a Windows host or a UNIX host.

Windows

In the directory *SAS-configuration-directory*\SASApp**RestrictedPool**, create a file named **sasv9.cfg**, and enter the following lines in the file: -
config "SAS-configuration-directory\SASApp\sasv9.cfg"

UNIX

In the directory, *SAS-configuration-directory*/**SASApp/RestrictedPool**, create a file named **workspaceServer.cfg**, and enter the following lines in the file: -**config !SASROOT/sasv9.cfg -config sasv9.cfg**

Later in the procedure, you will test the connection to the workspace server. If the test fails, you can remove the comments from the lines that relate to logging in order to enable logging. You can then repeat the test and check the workspace server log file for error messages.

3. Choose one of the following authentication methods for the workspace servers to use in connecting to the metadata server. Also, perform any tasks associated with the method that you choose.
 - If you use the TRUSTSASPEER object server parameter in your metadata server's configuration file (the default), then you can rely on that mechanism for workspace server authentication. The restricted pool workspace servers connect to the metadata server under the rpoolsrv identity when launched by SAS Web Report Studio and connect under the end user's identity when launched by a desktop application. In this mode, if you are working with an external DBMS, you must ensure that both the Restricted Client-side Pool Puddle Login group and any allowed individuals have database credentials.
 - You can also use the METAUSER and METAPASS options in the restricted client-side pool workspace server's configuration file. With this approach, the TRUSTSASPEER option is not required, and only the Restricted Client-side Pool Puddle Login group needs database credentials. For this approach, edit the configuration file that you created in step 2 to add these lines:

```
-metauser "rpoolsrv"
-metapass "encrypted-rpoolsrv-password"
```

On Windows systems, be sure to prepend a domain or host-name qualifier to the user ID. You can encrypt the password using PROC PWENCODE.

Note: The configuration file for the restricted workspace server must be locked down at the operating system level. The client-side pool administrator launches workspace servers under an operating system user ID of rpoolsrv. Workspace servers executed against this pool (such as those that are run by

the SAS Information Map Studio Test dialog box) run under the end-user ID. Remember to change this configuration file if site policy requires periodic changes on service accounts.

4. In SAS Management Console, define a new SAS Application Server, named **RestrictedPool**, that contains a workspace server.
 - a. Right-click **Server Manager**, and select **New Server** from the pop-up menu. The New Server Wizard starts.
 - b. On the wizard's first page, select **SAS Application Server** and click **Next**.
 - c. On the wizard's second page, enter the name **RestrictedPool** in the **Name** box and click **Next**.
 - d. On the wizard's third page, accept the default values and click **Next**.
 - e. On the wizard's fourth page, select **Workspace Server** and click **Next**.
 - f. On the wizard's fifth page, select the **Custom** radio button and click **Next**.
 - g. On the wizard's sixth page, enter the following values in the **Command** and **Object server parameters** boxes. The first command is appropriate for a workspace server that is running on a Windows host, and the second is appropriate for a UNIX host. The value that you enter in the **Object server parameters** field is not dependent on the operating system.

Command (Windows)

```
sas -config "SAS-configuration-directory\SASApp\RestrictedPool\sasv9.cfg"
```

Command (UNIX)

```
SAS-configuration-directory/SASApp/sas.sh -config  
RestrictedPool/workspaceServer.cfg
```

Then click **Next**.

- h. On the wizard's seventh page, specify the following values.

Authentication domain
Specify the same authentication domain that you used when you defined your first workspace server. By default, this will be **DefaultAuth**.

Bridge port
Change the default value, 8591, to the number of an unassigned port, such as 9591.
- i. On the wizard's eighth page, click **Finish**.
5. Update the metadata definition of your object spawner to indicate that the spawner should start processes for the new workspace server.
 - a. Right-click the icon that represents the spawner, and select **Properties** from the pop-up menu. A Spawner Properties dialog box appears.
 - b. Select the **Servers** tab.
 - c. Move **RestrictedPool - Workspace Server** from the list of **Available servers** to the list of **Selected servers**.
 - d. Click **OK**.
 - e. Restart your object spawner.
6. Test the connection to your new workspace server.
 - a. In the left pane of SAS Management Console, select **RestrictedPool - Workspace Server**. Information about a connection displays in the right pane.

- b. Right-click the icon representing the connection, and select **Test Connection** from the pop-up menu.
- c. If you are logged in to SAS Management Console as an unrestricted user, such as sasadm@saspw, you will be prompted for the credentials of a user who can start a workspace server. Enter the credentials for a user such as sasdmo. You should see a message indicating that the test was successful.

Note: If you happen to enter invalid credentials for a login, clear the credentials cache and the SAS Management Console will prompt you again to re-enter the credentials (**File** ⇒ **Clear Credentials Cache**.)

If the connection test fails, look at the log files for the object spawner and the new workspace server. The most likely cause of the problem is that you made a mistake in editing the configuration file for the new workspace server—the configuration file in the **RestrictedPool** directory.

7. Convert the new workspace server to client-side pooling.
 - a. Right-click **RestrictedPool - Logical Workspace Server**, and select **Convert To** ⇒ **Pooling** from the pop-up menu.
 - b. You are asked whether you want to continue. Click **Yes**. The Pooling Options dialog box appears.
 - c. In this dialog box, click **New** to bring up the New Puddle dialog box.
 - d. In the New Puddle dialog box, supply the following values:

Table 9.3 Defining a New Puddle

Field	Value
Name	restrictedPoolPuddle
Minimum available server	0
Minimum number of servers	0
Login	rpoolsrv
Grant access to group	Restricted Pool Puddle Access

Then click **OK**.

- e. Click **OK** in the Pooling Options dialog box.

Assign Libraries to the New Server

You must assign each library that you plan to access from the locked-down instance of SAS Web Report Studio to the server RestrictedPool. To assign each library, follow these steps:

1. Right-click the icon for the library, and select **Edit Assignments** from the pop-up menu. The Edit Assignments dialog box appears.

2. Hold down the CTRL key and click the list entry for **RestrictedPool**. (This action selects **RestrictedPool** and leaves items that are already selected in a selected state.) Then click **OK**.
3. Right-click the library again, and select **Properties** from the pop-up menu. The Library-name Properties dialog box appears.
4. Select the **Options** tab.
5. Click the **Advanced Options** button. The Advanced Options dialog box appears.
6. Select the **Library is pre-assigned** option, and click **OK**.
7. Click **OK** in the properties dialog box.

In the future, when you create information maps that you want to access from the locked-down instance of SAS Web Report Studio, make sure that you locate relational data sources by using the RestrictedPool server. Also, save these maps in a separate folder: `/BIP Tree/ReportStudio/RestrictedData/Maps`.

Create a Second SAS Web Report Studio Deployment

When you deploy SAS, the SAS Deployment Wizard creates an initial SAS Web Report Studio deployment using the inputs that you supply. Later, you can re-run the SAS Deployment Wizard to create additional deployments. For more information, see “Install and Configure SAS Interactively” in Chapter 5 of *SAS Intelligence Platform: Installation and Configuration Guide*.

Follow these guidelines when running the SAS Deployment Wizard to add another SAS Web Report Studio deployment:

- Use a deployment plan that contains the SAS middle tier.
- Choose the **Custom** configuration prompting level. You want to be able to access the configuration dialog box where you can set the SAS Metadata Server and the SAS Web Report Studio deployment instance name. The Express prompt level does not allow you to access these configuration settings.
- Define the SAS Metadata Server connection information for the metadata server that you are currently using.
- Use the correct machine name on which you are adding the SAS Web Report Studio deployment.
- Name the second SAS Web Report Studio deployment instance, `RestrictedDataReporting`.

Secure Sensitive Data Resources

Ensure that sensitive data resources are readable only by `rpoolsrv` (not `sassrv`) and the IT staff.

For target data that is in third-party databases, set up credentials in the metadata to enable the puddle account to access those servers. You can make credentials for a database server available to the puddle account by storing those credentials in a login as part of the Restricted Puddle Access group definition.

For example, to enable the puddle account to access a DB2 server, you would give the Restricted Puddle Access group a login that includes a DB2 user ID and password and that is associated with the DB2 server's authentication domain.

Note: Some members of your IT staff will also need to be able to authenticate to the database server.

Part 4

Server Administration

<i>Chapter 10</i>	
Managing SAS Application Servers	81
<i>Chapter 11</i>	
Managing Workspace Servers and Stored Process Servers	91
<i>Chapter 12</i>	
Managing the Object Spawner	105
<i>Chapter 13</i>	
Administering SAS OLAP Servers	125
<i>Chapter 14</i>	
System Options for SAS Application Server Components	139
<i>Chapter 15</i>	
IOMOPERATE Procedure	153

Chapter 10

Managing SAS Application Servers

Defining Multiple Application Servers	81
Add an Additional SAS Application Server	81
Scenario 1: Using SAS Data Integration Studio to Access Remote Data	82
Scenario 2: Using Multiple Application Servers with SAS Web Report Studio	83
Add a New Logical Server in an Existing SAS Application Server	84
Adding a New Server in an Existing Logical Server	85
Overview of Adding a New Server in an Existing Logical Server	85
Add a New Server in an Existing Logical Server	85
Modify a Server Definition	88
Remove Logical Servers	89

Defining Multiple Application Servers

Add an Additional SAS Application Server

When you deploy SAS, the SAS Deployment Wizard creates an initial SAS Application Server using the inputs that you supply. Later, you can re-run the SAS Deployment Wizard to create additional application servers on new machines. For more information, see Appendix 3, “Adding SAS Products,” in *SAS Intelligence Platform: Installation and Configuration Guide*.

Follow these guidelines when running the SAS Deployment Wizard to add a new SAS application server:

- The machine on which you want to add the new logical server should have a network connection to your site's SAS Software Depot. If it does not, then you must find a way to get the depot to the machine on which you want to create the new SAS application server.

For more information, see “Overview of SAS Software Depots” in Chapter 3 of *SAS Intelligence Platform: Installation and Configuration Guide*.

- Use your deployment plan that contains the SAS application server.

Note: There are standard deployment plans that contain a SAS application server machine. For more information, see “About Deployment Plans” in Chapter 5 of *SAS Intelligence Platform: Installation and Configuration Guide*.

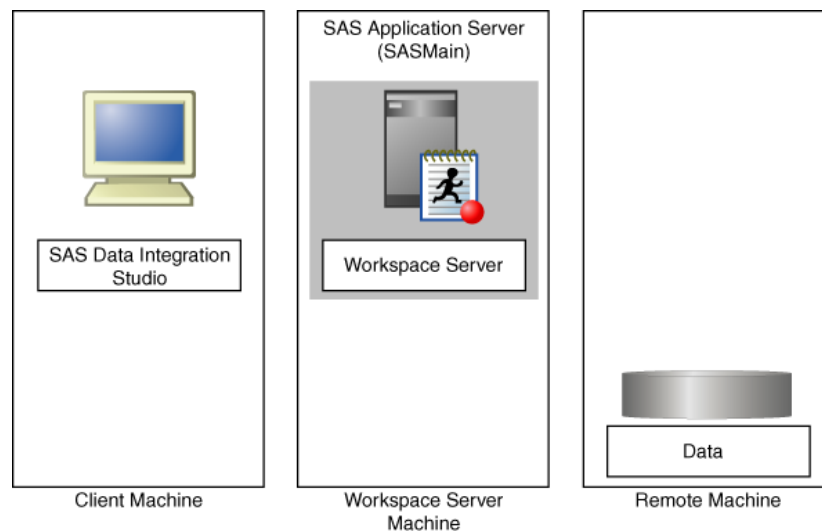
- If you are adding logical servers that require an object spawner (one of the workspace servers and the stored process server), then when you deploy your SAS application server, the wizard will also deploy an object spawner to start the servers. If the machine on which you are adding the SAS application server already contains an object spawner, then the wizard will update the pre-existing spawner definition for you to include the new servers that you want your spawner to manage.
- Specify a new server context (for example, SASApp2).
- Obtain the SAS Metadata Server connection information for the metadata server that you are currently using. The wizard will prompt you for this information.
- Specify the correct machine name on which you are adding the SAS application server.

There are several reasons why you might want to create a second application server. A couple of these reasons are discussed in the following sections.

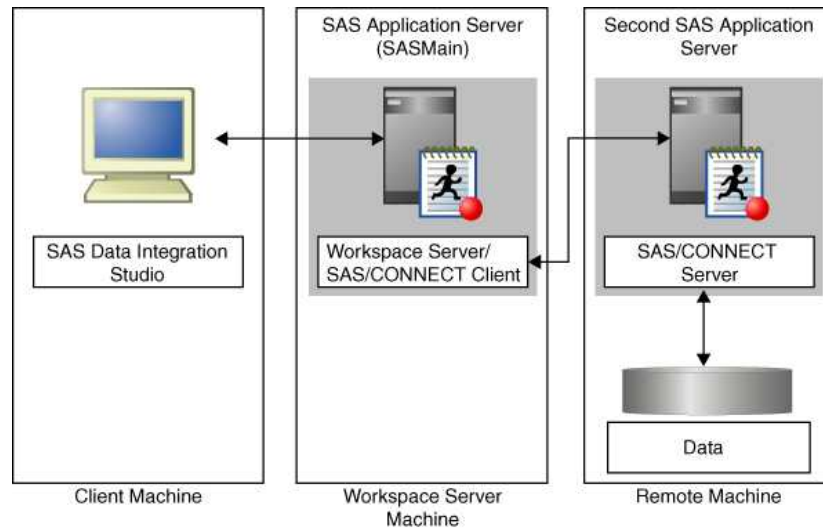
Scenario 1: Using SAS Data Integration Studio to Access Remote Data

Imagine that you are using SAS Data Integration Studio to process a large amount of data that resides on a machine different from the workspace server to which the application submits its code, as shown in the following figure.

Figure 10.1 Application Server and Data on Different Machines



One way to execute such a job efficiently is to define two application servers. One is the default application server for SAS Data Integration Studio and contains the workspace server to which the application will submit its generated code. The other application server contains a SAS/CONNECT server (which is collocated with the data), and the library of data to be processed is assigned to this application server. See the following figure.

Figure 10.2 Create an Application Server That Is Collocated with the Data

If you choose for the transformations in the job to be executed on the remote host, then SAS Data Integration Studio generates the code that is necessary for the transformations to be executed by the SAS/CONNECT server. The overall job is submitted to the workspace server. However, the workspace server then submits the code for each transformation to the remote server.

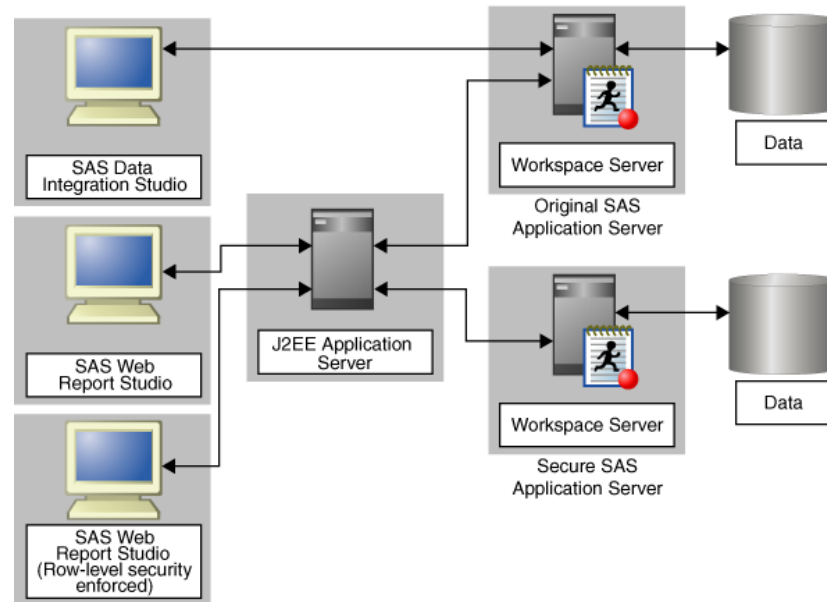
The generation of this type of code is made possible by the definitions of the two application servers, one of which is the default application server for SAS Data Integration Studio and the other the application server to which the data library is assigned.

Note: For more information, see “Setting Up Multi-Tier Environments” in Chapter 5 of *SAS Intelligence Platform: Desktop Application Administration Guide*.

Scenario 2: Using Multiple Application Servers with SAS Web Report Studio

Suppose that you want to create an environment in which row-level security can be strictly enforced for a set of SAS Web Report Studio users. (For a detailed description of this feature, see Chapter 1, “Overview of BI Row-Level Permissions,” in *SAS Guide to BI Row-Level Permissions*.)

Part of the setup is to create a special workspace server for use by the report creators who need the secure environment. This special workspace server is a component of a new SAS Application Server. See the following figure.

Figure 10.3 Using a Second Application Server to Enable Row-Level Security

The original workspace server can service users of other applications and users of SAS Web Report Studio whose access to data does not need to be so closely controlled.

Add a New Logical Server in an Existing SAS Application Server

A SAS Application Server can contain one of each of the following logical servers: workspace, stored process, OLAP, grid, CONNECT, batch, or metadata. Each logical server type must have at least one server defined.

Using the SAS Deployment Wizard, you can add a new logical server to a pre-existing SAS application server. For more information, see Appendix 3, “Adding SAS Products,” in *SAS Intelligence Platform: Installation and Configuration Guide*.

Follow these guidelines when running the SAS Deployment Wizard to add a logical server to an existing SAS application server:

- The machine on which you want to add the new logical server should have a network connection to your site's SAS Software Depot. If it does not, then you must find a way to get the depot to the machine on which you want to create the new logical server.

For more information, see “Overview of SAS Software Depots” in Chapter 3 of *SAS Intelligence Platform: Installation and Configuration Guide*.

- Use your deployment plan that contains the server component that you are adding.

Note: There are standard deployment plans that contain a SAS application server machine. For more information, see “About Deployment Plans” in Chapter 5 of *SAS Intelligence Platform: Installation and Configuration Guide*.

- Choose **Configure SAS Software** and uncheck **Install SAS Software**, unless you know that the server that you are adding was never installed before.

- Specify the server context to match the name of your current SAS Application Server (for example, SASApp).
- Obtain the SAS Metadata Server connection information for the metadata server that you are currently using. The wizard will prompt you for this information.
- If you are adding logical servers that require an object spawner (one of the workspace servers and the stored process server), then when you deploy your logical server, the wizard will also deploy an object spawner to start the server. If the machine on which you are adding the SAS application server already contains an object spawner, then the wizard will update the pre-existing spawner definition for you to include the new server that you want your spawner to manage.

Adding a New Server in an Existing Logical Server

Overview of Adding a New Server in an Existing Logical Server

Defining a new server in an existing logical server, consists of:

1. Installing the new server software on the new machine with the SAS Deployment Wizard.
2. If you are installing a server that requires an object spawner and there isn't already a spawner on the new machine, then when you install the server, you will also install and configure the object spawner.
3. Copying the necessary configuration directories from the existing logical server machine to the new machine.
4. Using the SAS Management Console to define the new servers in the metadata repository.

Add a New Server in an Existing Logical Server

You might decide that you need add another server in your logical server. The most common scenario is that you want to establish a server cluster to achieve load balancing. If you are adding one of the workspace servers or a stored process server then you will also need to add an object spawner to manage it, unless there is already a spawner on the machine. Follow these steps to add a new server and "if necessary" an object spawner to an existing logical server:

1. The machine on which you want to add the new server should have a network connection to your site's SAS Software Depot. If it does not, then you must find a way to get the depot to the machine on which you want to create the new server.

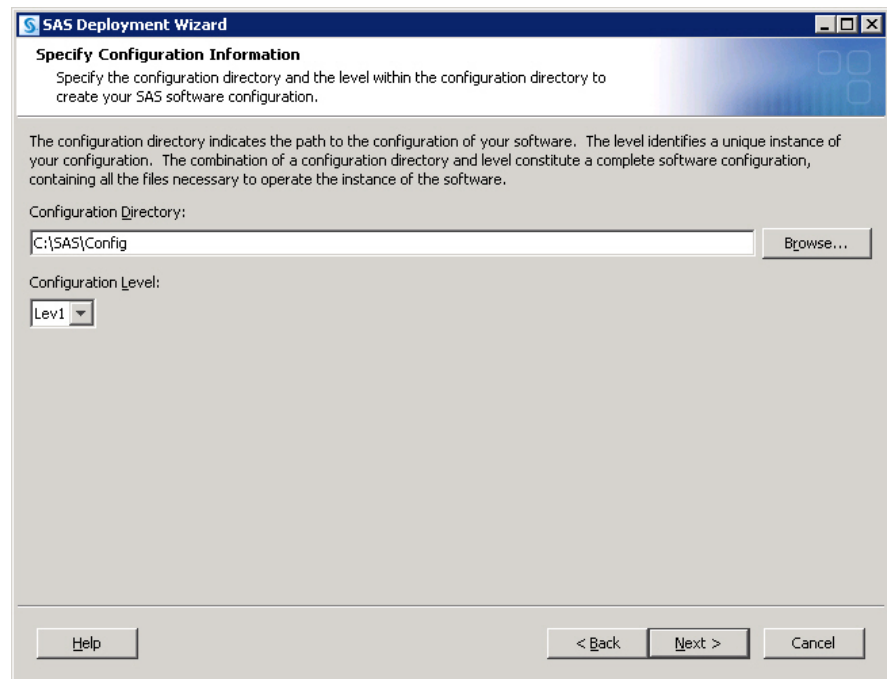
For more information, see "Overview of SAS Software Depots" in Chapter 3 of *SAS Intelligence Platform: Installation and Configuration Guide*.

2. Make sure that you have first installed the new server on the new machine by referring to Appendix 3, "Adding SAS Products," in *SAS Intelligence Platform: Installation and Configuration Guide*. When installing the new server, follow these guidelines:

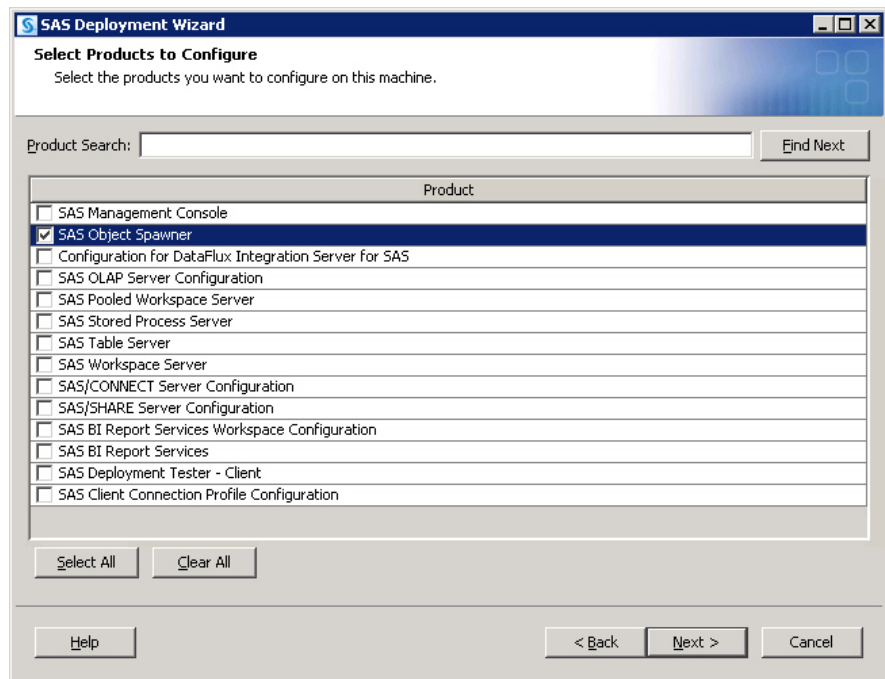
- a. If you are installing one of the workspace servers or a stored process server, then you will also need an object spawner to be on the new machine. Do one of the following:
- b. If you:
 - Need an object spawner:

then when installing the new server with the SAS Deployment Wizard, make sure that you choose both **Install SAS Software** and **Configure SAS Software**.
 - Don't need an object spawner:

then when installing the new server with the SAS Deployment Wizard, make sure that you choose **Install SAS Software** and deselect **Configure SAS Software**.
- c. When you see the wizard's Specify Configuration Information dialog box, make sure that the configuration directory that you enter matches the SAS configuration directory on the machine that contains the pre-existing SAS application server. For example, **C:\SAS\Config**.



- d. When you see the wizard's Select Products to Configure dialog box, check **SAS Object Spawner** only. Server configuration is described in later steps in this topic.



3. Create a configuration path on the new machine that matches the configuration path on the pre-existing machine.

For example, if the configuration path on machine 1 is `C:\SAS\Config\Lev1`, then create the same path on machine 2.

4. Copy the configuration directories of the pre-existing SAS application server to the new machine.

For example, on Windows, copy the following directory from machine 1 to machine 2:

C:\SAS\Config\Lev1\SASApp

5. Copy the configuration directories of any pre-existing logical servers to which you are adding additional servers, to the new machine.

For example, if you are adding another workspace server, copy the following directory from machine 1 to machine 2:

C:\SAS\Config\Lev1\SASApp\WorkspaceServer

6. Start SAS Management Console and connect to a metadata repository.
7. In the SAS Management Console navigation tree, select and expand the **Server Manager**, and locate the SAS application server under which you want to add the new server.
8. Expand the SAS application server and locate the logical server under which you want to add the new server.

Note: The logical server must be the same type of server as the server that you are adding. If there is no logical server, go to the following topic, [“Add a New Logical Server in an Existing SAS Application Server”](#) on page 84.

9. With the logical server selected, choose **Actions** ⇒ **Add Server** from the menu bar. The New Server Wizard displays.

10. Enter a name (required) and a description (optional) for the new server. The server name should indicate the machine, server type, and unique port on the new machine for the server. When you are finished, click **Next**.

The name that you provide will be the name of the server that displays in the SAS Management Console Server Manager plug-in.

11. Specify whatever configuration information is required for the type of server that you are adding. The number of wizard pages that you see and the information that you must supply, varies depending on the server type. Use the wizard online Help if necessary. For information about object server parameters, see [“OBJECTSERVERPARMS System Option” on page 142](#).

Note: To avoid a configuration error and failed client connections, you must enter user credentials when defining a pooled workspace and a stored process server.

12. When you see a wizard page that states that you have successfully completed a new server definition, review the information supplied and do one of the following:
 - click **Finish** to add the new server and close the New Server Wizard.
 - click **Back** to go to an earlier wizard page to change an input.

Modify a Server Definition

Using the SAS Management Console, you can modify the properties definition for a logical server or a server component. Examples of server properties are: the command used to start the server, load-balancing settings, and so on. For information about object server parameters, see [“OBJECTSERVERPARMS System Option” on page 142](#).

Follow these steps to modify a server definition:

1. Start SAS Management Console and connect to a metadata repository.
2. In the SAS Management Console navigation tree, expand the Server Manager to find the server object that you want to modify.
3. Select the server object, and then select **File** ⇒ **Properties** from the menu bar.
4. Select the appropriate tabs, and enter the necessary changes.

For a description of the server definition fields, see the SAS Management Console online Help.

5. When you are finished, click **OK** to return to the SAS Management Console main window.
6. If you changed any authorization permission on a workspace server, a pooled workspace server, or a stored process server, restart the object spawner.

For more information, see Chapter 5, “Operating Your Servers,” in *SAS Intelligence Platform: System Administration Guide*.

Remove Logical Servers

Using the SAS Deployment Manager, you can remove a logical server from a SAS application server.

Note: Removing logical servers on which libraries are dependent can prevent SAS applications from accessing data required for jobs, reports, and so on.

Follow these steps to remove a logical server definition:

Note: These steps attempt to summarize how to remove a logical server. For more detailed information, refer to Appendix 3, “Removing a SAS Configuration,” in *SAS Intelligence Platform: Installation and Configuration Guide*.

1. On the host machine for the components whose configurations you are removing, navigate to *SAS-installation-directory\SASDeploymentManager\9.3* and launch **config.exe** (on Windows systems) or **config.sh** (on UNIX and z/OS systems).
2. In SAS Deployment Manager, on the Select Configuration Task dialog box, choose **Remove Existing Configuration**.
3. On the next dialog box, choose the configuration directory and level that you want to modify.
4. On the Connection Information dialog box, specify a set of valid credentials for an unrestricted administrative user to connect to the metadata server .
5. On the Select Product Configurations dialog box, choose one or more logical servers that you want to remove from the SAS application server.
6. On the Summary dialog box, a list of the logical servers that you want to remove or unconfigure is listed. Do one of the following:
 - click **Start**"
This causes the deployment manager to remove the specified logical servers.
 - click **Back**"
 - Go back to an earlier dialog box to change a selection.
7. When the deployment manager has completed removing the logical server, click **Finish**.
8. If you removed a logical workspace server, a logical pooled workspace server, or a logical stored process server, restart the object spawner.

For more information, see Chapter 5, “Operating Your Servers,” in *SAS Intelligence Platform: System Administration Guide*.

Chapter 11

Managing Workspace Servers and Stored Process Servers

Managing Data and Catalogs for Servers on Multiple Machines	91
Overview of Managing Data and Catalogs for Servers on Multiple Machines	91
Update SAS Libraries	92
Update References to User-Defined Formats	92
Access Data in Database Management Systems	93
Adding or Modifying E-Mail Settings for SAS Application Servers	93
Overview of Adding or Modifying E-Mail Settings for SAS Application Servers .	93
Add or Modify E-Mail Settings for SAS Application Servers	93
Moving Workspace Servers and Stored Process Servers	94
Overview of Moving Workspace Servers and Stored Process Servers	94
Move Workspace Servers and Stored Process Servers	95
Required Tasks After You Move Workspace Servers and Stored Process Servers	97
Encoding and Locale Information	97
Adding Environment Variables to Server Invocations	98
Run SAS Code at Server Session Boundaries	98
Workspace Server Configuration Tasks	99
Tune Workspace Servers for Best Performance	99
Configure Workspace Servers for a Locale	100
Add System Options to the Workspace Server Launch Command	101
Configure Storage for Temporary OLAP Cube Build Files on SAS Workspace Servers	102
Clean Up Temporary Files after Abnormal OLAP Server Shutdowns	103

Managing Data and Catalogs for Servers on Multiple Machines

Overview of Managing Data and Catalogs for Servers on Multiple Machines

After placing load-balanced servers on multiple machines, you need to ensure that the new server can access the data, and possibly format catalogs, that the original server was working with. For example, suppose that your original server was using a library of SAS data sets and that the metadata object that represents the library contains the path `c:`

`\SAS\configuration-directory\SASApp\Data`. Your new server will not be able to access this library at its original location. One possible solution to this problem is to move a copy of the data to the new server and place it at the location that is stored in the library metadata object. However, this strategy might introduce data-synchronization problems. A preferable solution is to make sure that both servers can access a single copy of the data by providing a network path to the library.

Update SAS Libraries

As mentioned previously, when you scale your system, you need to update any SAS libraries that are being referenced with a local path. To update the definition of a library on a Windows machine that currently contains the path `C:\SAS\configuration-directory\SASApp\Data`, follow these steps:

1. In SAS Management Console, expand the **Data Library Manager** and the **SAS Libraries** folder so that you see the icon that represents your SAS library.
2. Right-click the library and select **Properties** from the pop-up menu that appears. A properties dialog box appears.
3. In the properties dialog box, select the **Options** tab.
4. Deselect the path that is currently selected by highlighting it in the **Selected items** list and clicking the left-arrow button.
5. Create a new path, and select it, by following these steps:
 - a. Click the **New** button to bring up the New Path Specification dialog box.
 - b. In the **Name** text box, enter the Universal Naming Convention (UNC) path to the library"for example, `\\D1234\SAS\Config\Lev1\SASApp\Data`. (Different machines on the LAN can use this same path to access the library.)
 - c. Click **OK**.
6. Click **OK** in the properties dialog box.

Update References to User-Defined Formats

Like SAS data sets, existing user-defined format catalogs might be available only to servers that run on your original SAS server host. It is common for a server to look for such catalogs in the configuration directory on the original server host in the directory `configuration-directory\SASApp\SASEnvironment\SASFormats`. It is also possible to specify the location of the catalog in a configuration file, as explained in Chapter 2, “Connecting to Common Data Sources,” in *SAS Intelligence Platform: Data Administration Guide*. However, the path recorded in such a file is often a local path.

One solution to this problem is to replicate the catalog on all server hosts, but this can be less than ideal if the catalog is subject to change. A better solution is to specify the location of the catalog in `configuration-directory\SASApp\sasv9.cfg` and to make sure that the path to catalog is a network path. For example, you might change

```
-set fmtlib1 "C:\SAS\configuration-directory\SASApp\Data\orformat"
-fmtsearch (fmtlib1.orionfmt)
```

to:

```
-set fmtlib1 "\\D1234\SAS\configuration-directory\SASApp\Data\orformat"
-fmtsearch (fmtlib1.orionfmt)
```

Access Data in Database Management Systems

If your new workspace or stored-process server needs to access data in a database management system (DBMS), then you might need to do some administrative work on the new server host before such access is possible. For example, you might need to:

- install a SAS/ACCESS product
- install database client software
- install a database driver
- configure a data source name

The simplest way to explain this is that you need to repeat whatever steps you took on the original SAS server host on the newly added server.

Adding or Modifying E-Mail Settings for SAS Application Servers

Overview of Adding or Modifying E-Mail Settings for SAS Application Servers

At installation time, the SAS Deployment Wizard prompts you for an e-mail server. This server is used by the SAS Metadata Server to send e-mail alerts to an administrator if journaling issues arise. The SAS Deployment Wizard also uses this e-mail server as the default for the SAS Application Server to provide e-mail services to various SAS clients. For example, with Data Integration Studio, you can use a Publish to Email transformation to alert users about various data changes. In order for SAS BI Dashboard to send alerts by e-mail to dashboard users and administrators, the port and host name must be configured for the e-mail server.

Add or Modify E-Mail Settings for SAS Application Servers

To change the application server e-mail server settings, follow these steps:

1. As a user with administrative privileges, log on to the SAS Application Server machine.
2. Using a text editor, open the application server configuration file.

By default the configuration file resides here:

- Windows:
SAS-configuration-directory\Levn\SASApp\sasv9_usermods.cfg
- UNIX and z/OS:
SAS-configuration-directory/Levn/SASApp/sasv9_usermods.cfg

3. Do one of the following:

- add e-mail settings:

add the following e-mail options: **-emailsys**, **-emailhost**, and **-emailport**, and save the file.

For more information, see “EMAILID= System Option” in *SAS System Options: Reference*.

- modify e-mail settings:

Locate the e-mail options (`-emailsys`, `-emailhost`, and `-emailport`), make the necessary changes, and save the file.

For more information, see “EMAILID= System Option” in *SAS System Options: Reference*.

The e-mail changes will take effect the next time the SAS application requests an e-mail service from the application server.

Moving Workspace Servers and Stored Process Servers

Overview of Moving Workspace Servers and Stored Process Servers

If you have installed a workspace server, pooled workspace server, or a stored process server on a machine separate from other SAS servers, you can use the instructions in the following topics to move the server to a new machine.

Note: If you are moving a server to a different machine, then the Update Host Name References feature of the SAS Deployment Manager might be useful. For more information, see Chapter 27, “Using the SAS Deployment Manager to Update Host Name References,” in *SAS Intelligence Platform: System Administration Guide*.

In addition to changing the machine name (and port number), if you move a server to a machine with a different operating system or to a machine with an operating system other than Windows, you might need to reconfigure the following:

- accounts for authentication. You might need to define accounts on the authentication provider for the new server machine.
- spawner start-up command. If you change operating systems when you move machines, you might need to change the spawner start-up command.
- metadata on the SAS Metadata Server. The following metadata definitions might require reconfiguration or additional configuration:
 - server definition. On the server definition, you might need to use the Server Manager plug-in to SAS Management Console to change the following parameters:
 - SAS start-up command. You might need to change the start-up command for the new operating system.
 - authentication domain. When you move a server, you might need to set up an additional authentication domain.
 - login definitions. For the login definitions that access the server and the login definitions that are used in the load-balancing configuration (for example, the SAS Guest user's login, if you performed an Advanced or Personal installation), you might need to use the User Manager plug-in to SAS Management Console to do one or more of the following:

- define a new login definition. When you move a server, you might need to create a new login definition for the new authentication domain.
- define a new login definition for a different authentication process. When you move a server, you might need to create a new login definition with credentials to access a server in a different operating system within the default authentication domain.
- change the format of the user ID in the login definition. When you move a server, you might need to change the fully qualified user ID for any login credentials used to access that server.
- stored process definitions. You might need to use SAS Management Console to specify a new location for your source code repository.

Move Workspace Servers and Stored Process Servers

To move a workspace, pooled workspace, or stored process server to a different machine, follow these steps:

1. Use SAS Management Console to reconfigure the server definition for the new machine:
2. In SAS Management Console, expand the Server Manager to locate the server definition for the specified server.
3. Right-click the icon for the server, and select **Properties** from the pop-up menu. A properties dialog box is displayed.
4. Click the **New** button. The Add New Machine dialog box is displayed.
5. In the **Host name** field, change the name to the host name of the new machine for your server, and then click **OK**. The new host name should display in the list of available servers.
6. If the server is no longer to be used on a machine, remove it from the **Selected** list. (Click the server name, and then click the left-pointing arrow button.)
7. If you need to change the server start-up command or add any object server parameters, make your changes in the appropriate fields. For more information, see SAS Management Console online Help and [“OBJECTSERVERPARMS System Option” on page 142](#).
8. Click **OK** to save the new configuration to the metadata repository.
9. If you need to change the authentication domain or the port that the new server uses, continue with the next step. Otherwise, skip to step 2.
10. Click the icon for the server, and in the SAS Management Console's right pane, select the **Connections** tab.
11. On the **Connections** tab, right-click the connection entry for the server, and select **Properties** from the pop-up menu. A connections properties dialog box is displayed.
12. Click the **Options** tab. In the **Bridge port** field, enter the port that you want the connection to use.
13. To change the authentication domain, click the **New** button. The New Authentication Domain dialog box is displayed.
14. In the **Name** field, enter the new authentication domain name for the server, and then click **OK**. The new authentication domain is displayed in the **Authentication domain** field.

15. Click **OK** to save the new configuration to the metadata repository.
16. Use SAS Management Console to reconfigure the spawner definition for the new machine:
 - a. In the SAS Management Console navigation tree, locate and select the spawner definition, and then right-click and select **Properties** from the pop-up menu.
 - b. If your spawner name contains the machine name, change the **Name** field to specify the name of the new machine.
 - c. Select the **Options** tab.
 - d. In the **Available** list, click the name of the new machine for your server, and then click the right-pointing arrow. The new machine name should display in the **Selected** list.
 - e. If the server is no longer to be used on a machine, remove it from the **Selected** list. (Click the server name, and then click the left-pointing arrow button.)
 - f. If you have any other servers associated with the spawner, select the **Servers** tab. In the **Selected servers** list box, select the other servers and move them to the **Available servers** list box. Click **OK**. You must then define a new spawner for these servers.
 - g. Click **OK** to save the new configuration to the metadata repository.
 - h. If you need to change the authentication domain or the spawner operation connection port that the new server uses, continue with the next step. Otherwise, skip to step 3.
 - i. Click the icon for the spawner definition, and in the SAS Management Console's right pane, select the **Connections** tab.
 - j. On the **Connections** tab, right-click the connection entry for the spawner, and select **Properties** from the pop-up menu. A connections properties dialog box is displayed.
 - k. Click the **Options** tab. In the **Port** field, enter the port that you want the connection to use.
 - l. If you changed the server's authentication domain, select the same new **Authentication domain** field for your spawner.
 - m. Click **OK** to save the new configuration to the metadata repository.
17. Edit the policy files for any applications that need to access the new server machine. (For details about editing policy files for the portal Web application and its components, see “Configuring and Deploying Restrictive Policy Files” in Chapter 3 of *SAS Intelligence Platform: Middle-Tier Administration Guide*.)
18. Install SAS 9.3 and SAS Integration Technologies on the new machine.
19. Copy your metadata configuration file (XML file) and spawner start-up script from your spawner configuration directory to the same directory on the new machine. If necessary, change the spawner start-up script for the new machine.
20. Ensure that the SAS Spawnee Servers user can authenticate against the host authentication provider for the machine.
21. Ensure that users who need to access the server are defined for the machine's host authentication provider.

Required Tasks After You Move Workspace Servers and Stored Process Servers

When you are finished modifying the server and spawner definitions, follow these steps:

- If you have added a new authentication domain for the machine, do both of the following:
 - Use the User Manager plug-in to SAS Management Console to add a login definition for access to the server.
 - Use the User Manager plug-in to SAS Management Console to modify the login definition for the multi-user login (for example, the SAS General Server group login, if you performed an Advanced or Personal installation). Modify the login definition to specify the new authentication, and, if required, the new user ID credentials.
- If you have changed operating systems and need to modify user credentials, use the User Manager plug-in to SAS Management Console to modify user and group login definition for the new user ID credentials of the new machine.
- If you need to move the stored process source code repositories to a different directory, use SAS Management Console to modify the stored process definition and change the **Source Code Repository** field on the **Execution** tab of the Stored Process Properties dialog box.
- If your stored process definitions reference content on the old stored process or workspace server machine, you must add the content to the directory that you defined in the stored process definition.
- Restart the object spawner.

For more information, see Chapter 5, “Operating Your Servers,” in *SAS Intelligence Platform: System Administration Guide*.

Encoding and Locale Information

For servers that are not Unicode (session encoding other than UTF-8), if your SAS server metadata contains characters other than those typically found in the English language, then you must be careful to start your SAS server with an ENCODING= or LOCALE= system option that accommodates those characters. For example, a SAS server that is started with the default US English locale cannot read metadata that contains Japanese characters. SAS will fail to start and will log a message that indicates a transcoding failure. In general, different SAS jobs or servers can run with different encodings (such as ASCII/EBCDIC or various Asian DBCS encodings) as long as the encoding that is used by the particular job or server can represent all of the characters for the data that is being processed. When first configuring your server, review the characters that are used in the metadata that describes your server (as indicated by the SERVER= objectserverparm) in order to ensure that SAS runs under an encoding that supports those characters.

For more information, see “Specifying a Locale” in Chapter 2 of *SAS National Language Support (NLS): Reference Guide* and “Setting the Encoding of a SAS Session” in Chapter 3 of *SAS National Language Support (NLS): Reference Guide*. See

also, “Reference: Configuration Files for SAS Servers” in Chapter 25 of *SAS Intelligence Platform: System Administration Guide*.

Adding Environment Variables to Server Invocations

You can add operating system environment variables and other shell-level functionality to a stored process server, workspace server, pooled workspace server, and OLAP server, by modifying the corresponding `_usermods.sh` (.bat) files provided in the server's configuration directory.

This can be useful for situations such as when you have to set an environment variable in order to access a third-party DBMS.

Because the `_usermods` files are sourced within each of the server wrapper scripts, the server inherits any logic or environment. SAS preserves `_usermods` files during software updates and migrations, unlike the server wrapper scripts, which SAS overwrites. For this reason, we discourage editing the wrapper scripts.

The `USERMODS_OPTIONS=` variable is used in the `_usermods` file to alter the SAS command line to invoke the server.

For more information, see “About Other SAS Server Configuration Files” in Chapter 2 of *SAS Intelligence Platform: System Administration Guide*.

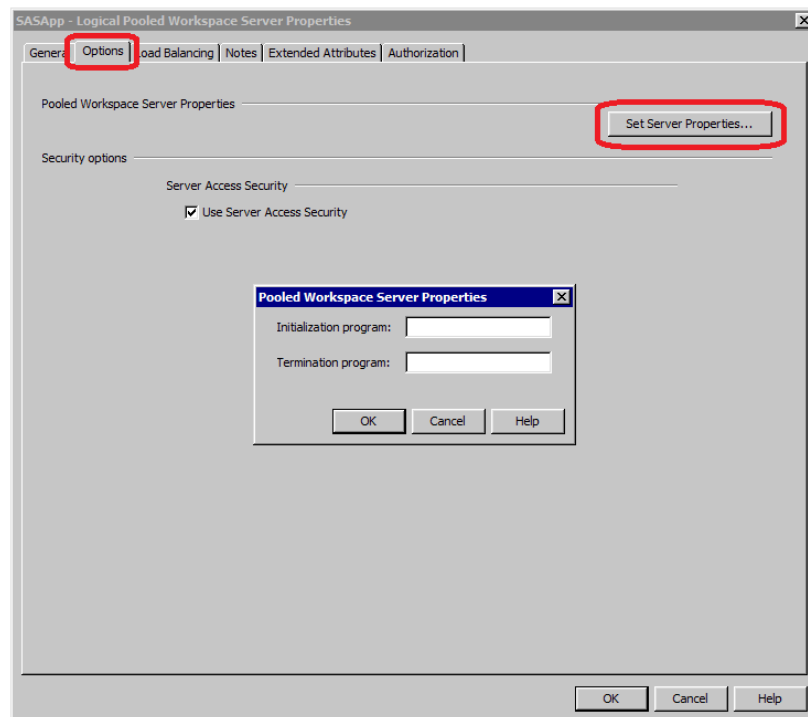
Run SAS Code at Server Session Boundaries

The stored process server and the workspace servers can be configured to run SAS code during start up and shutdown of their server *sessions*.

Note: In SAS 9.3, the IOM servers also support running SAS code at server start-up and shutdown (*server* boundaries). For more information, see “INITSTMT= System Option” in *SAS System Options: Reference* and “TERMSTMT= System Option” in *SAS System Options: Reference*.

For example, you might start and stop an ARM transaction to accurately trace a user's activity on a workspace server. Or, you might initialize certain SAS librefs for a particular user.

You configure this feature for the server from the Properties dialog box (**Properties** ⇒ **Options** ⇒ **Set Server Properties**).

Figure 11.1 Specifying SAS Code to Run at Server Start or Stop

In another example, you might want to configure your stored process server to run SAS code to override or change the value of the default SAS options before the execution of the requested stored process. This is a server-wide setting that runs before each execution.

Suppose you want to change the line size and page size. You can write a simple stored process that sets line size to 150 and page size to 30:

```
options ls=150 pagesize=30;
```

and locate the stored process containing your new line and page size options in a directory where the stored process server can read it.

In the **Initialization program** text box, you type the path and filename that contains your SAS options code.

For example:

```
C:\myOptions\setoptions.sas
```

Now, the stored process server runs your option override code before each requested stored process.

Workspace Server Configuration Tasks

Tune Workspace Servers for Best Performance

To obtain the best performance from your SAS Web applications, you should consider tuning the workspace server that your SAS Web application is using. The changes that you can make include specifying the following:

- an appropriate work folder
- a buffer size for writing files to the work area
- a limit on the total amount of memory that SAS uses at any one time

Note: For memory-intensive workloads on z/OS, you might need to adjust the user's USS RACF profile. For more information, see “Specify a Larger Region Size” in Chapter 22 of *SAS Companion for z/OS*.

The following table lists some of the SAS system options that you can set. “[Add System Options to the Workspace Server Launch Command](#)” on page 101 explains how to add the system options to the command that starts the workspace server.

Table 11.1 System Options for the Tuning the Workspace Server

System Option	Explanation
-RSASUSER	Opens the SASUSER library in read-only mode. Declaring this library read only makes the workspace server much faster for SAS Web Report Studio.
-work <i>work-folder</i>	Specifies the pathname for the directory that contains the Work data library. This directory should reside on a disk that emphasizes fast write performance.
-ubufsize <i>size-value</i>	Specifies a buffer size for writing files to the work area.
-memsize <i>size-value</i>	Specifies a limit on the total amount of memory that SAS uses at any one time.
-realmemsize <i>size-value</i>	Indicates the amount of RAM that is available to a process before it begins to page. Keeping this number low limits the amount of RAM that is consumed by a SAS server in order to reduce paging activity.
-sortsize <i>size-value</i>	Limits the amount of memory that can be used temporarily for sorting. Larger sort sizes reduce the use of the work folder, but increase the possibility of paging.
-cpucount <i>processors-number</i>	Specifies the number of processors that thread-enabled applications should assume will be available for concurrent processing. This setting maximizes the effectiveness of the SAS Web Report Studio sorting algorithm.

Note that the arguments to these options will be specific to the site and the job. Take care in choosing these values, and consult a SAS representative if necessary.

Configure Workspace Servers for a Locale

When SAS Web Report Studio is configured to run under certain locales, you must add the **-nosyntaxcheck** option to the start-up command for the workspace server. “[Add System Options to the Workspace Server Launch Command](#)” on page 101 explains how to add system options to the start-up command.

Note: Alternatively, you can create a new SAS Application Server (parallel to SASApp) and a new workspace server within the new application server context. You can then change the command (using the **-nosyntaxcheck** option) for the new workspace

server, without affecting the workspace server under SASApp. You can then assign selected libraries to the new application server, including any libraries that you intend to query for reports. See [“Configuring a Client-side Pooling Workspace Server to Enforce Row-Level Security” on page 72](#) for information about creating a new SAS Application Server and workspace server.

Add System Options to the Workspace Server Launch Command

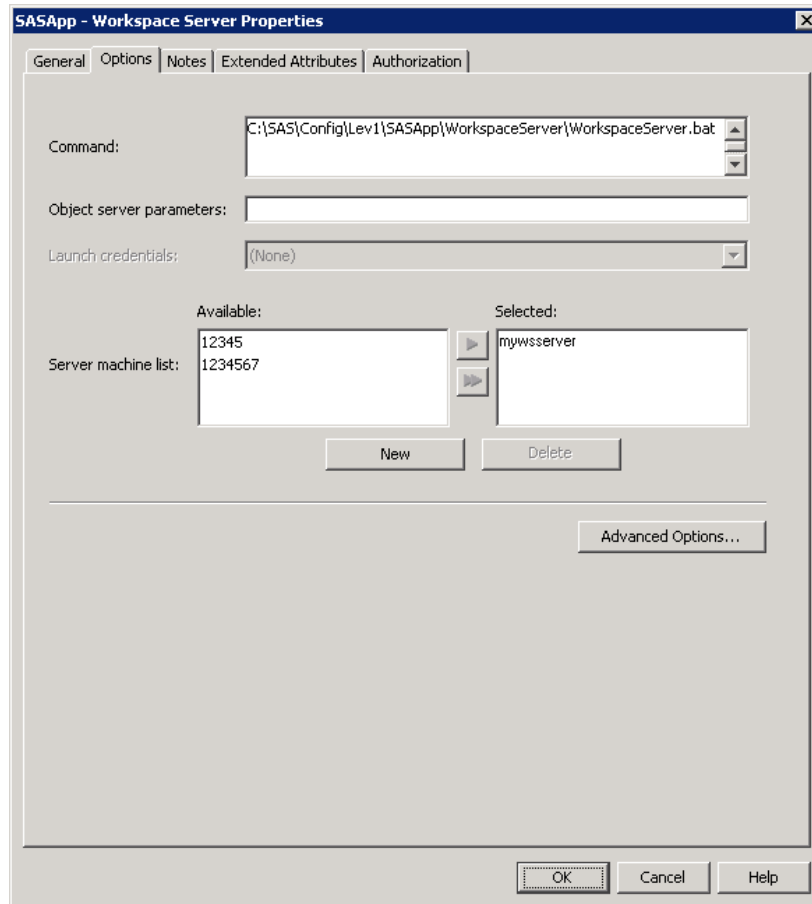
After you have determined the system options that you want to use to start your workspace server, follow the directions in this section to edit the `sas` command that starts the server.

Note: For information about workspace server logging options, see Chapter 9, “Administering Logging for SAS Servers,” in *SAS Intelligence Platform: System Administration Guide*.

Note: At the end of this procedure, you will have optimized your workspace server for use with SAS Web Report Studio. If you are using other applications and these applications can benefit from a workspace server that is configured differently, you must create a new logical workspace server (under SASApp) and add a workspace server to it.

To add system options to the workspace server launch command, follow these steps:

1. In SAS Management Console, expand the **Server Manager** node, and then expand the **SASApp - Logical Workspace Server** node. You will see a tree node that represents the physical workspace server.
2. Right-click the icon for the physical workspace server, and select **Properties** from the pop-up menu. A Workspace Server Properties dialog box appears.
3. Click the **Options** tab. You will see the information that is shown in the following display.



4. Edit the text in the **Command** text box, which by default is set to:

```
configuration-directory\SASApp\WorkspaceServer\WorkspaceServer.bat
```

For example, here is a command with options that improve performance for a workspace server:

```
configuration-directory\SASApp\WorkspaceServer\WorkspaceServer.bat
-rsasuser -work work-folder -ubufsize 64K -memsize 512M
-realmemsize 400M -sortsize 256M
```

5. Click **OK** in the Workspace Server Properties dialog box.

Configure Storage for Temporary OLAP Cube Build Files on SAS Workspace Servers

When you build a SAS OLAP cube, the SAS Workspace Server generates temporary utility files. At the end of the build, the server deletes the temporary files. If the build process terminates abnormally, remove the temporary files as described in [“Clean Up Temporary Files after Abnormal OLAP Server Shutdowns”](#) on page 103 .

Follow these steps to allocate sufficient storage for temporary cube build files:

1. For a SAS Workspace Server that will be used to build cubes, determine the storage capacity that will be required for temporary build files. In general, temporary files can be expected occupy two to three times the disk space that will be used to store the source data of the largest cube that will be built by that server.

2. Divide in half the estimated storage capacity. Locate two suitable disks on the SAS Workspace Server. The disks need to retain, over time, an amount of free space that is equal to half of the estimated storage capacity for temporary build files.
3. In the launch command for the SAS Workspace Server, specify a value for the WORK= system option that consists of a path to one of the two disks. To allocate the other half of the storage requirement, specify a path to the second disk in both of the system options SPDEUTILLOC= and UTILLOC=. To learn how to add system options to the launch commands of SAS Workspace Servers, see [“Add System Options to the Workspace Server Launch Command” on page 101](#) .
4. Run test builds to validate and adjust your storage estimate.
5. If your cubes are expected to grow in size over time, periodically reassess and reassign system options to provide sufficient storage capacity.
6. Either ask your cube builders not to assign the WORKPATH= option in the PROC OLAP statement, or to assign that option with a value that includes the two paths that you specified for WORK= and SPDEUTILLOC=/UTILLOC=.
7. Be sure to allocate sufficient storage capacity on all SAS Workspace Servers that will be used to build SAS OLAP cubes.

For additional information about the memory and storage capacity of SAS Workspace Server, contact your SAS support representative.

Clean Up Temporary Files after Abnormal OLAP Server Shutdowns

When a SAS Workspace Server completes a cube build, it deletes the temporary files that it created during the build. If a SAS Workspace Server terminates incorrectly, some temporary files might remain. To retain the full amount of temporary storage space, delete the temporary files before your next cube build.

To delete temporary files, obtain the directory paths that are specified in the SAS OLAP Server configuration file for the system options WORK=, SPDEUTILLOC=, and UTILLOC=.

Chapter 12

Managing the Object Spawner

Object Spawner Configuration Tasks	105
Overview of Object Spawner Configuration Tasks	105
Define a New Object Spawner	106
Add a Connection to the Object Spawner	106
Modify an Object Spawner Definition	107
Change Object Spawner-Managed Server Ports	107
Configure Object Spawner Logging	108
Configure the Object Spawner to Accept Single Sign-on Connections	109
Refresh the Object Spawner	110
Update a Windows Object Spawner Service	111
Using Telnet to Administer the Spawner	111
Configuring and Starting the Object Spawner on z/OS	113
Overview of Configuring and Starting the Object Spawner on z/OS	113
Task 1: Configure TCP/IP	113
Task 2: Create the Object Spawner Started Task	113
Task 3: Create a SAS Start-up Command	115
Spawner Invocation Options	117
Overview of Spawner Invocation Options	117
General Options	117
Metadata Connection and Security Options	121
Service Options	122

Object Spawner Configuration Tasks

Overview of Object Spawner Configuration Tasks

This section contains the following object spawner configuration topics:

- [“Define a New Object Spawner” on page 106](#)
- [“Add a Connection to the Object Spawner” on page 106](#)
- [“Modify an Object Spawner Definition” on page 107](#)
- [“Change Object Spawner-Managed Server Ports” on page 107](#)
- [“Configure Object Spawner Logging” on page 108](#)
- [“Configure the Object Spawner to Accept Single Sign-on Connections” on page 109](#)
- [“Refresh the Object Spawner” on page 110](#)

- “Update a Windows Object Spawner Service” on page 111
- “Using Telnet to Administer the Spawner” on page 111

Define a New Object Spawner

Workspace servers, pooled workspace servers, and stored process servers are initialized by the SAS Object Spawner. An object spawner runs on each machine where you want to run one of the workspace servers or a stored process server, listens for requests, and launches the servers as necessary. If your machine already contains an object spawner, you can modify its definition to include new servers to manage.

If your machine is new to your deployment and you new to add a new object spawner, see “Add a New Server in an Existing Logical Server” on page 85.

Add a Connection to the Object Spawner

To add additional server connections to an object spawner by using the SAS Management Console, follow these steps:

1. Start SAS Management Console and connect to a metadata repository.
2. In the SAS Management Console navigation tree, select and expand the **Server Manager**, and locate the spawner object that you want to modify.
3. Right-click the icon for the object spawner and select **Add Connection** from the pop-up menu.

The New Connection Wizard is displayed.

4. Select one of the following:

- **UUID Connection:**

specifies that communications between the client and the spawner use the UUID protocol.

Note: Entering a UUID node is required on UNIX and z/OS hosts.

- **PortBank Connection:**

defines a connection that the spawner uses to allow clients to connect to and disconnect from a spawned server-side pooled workspace server.

Note: To connect to a logical pooled workspace server, the spawner must have at least one PortBank connection defined.

5. Follow the remaining prompts. Click **Next** to proceed or **Back** to change any input.
For more information, refer to SAS Management Console online Help.
6. When you see a wizard page that states that the following connection will be created, review the information that is supplied and do one of the following:
 - Click **Finish** to create the new connection and close the New Server Wizard.
 - Click **Back** to go to an earlier wizard page to change an input.
7. Restart the object spawner.

For more information, see Chapter 5, “Operating Your Servers,” in *SAS Intelligence Platform: System Administration Guide*.

Modify an Object Spawner Definition

To modify an object spawner definition by using the SAS Management Console, follow these steps:

1. Start SAS Management Console and connect to a metadata repository.
2. In the SAS Management Console navigation tree, expand the Server Manager to find the object spawner that you want to modify.
3. Right-click the icon for the object spawner and select **Properties** from the pop-up menu.
4. Select the appropriate tabs, and enter the necessary changes.

For more information, refer to SAS Management Console online Help.

5. When you are finished, click **OK** to return to the SAS Management Console main window.
6. Restart the object spawner.

For more information, see Chapter 5, “Operating Your Servers,” in *SAS Intelligence Platform: System Administration Guide*.

Change Object Spawner-Managed Server Ports

For most of the SAS Object Spawner managed ports, the SAS Deployment Wizard designates a unique port number by default. The following table shows these default values.

Table 12.1 Defaults for Object Spawner-Managed Ports

Port	Default
Spawner operator port	8581
Workspace server port	8591
Pooled workspace server port	8701
Stored process server bridge port	8601

With the exception of the operator port, these various spawner processes can all use the same port. Using the SAS Management Console, you can change these default port designations and reduce the number of ports that need to be defined in SAS metadata and opened up through your firewall.

Note: You cannot use same port number for the spawner port bank and the stored process server MultiBridge ports. Port bank and MultiBridge ports all must use unique ports.

To change object spawner-managed server ports, follow these steps:

1. In SAS Management Console, expand the Server Manager to locate the server definition, or the object spawner definition.

2. Click the icon for the server or for the spawner, and in the SAS Management Console's right pane, select the **Connections** tab.
3. On the **Connections** tab, right-click the icon for the connection, and select **Properties** from the pop-up menu. A connection properties dialog box appears.
4. In the connection properties dialog box, click the **Options** tab.
5. In the **Port** field, enter the port that you want the connection to use and then click **OK**.
6. Restart the spawner for the port changes to take effect.

For more information, see Chapter 5, “Operating Your Servers,” in *SAS Intelligence Platform: System Administration Guide*.

Configure Object Spawner Logging

For users with administer permissions on the spawner, you can view the SAS Object Spawner log in the SAS Management Console by first opening a connection to the object spawner and then clicking the **Log** tab. For more information, see “Using SAS Management Console to Monitor SAS Servers” in Chapter 8 of *SAS Intelligence Platform: System Administration Guide*.

You can also view spawned server activity in real time by using the Spawned Server Activity tab also in the SAS Management Console. For more information, see “Use the Spawned Server Activity Tab in the Server Manager” in Chapter 8 of *SAS Intelligence Platform: System Administration Guide*.

By default, the object spawner logs activity at the **information** level. You can temporarily adjust this logging level for a spawner session through the SAS Management Console, or more permanently by editing the spawner logging configuration file.

CAUTION:

Excessive logging can degrade performance. You should not use the TRACE and DEBUG logging levels unless you are directed to do so by SAS Technical Support.

- To temporarily adjust logging for the spawner session:

Select the **App.ObjectSpawner** logger in the **Loggers** tab, and edit the logger properties. For more information, see “Use the Loggers Tab in the Server Manager, and (If Necessary) Change Logging Levels for Individual Loggers” in Chapter 8 of *SAS Intelligence Platform: System Administration Guide*.

- To adjust logging for more than one spawner session:

Edit the spawner logging configuration file and add the following information:

```
<logger name="App">
    <level value="level"/>
</logger>
```

where *level* is a value such as **trace**. For more information, see “Use the Loggers Tab in the Server Manager, and (If Necessary) Change Logging Levels for Individual Loggers” in Chapter 8 of *SAS Intelligence Platform: System Administration Guide*.

Restart the object spawner.

For more information, see Chapter 5, “Operating Your Servers,” in *SAS Intelligence Platform: System Administration Guide*.

By default, the spawner logging configuration file resides here:

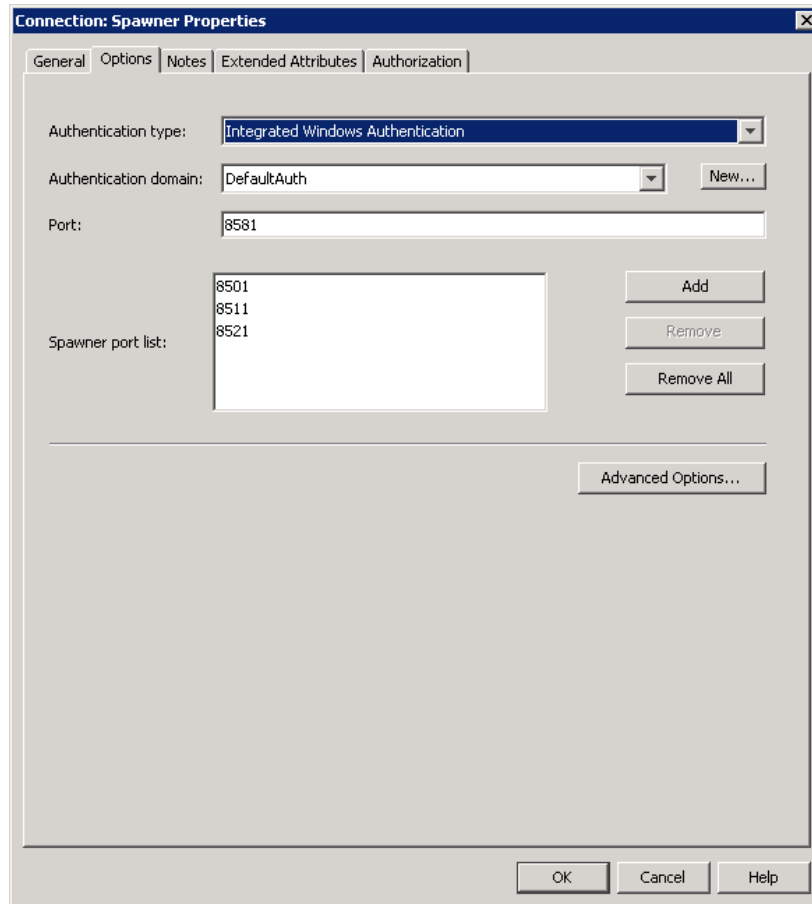
- Windows:
`configuration-directory\ObjectSpawner\logconfig.xml`
- UNIX:
`configuration-directory/ObjectSpawner/logconfig.xml`

Configure the Object Spawner to Accept Single Sign-on Connections

On Windows machines on which the SAS Object Spawner runs, you can configure the spawner to accept Integrated Windows Authentication (IWA) connections from SAS clients.

To configure the object spawner to accept single sign-on connections, follow these steps:

1. In the SAS Management Console, expand the **Server Manager** node and then click on the **Object Spawner** node.
2. In the right pane of the SAS Management Console, click the **Connections** tab.
3. Right-click **Connection: Spawner**, and then select **Properties** from the pop-up menu. A spawner properties dialog box is displayed.
4. Select the **Options** tab. You will see the information that is shown in the following display.



5. In the **Authentication type** drop-down list, choose **Integrated Windows Authentication**, and then click **OK**.
6. Restart the object spawner.

For more information, see Chapter 5, “Operating Your Servers,” in *SAS Intelligence Platform: System Administration Guide*.

Refresh the Object Spawner

When you refresh the object spawner, you reinitialize the spawner and force it to reread its configuration in the metadata. As part of this refresh, the spawner quiesces any servers that it has started. The servers shut down when their clients have completed their work. If changes are made to the server or spawner configurations, the spawner can be refreshed in order to pick up and apply these new changes.

To refresh an object spawner, follow these steps:

1. For users with administer permissions on the spawner, in the SAS Management Console, expand the **Server Manager** node and then right-click on the **Object Spawner** node.
2. From the pop-up menu choose **Connect**.
3. Right-click on the **Object Spawner** node again, and from the pop-up menu choose **Refresh Spawner**.
4. In the confirmation dialog box, click **Yes**.

Update a Windows Object Spawner Service

To update an existing Windows service for the spawner (for example, to change the path to your metadata configuration file), you must remove the service and create a new one.

To update a Windows object spawner service, follow these steps:

1. Stop the spawner service by using the Stop SAS Object Spawner shortcut in the Windows Start menu.
2. Remove the spawner service by invoking `ObjectSpawner.bat -remove`. This file is located in the `SASApp\ObjectSpawner` subdirectory of your SAS Configuration Directory. For example, `C:\SAS\MyDeployment\Lev1\SASApp\ObjectSpawner`.
3. Open the `ObjectSpawner.bat` file in an editor, and then add your options to the install and start commands.
4. Save the file.
5. Reinstall the spawner service by using the command `ObjectSpawner.bat -install`.

Using Telnet to Administer the Spawner

Overview to Using Telnet to Administer the Spawner

The object spawner can be controlled and monitored using a Telnet client connected to the operator port or service.

Enable Telnet Access to an Object Spawner

For security reasons, Telnet access to the object spawner is disabled by default. However, using the SAS Management Console, you can enable Telnet access.

To enable Telnet access to an object spawner, follow these steps:

1. In SAS Management Console, select and expand the Server Manager to locate the object spawner for which you want to enable Telnet access.
2. Right-click the object spawner for which you want to enable Telnet access, and choose **Properties**.
3. In the Properties dialog box, choose the **Initialization** tab, and then uncheck **Disable telnet access**.
4. In the **Operator login** drop-down list, choose a login to verify users that connect to the spawner's operator port via the Telnet interface.

Note: For more information about using the interface to choose an operator login, refer to SAS Management Console online Help.

5. Once you have chosen a login, click **OK** to save your changes and to close the Properties dialog box.
6. Restart the object spawner.

For more information, see Chapter 5, “Operating Your Servers,” in *SAS Intelligence Platform: System Administration Guide*.

Connect to an Object Spawner

To connect to an executing object spawner, Telnet to the operator interface port or service that is specified in the spawner definition.

The following example, run on UNIX, assumes that **8581** was specified as the port for the operator:

```
myHost> telnet serverhost 8581
Trying...
Connected to serverhost.
Escape character is '^]'.

```

After the Telnet conversation is active, enter the operator password that is specified. If the operator password was not specified, use **sasobjspawn** as the password.

Note: You will not be prompted for the password. For example:

```
sasobjspawn
Operator conversation established

```

You can now interact with the executing spawner by issuing any of the “[Commands for Spawner Operator Interface](#)” on page 112.

Commands for Spawner Operator Interface

The following is a list of commands that are available via the spawner's operator interface:

Table 12.2 Commands for Spawner Operator Interface

Command	Description
bye	Terminates the spawner execution. <i>Note:</i> You cannot shut down an object spawner while there are current or pending load-balancing tasks.
help	Lists available operator commands.
quit	Exits operator conversation.
refresh	Reinitializes the spawner. The spawner rereads its configuration out of the metadata. As part of this refresh, the spawner shuts down any servers that it currently has started. If changes are made to the server or spawner configurations, the spawner can be refreshed in order to pick up and apply these new changes.

Configuring and Starting the Object Spawner on z/OS

Overview of Configuring and Starting the Object Spawner on z/OS

On a z/OS server, the object spawner starts a SAS server session in response to a request from a client. The client uses TCP/IP to communicate first with the spawner, and then with the object server. The object spawner runs as a started task. Therefore, before the object spawner can handle client requests, you must start the spawner by using a started task procedure.

If you used the SAS Deployment Wizard to deploy the Intelligence Platform, then you already have an initial z/OS spawner configuration.

If you did not use the SAS Deployment Wizard, then the following setup tasks are required:

1. Configure TCP/IP.
2. Create the object spawner started task.
3. Create a SAS start-up command.

Note: This topic is intended to serve as an outline of the process, rather than a step-by-step guide, for setting up a spawner on a z/OS platform.

Task 1: Configure TCP/IP

The overall configuration of TCP/IP is outside the scope of this discussion. Assuming that a functioning TCP/IP link is in place between the client and the z/OS server, verify that the TCP/IP SERVICES configuration is available to both the object spawner and its object servers. For more information, see the *Configuration Guide for SAS 9.3 Foundation for z/OS*.

If your TCP/IP site requirements include customization of `/etc/services` (or equivalent) for applications that listen on well-known ports, it will be necessary to add object spawner definitions to the `/etc/services` file. To define these in the TCP/IP services file, add the following two lines:

```
sasobjoper      8582/tcp
sasobjspawn    8581/tcp
```

Task 2: Create the Object Spawner Started Task

Overview of Create the Object Spawner Started Task

The object spawner runs as a started task (STC). Its purpose is to listen for requests from clients and pass them to the start-up command that is associated with the service and port in which there is activity. The start-up command will start a server session. You must create a procedure in a system PROCLIB library (SYS1.PROCLIB, for example).

Create the Procedure

Because z/OS Job Control Language has a parameter line length restriction of 100 characters, you can use DDNames to identify filenames in object spawner parameters. When a file pathname is eight characters or less, the file pathname is first checked to see whether it matches a DDName. If so, the DDName is used. If DDNames are not used for the configuration file and a log file, you need to specify a configuration file and a log file in the UNIX file system.

If you need to specify more than 100 characters for command-line parameters, put the additional parameters in a z/OS data set or UNIX file. Reference the data set or file by using the =<//DDN:PARMS parameter.

The following procedure explicitly specifies the pathname for the configuration file and uses a DDName to reference the log file in the command line parameters for the object spawner.

```
//OBJSPAWN PROC PROG=OBJSPAWN,
//  OPTIONS='-XMLCONFIGFILE /usr/lpp/SAS/objspawn.xml',
//  OPT2='-LOGCONFIGLOC /usr/lpp/SAS/logconfig.xml'
//OBJSPAWN EXEC PGM=&PROG,REGION=512M,
//          PARM='&OPTIONS &OPT2 =<//DDN:PARMS'
//STEPLIB DD DISP=SHR,DSN=SYS2.SAS.LIBRARY
//PARMS   DD DISP=SHR,DSN=SYS2.OBJSPAWN.PARMS
//TKMVSENV DD DISP=SHR,DSN=SYS2.OBJSPAWN.TKMVSENV
//TKMVSJNL DD PATH='/tmp/objspawn/JNL.&LYYMMDD.&LHHMMSS..txt',
//          PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIROTH),
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC)
```

The -XMLCONFIGFILE parameter identifies the SAS Metadata Server system configuration file that the spawner is to use.

The -LOGCONFIGLOC parameter identifies the file for configuring the logging of useful information for diagnosing connection problems. It is a good idea to include logging until you are satisfied that everything is working correctly.

Define the Object Spawner System Security Configuration

The z/OS system considers the object spawner a daemon process. Therefore, if the BPX.DAEMON profile of the RACF Facility class is active and RACF program control is enabled, then the SAS load library that is specified in the STC procedure must be program controlled. However, the user ID under which the object spawner runs does not require RACF READ access to the BPX.DAEMON profile.

If the following messages appear in the z/OS system log when a client attempts to connect, then a necessary library is not program controlled.

```
ICH420I  PROGRAM program-name [FROM LIBRARY dsname] CAUSED THE
          ENVIRONMENT TO BECOME UNCONTROLLED.
BPXP014I ENVIRONMENT MUST BE CONTROLLED FOR DAEMON (BPX.DAEMON) PROCESSING
```

Verify the Metadata Configuration File and SAS Management Console Definitions

You must also define your server and spawner by using SAS Management Console. When you define the server, enter the following command in the **Command** field of the **Advanced > Options > Launch Commands** tab:

```
/usr/lpp/SAS/SAS_9.3/startsas.sh
```

Start the Object Spawner

After you have created the STC procedure, you can start the object spawner by issuing the following command:

```
START OBJSPAWN
```

For a list of all available spawner invocation options, see [“Spawner Invocation Options” on page 117](#). If there are no configuration errors, the object spawner will assume a listening state by entering a detected wait state (DW).

Task 3: Create a SAS Start-up Command**Overview of Create a SAS Start-up Command**

The start-up command is meant to build a parameter string that is capable of launching SAS. Here is a sample shell script (`startsas.sh`):

```
#!/bin/sh
#
# Initialize SAS startup command...
#
cmd="/bin/tso -t %SASTREXX \
-sasrxsysconfig '&prefix.SASRXCFG(ENWONO)'"

#
# Quote arguments with embedded blanks
#
Quoteme() {
    if [ $# -gt 1 ]; then
        quoteme="\ "$*\ ""
    else
        quoteme=$1
    fi
}
for arg in "$@" ; do
    Quoteme $arg
    tmp="$quoteme"
    cmd="$cmd $tmp"
done

#
# pass the metadata authentication strings to SAS
#
if [ -n "$METAUSER" ] ; then
    cmd="$cmd -metauser \"$METAUSER\""
fi
if [ -n "$METAPASS" ] ; then
    cmd="$cmd -metapass \"$METAPASS\""
fi

#
# Set additional environment variables...
# SYSPROC specifies the data set containing the SAS REXX
#
export SYSPROC=&prefix.SASRX
export STEPLIB=
export TSOOUT=
```

```
#
# Start SAS
#
exec $cmd
```

The sample invokes the `/bin/tso` UNIX command to execute the REXX exec `SASTREXX`. Replace the REXX exec data set name `&prefix.SASRX` in the `SYSPROC` environment variable with the data set name that is appropriate for your site. The `SASTREXX` exec should be invoked with the following parameter, which you can specify in metadata on the command line to launch the server:

```
NOSASUSER
```

`NOSASUSER` allows more than one concurrent SAS session per user. `NOSASUSER` suppresses allocation of a `SASUSER` data set.

Specify Account Data

The IOM spawner on z/OS uses the UNIX System Services `spawn` function to initiate a process to run an IOM server. This process runs in a USS initiator (`BPXAS`). By default, the process runs with the default Work Load Manager (WLM) service class that was assigned to OMVS work during installation. The default service class might have been defined with a goal of providing USS shell commands with good response times. This default service class assumes that the requests are relatively short. Because work associated with IOM requests might require more time, it might be desirable to assign IOM servers to a different service class.

You can use MVS accounting data to assign the work to a specific Work Load Manager service class. To set the accounting data, use the `_BPX_ACCT_DATA` environment variable in the `startsas.sh` script that starts that SAS IOM server session. The server session then runs with the accounting data. For example:

```
export _BPX_ACCT_DATA=MYNAME1
```

To assign a Work Load Manager service class based on the accounting data, use the WLM AI classification rule. For example (in the WLM ISPF dialog box):

		Qualifier		Class	
Type	Name	Start	Service	Report	
			DEFAULTS:	OMVSSHRT	_____
1	AI			OMVSLONG	_____

For more information about using accounting information with USS processes, defining WLM service classes with appropriate characteristics, and specifying classification rules to use these classes, see your IBM documentation.

Because you might define different IOM servers, in order to separate different work loads, you can also specify that these servers run in different service classes. To specify different service classes, create a separate server definition for each class of work in the SAS Management Console configuration, and assign client requests to the listen port that is associated with each server.

Spawner Invocation Options

Overview of Spawner Invocation Options

The object spawner controls the execution of the workspace server and the stored process server through an IOM bridge connection. Spawner invocation options consist of options that are specific to the spawner and several SAS system options that you can use to run and configure the object spawner from the command line. On Windows and UNIX, when you install and configure SAS servers with the SAS Deployment Wizard, a spawner batch or script file is created by default in the object spawner configuration directory (`ObjectSpawner.bat` or `objectspawner.sh`).

For more information about the various object spawner configuration files and their purpose, see “Configuration Files for SAS Object Spawners and SAS/CONNECT Spawners” in Chapter 25 of *SAS Intelligence Platform: System Administration Guide*.

On z/OS, the spawner is run as a started task. If you want to create a SAS start-up command, see the section, “Configuring and Starting the Object Spawner on z/OS” on page 113 .

Note: On z/OS and UNIX, to set the object spawner temp directory, update the `TKOPT_ENV_UTILLOC` environment variable with the new path. For more information, “TKMVSENV File” in Chapter 1 of *SAS Companion for z/OS* or your UNIX documentation.

The object spawner executable resides by default in:

- Windows:

`install-dir\SASFoundation\9.3\objspawn.exe`

- UNIX:

`install-dir/SASFoundation/9.3/utilities/bin/objspawn`

The spawner must be refreshed through the SAS Management Console in order to reflect configuration updates. The SAS Management Console provides an interface to refresh the spawner with configuration changes. For more information, see “Using SAS Management Console to Operate SAS Servers” in Chapter 5 of *SAS Intelligence Platform: System Administration Guide*. If your spawner is configured as a Windows service, then you must redefine the service to change the invocation options. For more information, see “Update a Windows Object Spawner Service” on page 111.

Spawner invocation options can be logically grouped into these categories:

- general options
- metadata connection options
- service options

General Options

Overview of General Options

Use the following general options for identifying which object spawner to invoke and for setting spawner options such as security and logging:

- [-dnsmatch](#) on page 118
- [-dnsname](#) on page 118
- [-hostknownby](#) on page 118
- [-sasspawnercn](#) on page 118
- [-lbaddtocluster](#) on page 118
- [-allowxcmd](#) on page 119
- [-authproviderdomain](#) on page 119
- [-encryptfips](#) on page 119
- [-conversationport](#) on page 120
- [-logconfigloc](#) on page 120
- [-generic](#) on page 120
- [-sspi](#) on page 120
- [-secpackage](#) on page 120
- [-secpackagelist](#) on page 120

Syntax Description

-dnsmatch *DNSAlias*

specifies a DNS alias that will be accepted by the object spawner as a match for the local machine name.

In addition, the spawner replaces the `dnsMatch` value with the local machine name in its list of servers. This option is necessary if your network configuration resolves a single DNS alias to multiple machines that run SAS object spawners. For example, you configure SAS servers and spawners on two different machines: `n1.my.org` and `n2.my.org`. The DNS alias `srv.my.org` resolves to both of these machines, so clients can send a request to the alias and one of the two spawners will receive it. To support this configuration, specify `-dnsMatch srv.my.org` in the spawner start-up command on each machine.

-dnsname | **-dns** *name*

specifies which IP stack is used for communication between the spawner and the servers that the spawner launches. This option can be abbreviated as `-dns`.

-hostknownby *DNSAlias*

specifies a DNS alias that will be accepted by the object spawner as a match for the local machine name.

-sasspawnercn | **-ssc** *name*

specifies the name (used in the SAS Management Console configuration) of the spawner object to use for this spawner invocation configuration. If you do not specify `-sasspawnercn`, the object spawner uses the first spawner definition (on the metadata server) with the same machine name as the current host.

Note: If none of the spawner definitions contain a host name of the current host, you must specify the `-sasspawnercn` option to designate which spawner definition to use. If you specify a spawner name that contains embedded blank spaces, then you must enclose the name in quotation marks (" "). This option can be abbreviated as `-ssc`.

-lbaddtocluster | **-lbadd** *logicalLoadbalancedServerName* *</serverName>*

specifies a logical, load-balanced server to which the object spawner should add itself. `-lbaddtocluster` enables you to add a new host to an existing load balancing

peer object without requiring a peer refresh. This feature is required for cloud computing and software as a service models.

serverName is optional and is the name of the server to add the load-balanced server information to. When you omit *serverName*, the Object Spawner updates the first server definition associated with the logical load-balanced server.

`-lbaddtocluster` must be used in conjunction with the `-sasSpawnercn` (`-ssc`) option.

For example:

```
C:\Program Files\SASHome\SASFoundation\9.3\objspawn.exe
-ssc "SASApp - Spawner"
-lbadd "SASApp - Logical Workspace Server" /compute.server.01.example.com
```

When using `-lbaddtocluster`, note the following:

- The Object Spawner definition must already be associated with at least one server definition in the logical server that you specify.
- The Object Spawner can add only itself to one cluster at a time. If you need to update multiple clusters, use the SAS Management Console.
- Any error encountered during an `-lbaddtocluster` spawned update of metadata causes the Object Spawner to shut down.
- Any error encountered after an `-lbaddtocluster` spawned update of metadata does not cause SAS to rollback or reset the metadata. The metadata changes made persist.

`-allowxcmd`

enables host commands and PIPE commands for all servers that are started by the spawner. You can use the SAS Management Console to enable host and PIPE commands on a server-by-server basis (**Server Properties** ⇒ **Options** ⇒ **Advanced Options** ⇒ **Launch Properties**). By default, the spawner starts all servers with the `-NOXCMD` SAS system option. When you specify `-allowxcmd`, the spawner no longer specifies `-NOXCMD` when launching server sessions.

CAUTION:

When you specify `-allowxcmd`, clients can use host commands to perform potentially harmful operations such as file deletion.

`-authproviderdomain | -authpd hostuser:domain`

associates a domain with the host authentication provider, because the spawner starts either a SAS Workspace Server or SAS Stored Process Server. Workspace and stored process servers authenticate only against the host. For example: `-authp hostuser:mydomain`

`-encryptfips`

causes the spawner to run in Federal Information Processing Standards (FIPS) compliance mode that is provided by SAS/SECURE software in its implementation of the FIPS 140–2 specification. The object spawner checks each server and the object spawner's operator port to ensure that AES is used as the encryption algorithm. If the object spawner's operator connection is not using AES, the object spawner terminates. SAS marks any servers that are not using AES as invalid. If the spawner finds no valid servers, it terminates.

If the object spawner's operator port is not compliant, the following error message is written to the log:

```
The object spawner is running in FIPS compliance mode,
therefore the only encryption algorithm that is supported for the
Operator connection for <Spawner Name=""> (<Spawner ID="">) is AES.
```

If a server is not compliant, the warning message output is:

```
The object spawner is running in FIPS compliance mode,
therefore the only encryption algorithm that is supported for %s (%s)
is AES. This server definition will not be included.
```

For more information, see “FIPS 140-2 Standards Compliance” in Chapter 1 of *Encryption in SAS*.

-conversationport | *-cp port*

specifies which port is used for communication between the spawner and the servers that the spawner launches. This option can be abbreviated as **-cp**.

-logconfigloc *filename*

sets the log options for the spawner in a log configuration file. For example: -
**logconfigloc "C:\SAS\Config\Levl\ObjectSpawner
 \logconfig.xml"** The file is an XML file that uses specific options. For more information, see “Using the SAS Logging Facility in the SAS Intelligence Platform” in Chapter 1 of *SAS Logging: Configuration and Programming Reference*.

Unless a path is specified, the spawner looks for the log config file in the current directory.

-generic

causes the spawner to use generic placeholders in data that it writes to its log. For example, user names are output as **<user identity>**, host names as **<IP address>**, connection and process IDs as **-1** or **<child process>**, configuration paths as **<configuration file path>**, and so on.

-sspi

identifies support for the Security Support Provider Interface for single sign-on connections to the object spawner.

For more information, see “SSPI System Option” on page 151.

-secpack *"package-name"* | **"negotiate"**

specifies the security package that the spawner should use to authenticate incoming client connections. (Use with **-sspi**.)

The *"package-name"* value specifies the security package that the spawner should use to authenticate incoming client connections. Enclose the security package name within double quotation marks (").

The default value is **"negotiate"**. This value enables the spawner to present a set of valid security packages (through **SECPACKAGELIST**), which the server uses to find a match with an incoming client connection. If the client specifies a security package in the list, then the server attempts to authenticate the client the matched security package. Enclose **negotiate** within double quotation marks (").

For more information, see “SECPACKAGE System Option” on page 148.

-secpackelist *"package-name-1,[package-name-2],[...]"*

identifies the security package that is used by the server to authenticate incoming client connections. (Use with **-sspi**.) The package-name value defaults to **"Kerberos,NTLM"**.

Enclose the security package name within double quotation marks ("). Delimit an additional package name with a comma (,).

For more information, see “SECPACKAGELIST System Option” on page 150.

Example

This example illustrates a typical spawner invocation:

```
C:\Program Files\SASHome\SASFoundation\9.3\objspawn.exe
-dnsMatch dl18374.na.sas.com -ssc "SAS [Config-Level] Object Spawner"
-xmlconfigfile xmlconfig.xml -sspi -logconfigloc logconfig.xml
```

Metadata Connection and Security Options

Overview of Metadata Connection and Security Options

Use the following metadata connection options for the object spawner to connect to the metadata server in order to find workspace server and stored process server definitions:

- -xmlconfigfile
- -metaserver
- -metaport
- -metauser
- -metapass
- -metaprofile
- -metaconnect
- -metarepository
- -metaspn
- -metaencryptalg
- -metaencryptlevel

For metadata server connection options, you can either specify individual metadata options or point to a file that contains connection options by using -xmlconfigfile. For more information about the metadata configuration options, see Chapter 6, “System Options for Metadata,” in *SAS Language Interfaces to Metadata*.

Syntax Description

-xmlconfigfile | -xcf *filename*

specifies a fully qualified path to a metadata configuration file that contains a SAS Metadata Server definition to connect to for the complete configuration. On Windows, enclose paths with embedded blank spaces in double quotation marks. On z/OS, specify filenames similar to UNIX file paths due to the requirement for z/OS UNIX System Services.

This option can be abbreviated as -xcf.

-metaserver *host name*

identifies the metadata server where configuration information is saved and can be queried.

-metaport *port*

designates the metadata server port.

-metauser *user ID*

identifies metadata server user ID.

-metapass *userPassword*

identifies metadata server password.

-metaprofile *filename*

specifies the location of the file that contains metadata profiles.

- metaconnect *name*
identifies the named connection from the -metaprofile file to use as the default metadata server connection.
- metarepository *repositoryName*
identifies the name of the metadata repository to query.
- metaspn *servicePrincipalName*
identifies the metadata server service principal name to use when communicating with a metadata server.
- metaencryptalg none | rc2 | rc4 | des | aes | tripledes | sas
specifies the type of encryption to use when communicating with a metadata server. (**sas** is the short form for **sasproprietary**.)
- metaencryptlevel credentials | everything
specifies the encryption level to use when communicating with a metadata server.

Examples

EXAMPLE 1: In this example, the object spawner connects to a metadata server by using command line options:

```
C:\Program Files\SASHome\SASFoundation\9.3\objspawn.exe
-metaserver "metaserver.unx.alphacorp.com"
-metaport 8561 -metauser "sastrust@saspw" -metapass "sasuser1"
```

EXAMPLE 2: In this example, the object spawner points to a connection file (**metadataConfig.xml**) that contains metadata server connection information:

```
C:\Program Files\SASHome\SASFoundation\9.3\objspawn.exe -xcf metadataConfig.xml
```

Service Options

Overview of Service Options

Use the following service options to create, modify, and delete object spawner service definitions on Windows only:

- -install
- -name
- -servdir
- -servuser
- -servpass
- -installdependencies
- -deinstall

For more information, see [“Update a Windows Object Spawner Service” on page 111](#).

Syntax Description

```
-install | -i <-name name> <-servdir directory> <-servuser user ID> <-servpass
password>
```

instructs the spawner to install as a Windows service. This option can be abbreviated as -i. When asked to install as a service, the spawner records all options that are specified at installation time in the registry under the following key: **"SYSTEM\CurrentControlSet\Services\service-name\Parameters"** You can

also specify options in the start-up parameters when you manually start the spawner service from the Microsoft Windows Services snap-in (services.msc).

-name *name*

specifies a Windows service name to use when installing the spawner as a service. Use with **-install**. The default value is **SAS Object Spawner Daemon III**.

If you specify a service name that contains embedded blank spaces, then you must enclose the name in quotation marks (" ").

Note: If you install more than one spawner as a service on the same machine, then you must use the **-name** option to give each spawner service a unique name.

-servdir *directory*

specifies the directory in which to run the Windows service. Use with **-install**. By default, the directory is: *install-dir\Config\Lev1\ObjectSpawner*.

-servuser | -su *user ID*

specifies a user name that the Windows service will run under, when you also specify the **-install** option. Use with **-install**. This option can be abbreviated as **-su**.

-servpass | -sp *password*

specifies a password for the user name that is specified in the **-servUser** option. Use with **-install**. This option can be abbreviated as **-sp**.

-installdependencies | -idep *service-1*<,*service-2*>;...>

specifies the Windows services that must be started before the spawner service starts. The *service* value is the name of the dependent service that is displayed in the Microsoft Windows Services snap-in (services.msc).

This option can be abbreviated as **-idep**.

-deinstall | -di -name *name*

instructs the spawner to uninstall as a Windows service. This option can be abbreviated as **-di**.

Note: The *name* value is the spawner service name that is displayed in the Microsoft Windows Services snap-in (services.msc).

Examples

EXAMPLE 1: In this example, the spawner is installed as a Windows service with “SAS [Config-Lev1] Object Spawner” displaying in the Microsoft Windows Services snap-in (services.msc).

```
C:\Program Files\SASHome\SASFoundation\9.3\objspawn.exe
-i -name "SAS [Config-Lev1] Object Spawner"
```

The spawner is installed under the Windows system user and runs in the default directory (*install-dir\ObjectSpawner*).

EXAMPLE 2: In this example, the spawner service is dependent on the metadata server starting first:

```
C:\Program Files\SASHome\SASFoundation\9.3\objspawn.exe
-idep "SAS [Config-Lev1] Metadata Server"
```

EXAMPLE 3: In this example, the spawner service is uninstalled:

```
C:\Program Files\SASHome\SASFoundation\9.3\objspawn.exe
-di -name "SAS [Config-Lev1] Metadata Server"
```


Chapter 13

Administering SAS OLAP Servers

Administrative Overview for SAS OLAP Servers	126
Migrating OLAP Cubes	127
Installing and Configuring SAS OLAP Servers	127
Considerations for SAS OLAP Servers	127
Initial Deployment	127
Add a SAS OLAP Server	128
Add a Load-Balancing Cluster of SAS OLAP Servers	128
Connecting to SAS OLAP Servers	128
Starting SAS OLAP Servers	128
Stopping, Pausing, and Resuming SAS OLAP Servers	129
Disabling and Enabling Cubes	129
Building Cubes: Overview for Administrators	129
Updating Cubes: Overview for Administrators	130
Overview of Updating Cubes: Overview for Administrators	130
Update a Cube in Place	130
Update a Cube Incrementally	131
Coalescing Cubes	131
Deleting Cubes	132
Authorizing Access to SAS OLAP Servers	132
Authorizing Access to OLAP Cubes and Cube Data	133
Overview of Authorizing Access to OLAP Cubes and Cube Data	133
Change Permissions and Disable Cubes Using SAS OLAP Cube Studio	133
Monitoring SAS OLAP Servers	133
Managing OLAP Sessions and Queries	134
Logging SAS OLAP Servers	134
Tuning SAS OLAP Servers with Advanced Server Options	134
Overview of Tuning SAS OLAP Servers with Advanced Server Options	134
Tune the Cube Cache	135
Tune the Subquery Caches	135
Tune the Query Thread Pool	136
Set Values for Flattened Row in the Server Tab of the Advanced Options Window	136
Set Values in the Performance Tab of the Advanced Options Window	137

Refreshing Cube Metadata for Calculated Members and Named Sets	138
Administering OLAP Schemas	138

Administrative Overview for SAS OLAP Servers

The SAS OLAP implementation consists of the following components:

SAS OLAP Server

references cube data to generate result data sets that are delivered to OLAP clients in response to queries from cube viewers. One server supports one or more concurrent client connections. SAS OLAP Servers can be grouped into load-balancing clusters to provide scalable support for one or more cubes.

SAS OLAP Cube Studio

generates cube metadata and creates SAS code that is executed to build cubes. . Other tools define aggregations, generate calculated members, export and import cubes between metadata repositories, and generate new OLAP schemas.

SAS Language Elements

PROC OLAP creates and updates cubes. PROC OLAPOPERATE provides programmatic administration of SAS OLAP Servers, as described in the *SAS OLAP Server: User's Guide*

OLAP Schema

groups cubes that are exclusively accessed by one or more SAS OLAP Servers. Each cube is listed in one and only one OLAP schema. Each SAS OLAP Server is required to use one OLAP schema. Multiple servers can use the same schema.

SAS Workspace Server

executes SAS code to build or update cubes. For information about the administration of SAS Workspace Servers, see [“Managing Workspace Servers and Stored Process Servers” on page 91](#).

SAS Metadata Server

stores metadata that defines cubes, cube data, cube data access permissions, and load-balancing data.

Cube Viewer

generates queries in the MDX language and displays result data sets as they are received from SAS OLAP Servers. The primary cube viewers are SAS Web OLAP Viewer, SAS Web Report Studio, and SAS Enterprise Guide. Because queries are submitted in the OLE DB for OLAP application programming interface, you can also view cubes with third-party viewers such as Microsoft Excel.

For more information about the MDX query language, see the *SAS OLAP Server: MDX Guide*.

SAS Management Console

displays SAS OLAP Servers and OLAP schemas and contains the OLAP server administrative plug-ins Server Manager and SAS OLAP Server Monitor.

Server Manager

displays OLAP servers and logical OLAP servers, provides server controls, and manages advanced server options. This application is provided in the Environment Management folder in the Plug-ins tree in SAS Management Console.

SAS OLAP Server Monitor

displays all OLAP servers and schemas, provides session controls, and manages advanced server options. This application is provided in the Monitoring folder in the Plug-ins tree in SAS Management Console.

Administrators support SAS OLAP Servers by configuring hosts, installing and configuring SAS OLAP Servers and SAS Workspace Servers, starting, stopping, and restarting servers, monitoring servers, managing server logs, tuning servers, and configuring libraries and permissions for users. These tasks are accomplished using the SAS Deployment Wizard and SAS Management Console.

User tasks for SAS OLAP Server include building cubes, viewing cubes, and tuning cubes. These tasks are accomplished with SAS OLAP Cube Studio. User tasks are documented in the SAS OLAP Server User's Guide and in the Help for SAS OLAP Cube Studio.

Migrating OLAP Cubes

OLAP cubes that were built in SAS 9.2 can be used in SAS 9.3 without changes or migration. Before rebuilding your cubes in SAS 9.3, review your cubes to ensure that all paths are correct.

Note: Cubes that you build in SAS 9.3 are not supported on SAS 9.2 OLAP Servers.

For cubes that you built in SAS 9.1.3 or earlier, follow the migration steps in “Rebuild SAS 9.1.3 OLAP Cubes” in Chapter 5 of *SAS Intelligence Platform: Migration Guide*. The migration process has not changed since the SAS 9.2 release.

Installing and Configuring SAS OLAP Servers

Considerations for SAS OLAP Servers

Consider the following with regard to the installation and configuration of SAS OLAP Servers:

- Cubes are built and updated on SAS Workspace Servers. Cubes are queried on SAS OLAP Servers.
- Install one SAS OLAP Server per host.
- Workspace servers and OLAP servers need to access the same physical locations.
- Workspace servers and OLAP servers do not need to be installed on the same host.
- Workspace servers require sufficient storage capacity for temporary files that are generated during cube builds, as described in “[Configure Storage for Temporary OLAP Cube Build Files on SAS Workspace Servers](#)” on page 102.

Initial Deployment

You initially install SAS OLAP Servers with the rest of the SAS Intelligence Platform using your SAS software depot, your deployment plan, and the SAS Deployment Wizard, as described in the *SAS Intelligence Platform: Installation and Configuration Guide*.

After your initial installation, you:

- Configure source data libraries, as described in *SAS Intelligence Platform: Data Administration Guide*
- Configure access control, as described in *SAS Intelligence Platform: Security Administration Guide*. You have the capability of limiting access within cubes by preventing the display of specified dimensions, hierarchies, levels, and members.
- Configure cube viewers, as described in *SAS Intelligence Platform: Desktop Application Administration Guide*

Add a SAS OLAP Server

After your initial deployment, demand for additional cubes might require you to add another SAS OLAP Server. Contact a SAS support representative to receive assistance with your hardware configuration plan.

To install a new SAS OLAP Server, you install a new SAS Application Server on a new host using the SAS Deployment Wizard. Use the wizard in Configuration mode to install metadata, software, and configuration files in a single pass. After installation, you either add cubes to the new OLAP schema file or you assign the new server to the cubes that are listed in an existing OLAP schema. To work with OLAP schemas in SAS Management Console, select the **Folders** tab, open the **Shared Data** folder, right-click the **OLAP schema** file, and select **Properties**.

Add a Load-Balancing Cluster of SAS OLAP Servers

Load-balancing clusters of OLAP servers improve query response times by dividing concurrent client sessions across multiple hosts. You can add or remove servers from the cluster in response to changes in the average number of concurrent sessions. To add a load-balancing OLAP cluster, see “[Understanding Server Load Balancing](#)” on page 34.

Connecting to SAS OLAP Servers

In SAS Management Console, you can connect to OLAP servers and validate connections. Connecting to a server indicates that the server is responsive. Validating a server confirms that the connection is valid and the server has made contact with its OLAP schema and with the cubes that are listed in that schema.

To connect to a SAS OLAP Server or to validate a connection, right-click the server in the Server Manager, or right-click the logical OLAP server in the SAS OLAP Server Monitor, and select **Connect** or **Validate**.

Starting SAS OLAP Servers

SAS OLAP Servers are background processes that remain active until you need to stop or pause them. You stop or pause and resume SAS OLAP Servers to change OLAP schemas or change server properties. To maximize availability, OLAP servers are generally configured to start when their hosts are initialized. After you change server properties, you restart the server.

Stopping, Pausing, and Resuming SAS OLAP Servers

In SAS Management Console, you can stop or quiesce OLAP servers to rebuild cubes, change OLAP schemas, or change server properties. When you stop an OLAP server, any active queries are immediately terminated. When you quiesce an OLAP server, the server accepts no new queries and terminates after the completion of the last active query.

Note: Stopping or pausing a SAS OLAP Server does not necessarily prevent access to a particular cube, because other servers might share the same OLAP schema. To update a cube without taking it off-line, you should disable the cube rather than stopping or pausing the server.

To stop, quiesce, pause, or resume a SAS OLAP server, right-click the server instance in the Server Manager and select the server from the pop-up menu, or right-click the logical OLAP server in the SAS OLAP Server Monitor.

Disabling and Enabling Cubes

Disable cubes when you want to prevent all access to the cube, without stopping any OLAP servers. You disable cubes as part of the cube update process, before clients can access a new version of a cube. When a cube is disabled, no new sessions can be started on that cube, on any OLAP server. Disabling a cube does not close existing sessions. Existing sessions either terminate on their own, or are closed using the SAS OLAP Server Monitor.

Note that you do not need to disable a cube to update a cube in-place. When you update a cube in-place, you add data without closing sessions. Although this method of update does not interrupt client access, the process does not enable testing beforehand, and it does not provide a previous version of the cube that you can put back into service.

To disable or enable a cube, follow these steps:

1. Start SAS Management Console.
2. Expand the Monitoring folder and expand the SAS OLAP Server Monitor.
3. Expand the OLAP Schema folder.
4. Right-click the cube and select **Disable** or **Enable**.

Building Cubes: Overview for Administrators

Users build cubes with SAS OLAP Cube Studio or PROC OLAP. This section describes the cube build process from an administrative perspective. For more information about cube builds, refer to the *SAS OLAP Server: User's Guide*.

When you build a cube that you specify the cube data and cube structure, and the SAS OLAP Server software creates the cube. The cube is assigned to an OLAP schema, and one or more SAS OLAP Servers that use that schema support client queries on the cube.

Due to the pre-summarization of data, and based on the amount and structure of the data, cube builds can be time-consuming and resource-intensive. For this reason, it is advantageous to build cubes using a non-production OLAP schema. When rebuilding existing cubes, follow these steps to minimize the time that the cube is out of service:

1. Change the OLAP schema of the existing cube from the production schema to a non-production schema.
2. Rebuild the cube using the non-production schema.
3. After the rebuild, change the OLAP schema of the new cube to the production schema.

Updating Cubes: Overview for Administrators

Overview of Updating Cubes: Overview for Administrators

Users update cubes with SAS OLAP Cube Studio or PROC OLAP. This section describes the update process from an administrative perspective. For more information about cube updates, refer to the *SAS OLAP Server: User's Guide*.

Starting in SAS 9.2, users can perform an in-place cube update to add data to cubes without rebuilding the entire cube, and without restarting the SAS OLAP Server. Users can also perform an incremental cube update.

Cube updates add new aggregation tables that combine with existing aggregation tables in previous versions of the cube. The updated version of the cube looks through to the previous versions to obtain all cube data.

As you accumulate cube updates and aggregation tables, it is possible that OLAP server performance might decline. To improve performance, users can coalesce the cube. The coalesce process combines all of the separate aggregation tables from previous updates into a single, more efficient set of aggregations.

Update a Cube in Place

When you update a cube in place, you add data and new members directly to an active cube, without disabling, stopping, or rebuilding the cube.

During and after the completion of the in-place update, existing sessions will continue to run on the old cube. When the last session ends on the old cube, the old cube is closed. After the old cube is closed, new sessions apply to the new cube. Although existing sessions are connected to the old cube, new sessions are also applied to the old cube. If you do not want to wait for all open sessions to close, you can enforce a switch to the new cube. To switch to the new cube, disable the cube, close all open sessions that query that cube, and enable the cube.

The in-place cube update runs like an incremental cube update. At the end of the in-place update, the cube metadata is updated to point to the new generation folder and the old generation folder is deleted. If the old generation cannot be deleted, because an existing session has locked the files, then the next cube update will attempt to delete the old generation folder.

Update a Cube Incrementally

When you run an incremental cube update, you use PROC OLAP to add data and new members to a new version of the cube. After the update, the new cube is immediately available for queries under its new cube name. The previous version of the cube remains active until you disable the old cube and enable the new version.

Incremental updates are beneficial because they:

- enable testing outside of the production environment
- can be archived to maintain a history changes
- provide for rollback or review

Each version of the cube remains completely viable. You can enable any version and receive a fully functional cube that contains all of the data that was previously available in that version.

Each version of the cube is represented by a generation folder. The folder contains an aggregation table. The aggregation table contains the new members and new data that was added to the cube in that version.

The totality of data in the new version of the cube is represented by all of the generation folders and the original aggregation table from which the generations were built.

When you generate a new version of a cube, the new version is immediately available for queries under its new name. To replace the old version with the new version, using the same cube name, follow these steps:

1. Disable the cube with PROC OLAPOPERATE, SAS OLAP Cube Studio, or the SAS OLAP Server Monitor.
2. Close all sessions on the old version of the cube.
3. Rename the old version of the cube.
4. Rename the new version of the cube to name of the old cube.
5. Enable the cube.

As versions accumulate, you can coalesce cubes to combine all versions into a single, more efficient aggregation table, as described in the next section.

After you create a new version of a cube, you can assign the old cube, under its new name, to a different OLAP schema for archival purposes.

You can delete old versions as you would with any other cube. Deleting a version removes its metadata. Physical data that is used by later versions is not deleted. To delete all generation folders back to the last coalesced cube, coalesce the new version of the cube.

Coalescing Cubes

Coalescing a cube combines all of the generation folders from previous versions of the cube into a single generation folder in the latest version of the cube. The old generation folders are deleted. The generation folders that are combined and deleted are those that were produced after the last time you coalesced the cube.

To coalesce a cube, follow these steps:

1. Disable the cube with SAS OLAP Cube Studio or with PROC OLAPOPERATE, or follow these steps:
 - a. Open SAS Management Console.
 - b. Expand the SAS OLAP Server Monitor.
 - c. Expand the OLAP schema that contains the cube.
 - d. Right-click the cube and select **Disable**.

For more information about OLAPOPERATE and SAS OLAP Cube Studio, see the *SAS OLAP Server: User's Guide*.

2. Close all open sessions on the cube by running PROC OLAPOPERATE programs for each SAS OLAP Server that supports queries on that cube, or by displaying and closing sessions in SAS OLAP Server Monitor.
3. Coalesce the cube using the COALESCE option in PROC OLAP.
4. When the coalesce operation is complete, enable the cube.

Deleting Cubes

Cube deletion takes place in PROC OLAP and SAS OLAP Cube Studio. If you delete cubes, that have not been updated, then the deletion operation removes cube metadata and physical files. For cubes that have undergone either in place or incremental update, the deletion of old versions and generation folders occurs programmatically. Manual file deletion is not required.

For cubes that have undergone in-place updates, the next in-place update will remove all old generation folders.

For cubes that have received incremental updates, deletion of metadata and physical files takes place as part of the coalesce process.

Authorizing Access to SAS OLAP Servers

To operate SAS OLAP Servers, you need Administer permission. Unrestricted access is not required, and no specific role or capability is required.

For access to all aspects of SAS Management Console other than server operation, OLAP administrators need to be members of the role Management Console: Advanced. This role is required to display the SAS OLAP Server Monitor under the **Monitoring** node in the **Plug-ins** tab.

Users of SAS OLAP Cube Studio need readMetadata and writeMetadata permissions.

Users of cube viewers need Read and readMetadata permissions, but writeMetadata might not be necessary.

Default roles are available for association with users of SAS cube viewers. If you, as the OLAP administrator, are a member of the role SAS Metadata Server: User Administration, you can, as needed, add cube viewer users to the roles Add-In for Microsoft Office: OLAP and Enterprise Guide: OLAP.

To read and update the authorizations for SAS OLAP Servers, follow these steps:

1. Open SAS Management Console.
2. Expand the Server Manager.
3. Right-click the application server, logical server, or server instance and select Properties.
4. In the Properties window, select the **Authorization** tab.

For more information about security concepts and tasks, see the *SAS Intelligence Platform: Security Administration Guide*. Of particular interest is the material that deals with the protection of server definitions in metadata.

Authorizing Access to OLAP Cubes and Cube Data

Overview of Authorizing Access to OLAP Cubes and Cube Data

In SAS Management Console, the **Authorization** tabs of cubes and OLAP schemas display the permissions that apply to those objects. With appropriate permission, you (the OLAP administrator) can grant or deny Read or Write permission to users, roles, and groups.

You can also control access to cube data, so that a single cube provides different data for different users. You can prevent access to specified dimensions, hierarchies, and levels, and you can define permission conditions to filter cube data within dimensions. To set permission conditions on a cube, use the MDX Expression Builder in SAS Management console, or open the cube's **Authorization** tab in SAS OLAP Cube Studio. For details and procedures, see the “OLAP Member-Level Permissions” chapter in the *SAS Intelligence Platform: Security Administration Guide*.

Change Permissions and Disable Cubes Using SAS OLAP Cube Studio

In SAS OLAP Cube Studio, you need Administer permission on both the cube and the client's SAS OLAP Server in order to change cube permissions and enable and disable cubes.

In SAS OLAP Cube Studio, to find the SAS OLAP Server to which the client is connected, select **Tools** ⇒ **Options** and refer to the **Server** field.

To view and change user permissions on the SAS OLAP Server, display the Properties window in SAS Management Console and select the **Authorization** tab.

Monitoring SAS OLAP Servers

In SAS Management Console, information about the status of SAS OLAP Servers is provided in the Server Manager and in the SAS OLAP Server Monitor. In either case, you select the server instance (not the logical server) to display status information in the right pane.

When you select an OLAP server in the Server Manager, it displays information about connections, clients, sessions, options and values, available loggers, and the text of the currently enabled log.

The SAS OLAP Server Monitor initially displays session information. You can select sessions to display query information. You can also manage sessions and queries.

Managing OLAP Sessions and Queries

Use the SAS OLAP Server Monitor in SAS Management Console to display information or close active client (cube viewer) sessions and the queries that are being processed in those sessions.

To display sessions, connect to the OLAP server. Current sessions appear in the right pane. For definitions of the displayed fields, select **Help** ⇒ **Help on SAS OLAP Server Monitor**. In the Help contents, select **About Sessions**.

To display queries, first display a session, and then double-click the session to see lists of data queries and metadata queries. Data queries return to the client a result set that consists of requested cube data. Metadata queries return to the client-requested metadata, such as a cube description, rather than cube data.

To close sessions or queries, right-click and select **Close**.

Logging SAS OLAP Servers

By default, SAS OLAP Servers create a new rolling log file on a daily basis. The rolling log is also displayed in the **Log** tab of the OLAP Server, in the Server Manager of SAS Management Console.

An ARM log file is created at server start or restart. The ARM log is normally disabled. Enable the ARM log only when testing or tuning cubes, when both the cube and the OLAP server are outside of the production environment, as described in “Enabling ARM Logs in SAS OLAP Servers,” in the *SAS Intelligence Platform: System Administration Guide*.

Tuning SAS OLAP Servers with Advanced Server Options

Overview of Tuning SAS OLAP Servers with Advanced Server Options

To display the advanced server options for a SAS OLAP Server, follow these steps:

1. Within SAS Management Console, expand the nodes under the **Server Manager** until you can see your SAS OLAP Server. The server components represent the actual machines that the SAS OLAP Servers are running on.
2. Right-click the server and select **Properties**.
3. In the Properties window, select the **Options** tab.

4. On the **Options** tab, select **Advanced Options**.

In the Advanced Options window, tune your servers on the **Performance**, **Cache**, and **Query Thread Pool** tabs. Use the other tabs (**Debug Query**, **Debug Server**, and **Journals**) only under the direction of SAS technical support, outside of the production environment, to generate detailed log information. The logging process severely limits server performance.

Tune the Cube Cache

The cube cache stores an in-memory copy of the cube's metadata. The metadata describes the calculated members and named sets that are used to parse the MDX query syntax. The metadata does not include any disk-resident aggregations.

As the server processes a query, the server first checks the cube cache to determine whether the cube metadata is in memory. If the cube metadata is in memory, the server uses the cached metadata. If the cube metadata is not in memory, the server loads the metadata from the SAS Metadata Server.

Cubes are added to the cache as they are queried. Cubes remain in the cache until the OLAP server stops, or until the cache reaches its maximum number of cubes. After the limit is reached, new queries on new cubes cause the new cube's metadata to replace the metadata of the oldest cube in cache (first in, first out).

The cube cache is implemented as least recently used. As the cache becomes full, cubes are removed based on usage.

The default limit on the number of cubes in the cache is 20. To change this value, use the field **Maximum number of cubes in cache** in the **Cache** tab of the Advanced Options window. Restart your OLAP server to activate the new value.

Tune the Subquery Caches

The **Cache** tab in the Advanced Options window enables you to set the maximum size of the subquery caches. You can also enable and disable the caching of empty intermediate result sets. By default, the maximum size of each subquery cache is 5 megabytes, and the caching of empty intermediate result sets is disabled.

SAS OLAP Servers maintain a unique subquery cache for each query for the duration of the processing of that query. When a server is processing concurrent queries, server memory contains multiple subquery caches.

During query processing, the subquery caches grow in size. If the subquery caches reach a specified maximum size, the contents of the caches are paged (swapped) into temporary disk storage.

Memory is not pre-allocated for subquery caches. Paging might therefore occur before the subquery caches reach their maximum size.

Paging for the subquery caches is indicated by disk writes by the SAS OLAP Server. This is the only case where the server writes to disk, except for the logging and debugging operations that take place outside of the production environment. If you see disk input and output from the server, you can do one of the following:

- change the subquery cache settings
- change the value of the MEMSIZE= system option for the server
- add physical memory to the host.

Smaller maximum sizes of the subquery caches might limit performance improvements due to memory-intensive paging to and from temporary disk storage, particularly if the caches include empty intermediate result sets.

Larger maximum sizes of the subquery caches, when combined with a large number of concurrent queries, might occupy an inefficient percentage of available server memory.

In many cases, the default maximum cache size (5 megabytes) is a good choice. It is recommended that the maximum subquery cache size not exceed 50 megabytes.

To set the maximum size of the subquery caches, use the field **Memory size for subquery cache**.

To cache empty subquery result sets, select the check box to show a check mark in the field **Cache the empty subquery result sets**. Generally, most of the set processing in OLAP queries involves very sparse matrices. For this reason, empty subquery result sets are not cached by default. If you have plenty of available memory, or if your cube or your queries are relatively dense, then caching empty subquery result sets might improve performance.

Tune the Query Thread Pool

The query thread pool is used to efficiently assign threads to query requests that are received by a SAS OLAP Server. As threads are assigned to queries, additional threads are allocated, up to the specified maximum number of threads. If the number of query requests and active queries exceeds the maximum number of threads, requests are queued and assigned to threads as threads are reclaimed.

You can set the following parameters of the query thread pool on the **Query Thread Pool** tab of the Advanced Options window:

- minimum number of threads in the pool. The recommended value is 1.
- maximum number of threads in the pool. The recommended value is twice the number of available CPUs on the host or lower.
- threshold number of query requests that are required before the requests are assigned to threads.
- time-out for thread reclamation.
- thread stack size.

As with other server option changes, you need to restart your OLAP server to activate your new values.

Set Values for Flattened Row in the Server Tab of the Advanced Options Window

The **Server** tab of the Advanced Options window for SAS OLAP Servers contains two options for flattened results sets: **Maximum number of flattened row** and **Maximum memory size for flattened rowset**. Increase the size of these options if queries fail with a “flattened rowset size limit reached” error.

Most applications get their query results as multidimensional result sets. However, there are two exceptions, drillthrough queries and SQL passthru queries:

Drillthrough queries

Drillthrough queries are not really multidimensional queries; they are two-dimensional queries against a relational table. As such, they can be requested as flattened result sets only.

SQL passthru queries

When using SQL to query a cube, the server always returns the data as flattened result set. This is because SQL understands rows and columns only.

Set Values in the Performance Tab of the Advanced Options Window

The **Performance** tab has four options that enable you to configure access to MOLAP aggregations in multidimensional databases. Three other options set limits on queries.

Memory available for group by operations

sets a memory allocation for MOLAP group-by operations, which are used to pre-summarize data on-demand, during queries. If a cube is pre-summarizing during queries, increasing this memory allocation might improve performance.

If a group-by operation exceeds its memory allocation, the SAS OLAP Server pages memory to the disk path that is specified in the option **Path to temporary working files**.

Note that the group-by memory is not pre-allocated, so the actual memory allocation might be limited by existing allocations.

Number of threads to spawn

specifies the number of threads that can be created to access MOLAP aggregations during queries. The default value 0 is preferable on multiprocessor hosts. When the value is 0, the number of threads is programmatically set to a value from 1 to 8, based the number of processors (CPUs) on the host. Note that this value does not limit the number of threads that are used to process queries. Query threads are set on the **Query Thread Pool** tab.

Path to temporary working files

specifies the disk directory that received memory swaps during MOLAP group-by operations.

Maximum segment ratio

controls the subsetting of indexes when processing MOLAP WHERE expressions. The ratio compares the number of segments that apply to the expression to the number of segments that do not apply to the expression. When the ratio exceeds the specified maximum, a subset index is not created before the expression is applied to the segments.

Maximum number of tuples in a set

sets the maximum size of the result sets that are sent to clients in response to queries. The default value is 1 million tuples.

MDX query timeout

automatically closes stale queries. No default value is provided.

Optimize queries that use the NONEMPTY and CROSSJOIN function

compares performance with or without optimization. Deselect this option to compare performance without optimization.

Refreshing Cube Metadata for Calculated Members and Named Sets

You can synchronize the calculated members and named sets in metadata with the data that is stored in the cube cache of a SAS OLAP Server by selecting **Actions** ⇒ **Refresh Cubes** in SAS Management Console. Another method is to select cubes in the SAS OLAP Server Monitor, right-click, and select **Refresh Cubes**. You can also select **Refresh Cube** in SAS OLAP Cube Studio.

To refresh cubes you must be connected to the SAS OLAP Servers and have the Administer permission.

Refresh Cubes does not add or delete aggregations, change display names, or change the value of the SECURITY_SUBSET option (which enables and disables member-level permissions). To make these changes, you do not need to rebuild the cube, but you do need to close all active sessions. Use **Stop**, **Pause**, or **Quiesce** before you use SAS OLAP Cube Studio or PROC OLAP to make these changes.

For information about updating all of the data in a cube, without rebuilding the cube, see the *SAS OLAP Server: User's Guide*.

Note that selecting **Refresh** in the Server Manager or SAS OLAP Server Monitor refreshes server connections and the **Folders** tab in SAS Management Console. **Refresh** does not synchronize calculated members or named sets.

Administering OLAP Schemas

OLAP schemas logically connect OLAP cubes to SAS OLAP Servers. Each cube appears in one schema. Each SAS OLAP Server is assigned to an OLAP schema. More than one server can be assigned to each schema.

To display schemas in SAS Management Console, select the **Plug-ins** tab, expand Monitoring, and expand the SAS OLAP Server Monitor. The schemas are displayed beneath the OLAP servers that are assigned to the schemas. Expand the schema folder to display the cubes that comprise the schema.

Schemas are also available in the **Inventory** tab, in the Shared Data folder.

To create OLAP schemas or assign servers to schemas, use the **OLAP Schema** tab in the Properties window of the SAS Application Server, as described in “Create or Assign an OLAP Schema” in the SAS Intelligence Platform: Data Administration Guide.

You can use SAS OLAP Cube Studio to create, assign, and delete schemas. You can also move a cube from one schema to another, as described in the *SAS OLAP Server: User's Guide*.

Chapter 14

System Options for SAS Application Server Components

Overview of System Options for SAS Application Server Components	139
Dictionary	139
OBJECTSERVER System Option	139
OBJECTSERVERPARMS System Option	142
SECPACKAGE System Option	148
SECPACKAGELIST System Option	150
SSPI System Option	151

Overview of System Options for SAS Application Server Components

There are several SAS system options available for you to use with application server components. The topics that follow discuss each available option in detail:

- [“OBJECTSERVER System Option” on page 139](#)
 - [“OBJECTSERVERPARMS System Option” on page 142](#)
 - [“SECPACKAGE System Option” on page 148](#)
 - [“SECPACKAGELIST System Option” on page 150](#)
 - [“SSPI System Option” on page 151](#)
-

Dictionary

OBJECTSERVER System Option

Specifies whether SAS is to run as an Integrated Object Model (IOM) server.

Valid in: configuration file, SAS invocation

Category: Environment control: Initialization and operation

PROC OPTIONS EXECMODES
GROUP=

Default: NOOBJECTSERVER

See: OBJECTSERVERPARMS System Option

Syntax

-objectserver

Syntax Description

-objectserver

when specified, SAS runs as an IOM server.

Details

An IOM server is a noninteractive SAS session that is run with the OBJECTSERVER system option. The spawner sets OBJECTSERVER for the SAS IOM servers that it invokes. In order to make it easy to specify the command, the server can be started by using a simple command with an option to connect back to the metadata server to obtain additional IOM-specific options.

You can specify the server start-up command in several locations:

- system command line
- script
- metadata (server definition in SAS Management Console)

The general form of the server start-up command is:

```
SAS-exec -objectserver other-system-options -objectserverparms "object-server-parameters"
```

- *SAS-exec*

is the absolute path to the SAS executable. For example:

- Windows:


```
C:\Program Files\SASHome\SASFoundation\9.3\sas
```
- UNIX:


```
/usr/local/bin/SAS/SASFoundation/9.3/sas
```

- **-objectserver**

launches this SAS session as an IOM server.

Note: In configuration files on z/OS, do *not* precede system options with a hyphen.

Also, an equal sign must precede the first parameter that follows the OBJECTSERVERPARMS keyword. For example:

```
objectserver objectserverparms="cel=credentials protocol=bridge port=8561"
```

- *other-system-options*

are optional, additional system options. System options that are typically used for servers include options such as: LOGCONFIGLOC, NOTERMINAL, and NOLOGO. For complete information about system options, see *SAS System Options: Reference*.

- **-objectserverparms** "object-server-parameters"

are IOM-server-specific options that are passed to the server by the OBJECTSERVERPARMS system option. For more information, see [“OBJECTSERVERPARMS System Option” on page 142](#).

Note: For SAS Workspace Servers that run on UNIX, it is sometimes necessary to call the SAS start-up command by using a wrapper script. For more information, see [“Managing Workspace Servers and Stored Process Servers” on page 91](#).

The server start-up command is obtained as follows:

- When the server is started by a spawner, the start-up command is stored in SAS metadata.

In the SAS metadata, there is one metadata field for the SAS start-up command and system options, and another field for the object server parameters. The object spawner combines these two fields, along with connection information and some spawner internal object server parameters, to create the complete SAS command. The object spawner then passes this command to the operating environment.

- When the server is started by a script or as a service or daemon, the command that is passed to the operating environment is not determined by SAS metadata.

However, SAS Workspace Servers, and any OLAP server can connect back to the SAS Metadata Server in order to obtain additional object server parameters and connection information (such as protocol engine and port number). (Some object server parameters cannot be obtained from the metadata. For more information, see [“OBJECTSERVERPARMS System Option” on page 142](#).)

When object server parameters are specified in the metadata, if there are any object server parameters that are also specified in the command, then the object server parameters in the command take precedence over those that are stored in the metadata.

Regardless of how the server is started, workspace servers, stored process servers, and OLAP servers, by default, also connect back to the metadata server in order to obtain configuration information, such as pre-assigned libraries, that is associated with the SAS Application Server.

The following table summarizes the ways that the SAS command, system options, and object server parameters can be specified for each type of IOM server. (1 = Object server parameters that are stored in metadata supplement the command-line object server parameters if the SERVER parameter is used.)

Table 14.1 Various Ways to Launch IOM Servers

Server	Spawner launched?	Use of SERVER parameter	Obtainable from metadata?		
			Command	Object server parameters	Librefs
Workspace server	Required	Spawner supplied	Yes (Spawner retrieves)	Yes (Spawner retrieves)	Yes
Stored process server	Required (Load-balanced)	Spawner supplied	Yes (Spawner retrieves)	Yes (Spawner retrieves)	Yes

Server	Spawner launched?	Use of SERVER parameter	Obtainable from metadata?		
			Command	Object server parameters	Librefs
OLAP server	Not allowed	Required	No	Yes	Yes (If SERVER is used)
Metadata server	Not allowed	Not allowed	No	No	No

Note: When you start the server with a script, some object server parameters, such as DNSMATCH, cannot be obtained from the metadata. Do not enter these object server parameters in your metadata. For details, see [“OBJECTSERVERPARMS System Option” on page 142](#).

Example

This example shows how OBJECTSERVER can be used in a configuration file used to start the metadata server:

```
-objectserver
-objectserverparms "cel=credentials protocol=bridge port=8561 classfactory=
0217E202-B560-11DB-AD91-001083FF6836 trustsaspeer=C:\SAS\
FoundationServers\Lev1\SASMeta\MetadataServer\trustedPeers.xml"
```

Note: In configuration files on z/OS, do *not* precede system options with a hyphen.

Also, an equal sign must precede the first parameter that follows the OBJECTSERVERPARMS keyword. For example:

```
objectserver objectserverparms="cel=credentials protocol=bridge port=8561"
```

See Also

System Options:

- [“OBJECTSERVERPARMS System Option” on page 142](#)

OBJECTSERVERPARMS System Option

Specifies start-up parameters for SAS Integrated Object Model (IOM) servers.

Valid in: configuration file, SAS invocation, metadata

Category: Environment control: Initialization and operation

PROC OPTIONS GROUP= EXECMODES (internal)

Applies to: workspace, stored process, OLAP, metadata

See: OBJECTSERVER System Option

Syntax

-objectserverparms "*parameter-1, ... , parameter-n*"

Syntax Description

parameter

The following table summarizes the valid parameters and where they can be used. (1=Do not store this parameter in metadata for workspace servers and stored process servers. The object spawner automatically adds this parameter when it starts these servers. 2=Use TIMEOUTSECONDS with OLAP servers, or stored process servers only. TIMEOUTSECONDS does not apply to workspace servers or to metadata servers.):

Table 14.2 OBJECTSERVERPARMS and Their Usage

Parameter	Valid In
CLASSFACTORY	Metadata, Command Line
CLIENTENCRYPTIONLEVEL	Command Line (only)
DNSMATCH	Command Line (only)
NOMETAAUTOINIT	Metadata, Command Line
PORT	Metadata, Command Line
PROTOCOL	Metadata, Command Line
SERVER	Command Line (only)
SERVICE	Metadata, Command Line
TIMEOUTSECONDS	Metadata, Command Line
TRUSTSASPEER	Metadata, Command Line

Parameter Description

You can use the following parameters with OBJECTSERVERPARMS:

"classfactory | clsid = *class-identifier*"

If you want to specify an alternate class to expose as the top-level class, use CLASSFACTORY to identify the class to IOM.

When using the **SERVER** object server parameter, the classfactory does not need to be specified because it is obtained from the logical server definition in the SAS Metadata Repository. This option is primarily used to start the SAS Metadata Server.

class-identifier

specifies the 36-character class ID number that specifies the type of server to instantiate. (For example, 0217E202-B560-11DB-AD91-001083FF6836)

specifies a SAS Metadata Server). An IOM server exposes one top-level class through its class identifier.

Valid in: Metadata, Command Line

Alias: clsid=

Default: workspace class

Restriction: Do not specify this option with spawned servers; it will be supplied automatically by the spawner.

"clientencryptionlevel | cel = none | credentials | everything"

The CLIENTENCRYPTIONLEVEL parameter specifies the degree of encryption for the IOM server to use when making outbound calls.

NONE

use no encryption.

CREDENTIALS

(default) encrypt only a user's credentials when establishing a client session.

EVERYTHING

encrypt all data (including a user's credentials) sent during a client session.

Valid in: Command Line (only)

Alias: cel=

Default: CREDENTIALS

Restriction: Do not specify this option with spawned servers; it will be supplied automatically by the spawner.

Interaction: This option is used only by the bridge protocol engine.

See: ["protocol = bridge | com | \(com, bridge\)" on page 145](#)

"dnsmatch | dns"

The object spawner replaces all instances of the DNSMATCH value with the local machine name in its list of servers. This option is necessary if your network configuration resolves a single DNS alias to multiple machines that run SAS servers.

For example, you configure SAS OLAP servers on two different machines: n1.my.org and n2.my.org. The DNS alias srv.my.org resolves to both of these machines, so clients can send a request to the alias and a server on one of the two machines will receive it. To support this configuration, specify DNSMATCH=srv.my.org in the server start-up command on each machine.

Valid in: Command Line (only)

Alias: dns

Default:

Restriction: Do not specify this option with spawned servers; it will be supplied automatically by the spawner, when you specify the -dnsMatch object spawner option.

"nometaautoinit | metaautoinit"

The NOMETAAUTOINIT parameter specifies that the IOM server should not connect back to the SAS Metadata Server during start-up in order to obtain additional configuration information such as object server parameters and pre-assigned libraries. In SAS 9.2 and later, IOM servers connect to the metadata server by default (METAAUTOINIT is on).

Valid in: Metadata, Command Line

Alias: (no alias)

Default: METAAUTOINIT

Restriction: The NOMETAAUTOINIT parameter is applicable only if you have specified your logical server with the SERVER object server parameter.

See: “Add System Options to the Workspace Server Launch Command” on page 101

"port = *port-number*"

The PORT parameter specifies the TCP/IP port on which the IOM bridge protocol engine listens for client connections.

port-number

is a valid TCP/IP port number.

Valid in: Metadata, Command Line

Alias: (no alias)

Default:

Restriction: Do not specify this option with spawned servers; it will be supplied automatically by the spawner.

"protocol = bridge | com | (com, bridge)"

The PROTOCOL parameter specifies the protocol engines to launch in server mode. Server mode indicates that the protocol engines will listen for client connections.

BRIDGE

is a protocol engine that you can launch in server mode.

COM

is a protocol engine that you can launch in server mode.

Valid in: Metadata, Command Line

Alias: (no alias)

Default:

Restriction: Do not specify this option with spawned servers; it will be supplied automatically by the spawner.

Tip: If you specify (com, bridge) then a multi-user server can simultaneously support clients that use different protocols. COM is not supported on all servers.

"server = '*logical-server-name*' | 'omsobj:LogicalServer/*object-ID*'"

The SERVER parameter can be used to retrieve many of the OBJECTSERVERPARMS options (including PORT, PROTOCOL, and CLASSFACTORY) from a SAS Metadata Repository.

'logical-server-name'

the logical server name for the IOM run-time and server application to use to locate configuration information in a SAS Metadata Repository. Enclose in single quotation marks, as shown in this example: **SERVER= 'sasapp - Logical OLAP'**.

*'omsobj:LogicalServer/*object-ID*'*

the object definition ID that is generated for the logical server for the IOM run-time and server application to use to locate configuration information in a SAS Metadata Repository. Enclose in single quotation marks.

To determine the generated object ID, in SAS Management Console, select the logical server definition, and then select **File** ⇨ ⇨ **Properties** from the menu bar. Use the Uniform Resource Identifier (URI) formatted value that is shown in the ID field as the argument for the object ID, as shown in this example:
SERVER= 'omsobj:LogicalServer/01234567.01234567'

Valid in: Command Line (only)

Alias: (no alias)

Default:

Restriction: Do not specify this option with spawned servers; it will be supplied automatically by the spawner.

"service = *service-name*"

The SERVICE parameter specifies the TCP service name for the port that the IOM Bridge protocol engine will use to listen for connections from clients.

service-name

a valid TCP service name (for example, from /etc/services on a UNIX system).

Valid in: Metadata, Command Line

Alias: (no alias)

Default:

Restriction: Do not specify this option with spawned servers; it will be supplied automatically by the spawner.

Interaction: Use with the IOM Bridge protocol engine.

See: "protocol = bridge | com | (com, bridge)" on page 145

"timeoutseconds = *seconds*"

The TIMEOUTSECONDS parameter specifies the time interval that an OLAP server or a stored process server waits before it stops a client process and cleans up the server run-time environment context.

seconds

the interval in seconds that an OLAP server or a stored process server waits before stopping a client process. *seconds* is a number that is equal to or greater than five.

Valid in: Metadata, Command Line

Alias: (no alias)

Default:

Restriction: Does not apply to the workspace servers or to the SAS metadata server.

"trustaspeer | tsaspeer = *pathname*"

The TRUSTSASPEER parameter enables SAS peer sessions from IOM servers to connect as trusted peer sessions.

This parameter is valid only for the command that starts the SAS Metadata Server. For more information, see Trusted Peer Connections in the *SAS Intelligence Platform: Security Administration Guide*.

pathname

is a pathname to a file containing XML that describes the allowable trusted peer connections, the users who can connect, and the machines from where the connections can be established.

Enclose the pathname in single quotation marks if there is a space in the path.

For example: 'C:\SAS 9.3\FoundationServers\Levl\SASMeta\MetaDataServer\trustedPeers.xml'.

Valid in: Metadata, Command Line

Alias: (no alias)

Default:

Restriction: This parameter is valid only for the command that starts the SAS Metadata Server.

Interaction: If you specify a blank or empty file, then you disable trusted peer support.

Details

All object server parameters are applicable on the command line that starts the server:

- In configuration files on z/OS, do *not* precede system options with a hyphen. Also, an equal sign must precede the first parameter that follows the OBJECTSERVERPARMS keyword. For example:

```
objectserver objectserverparms="cel=credentials protocol=bridge port=8561"
```

- For servers that are started by the object spawner, the object server parameters come from your server definition in the SAS Metadata Repository. (The server definition is located under the Server Manager plug-in of SAS Management Console. In the server definition, select the **Options** tab to locate the **Object Server Parameters** field).
- For servers that are not spawned (such as those that are run from command scripts or those that are run as services or daemons) you use the `sas -objectserver -objectserverparms "parameters"` invocation to specify the object server parameters on the command line.

To simplify the command that is needed to invoke an IOM server, the server start-up sequence by default connects back to the metadata server in order to fetch additional information, including object server parameters. You can fetch object server parameters from metadata as follows:

- When you start the server with a script, some object server parameters cannot be obtained from the metadata. (In the preceding table, only those parameters that are designated as “Metadata” in the “Valid In” column can be stored in the metadata.)
- When you start the server with a spawner, all object server parameters can be obtained from the metadata.
- Certain object server parameters are automatically added by the object spawner when it starts workspace and stored process servers. Therefore, the following parameters should not be stored in metadata for workspace and stored process servers: CLASSFACTORY, CLIENTENCRYPTIONLEVEL, PORT, PROTOCOL, SERVER, and SERVICE.

Note: Object server parameters that are specified on the command line always override object server parameters obtained from a SAS metadata repository.

Example

This example shows how the OBJECTSERVERPARMS system option can be used in a configuration file that is used to start the metadata server:

```
-objectserver
-objectserverparms "cel=credentials protocol=bridge port=8561 classfactory=
0217E202-B560-11DB-AD91-001083FF6836 trustsaspeer=C:\SAS\
FoundationServers\Lev1\SASMeta\MetadataServer\trustedPeers.xml"
```

Note: In configuration files on z/OS, do *not* precede system options with a hyphen.

Also, an equal sign must precede the first parameter that follows the OBJECTSERVERPARMS keyword. For example:

```
objectserver objectserverparms="cel=credentials protocol=bridge port=8561"
```

See Also

System Options:

- “[OBJECTSERVER System Option](#)” on page 139

SECPACKAGE System Option

Identifies the security package that the IOM server uses to authenticate incoming client connections.

Valid in:	configuration file, SAS invocation, metadata
Categories:	Environment control: Initialization and operation System Administration: Security
PROC OPTIONS GROUP=	EXECMODES SECURITY
Default:	negotiate
Restriction:	Windows operating environment only
See:	SECPACKAGELIST System Option SSPI System Option

Syntax

-secpackage "*package-name*" | "**negotiate**"

Syntax Description

"package-name"

specifies the security package that the IOM server should use to authenticate incoming client connections.

Enclose the security package name within double quotation marks (").

"negotiate"

(default) enables the server to present a set of valid security packages (through the SECPACKAGELIST system option) that the server uses to find a match with an incoming client connection. If the client specifies a security package in the list, then the server attempts to authenticate the client using the matched security package.

Enclose **negotiate** within double quotation marks (").

Details

The SECPACKAGE system option identifies the security package that the IOM server uses to authenticate incoming client connections.

Security packages are provided by vendors. Therefore, the package names are not validated against a list of names. Names need to be entered (casing and exact spelling) per instructions from the vendor.

When you specify **-SECPACKAGE "negotiate"**, the IOM server uses the SECPACKAGELIST option to determine which package to use. SECPACKAGELIST specifies the names of the security packages that can be used by the server to authenticate incoming client connections. SECPACKAGE and SECPACKAGELIST are

required to support single sign-on (SSO) to IOM servers. The client should initialize with a matching package name. Specifying an unknown package name (such as "disable") will effectively disable SSO.

In order to use SECPACKAGE, you must also specify **SSPI**.

Examples

Example 1

In the following example, the IOM server specifies either Kerberos or NTLM security for authenticating incoming client requests:

```
-sspi
-secpackage "negotiate"
-secpackagelist "Kerberos,NTLM"
```

Example 2

In the following example, the IOM server specifies Kerberos security only for authenticating incoming client requests:

```
-sspi
-secpackagelist "kerberos"
```

In the preceding example, SECPACKAGE does not have to be specified because it defaults to **negotiate**. The only protocol in the list to negotiate is Kerberos. Therefore, all clients that connect to the server must use Kerberos or fail the connection. It is important that the protocols of both the client and server match. The client is also forced to use Kerberos if the server displays only Kerberos in the package list.

Example 3

In the following example, the IOM server specifies NTLM security only for authenticating incoming client requests:

```
-sspi
-secpackagelist "ntlm"
```

In the preceding example, SECPACKAGE does not have to be specified because it defaults to **negotiate**. The only protocol in the list to negotiate is NTLM. Therefore, all clients that connect to the server must use NTLM or fail the connection. It is important that the protocols of both the client and server match. The client is also forced to use NTLM if the server displays only NTLM in the package list.

See Also

System Options:

- [“SECPACKAGELIST System Option” on page 150](#)
- [“SSPI System Option” on page 151](#)

Other SAS Documents:

- “Integrated Windows Authentication” in Chapter 10 of *SAS Intelligence Platform: Security Administration Guide*
- *SAS Companion for UNIX Environments*
- *SAS Companion for Windows*

SECPACKAGELIST System Option

Specifies the security authentication packages used by the server.

Valid in:	configuration file, SAS invocation, metadata
Categories:	System Administration: Security Environment control: Initialization and operation
PROC OPTIONS GROUP=	EXECMODES SECURITY
Default:	"Kerberos,NTLM"
Restriction:	Windows operating environment only
See:	SECPACKAGE System Option SSPI System Option

Syntax

```
-secpackagelist "package-name-1,[package-name-2],[...]"
```

Syntax Description

"package-name"

Identifies the security package that is used by the server in order to authenticate incoming client connections. The default is **"Kerberos,NTLM"**.

Enclose the security package name within double quotation marks ("). Delimit an additional package name with a comma (,).

Details

The SECPACKAGELIST system option, in conjunction with SECPACKAGE, identifies to the IOM server one or more security packages that can be used to authenticate incoming client connections. The default value of SECPACKAGELIST is Kerberos and NTLM.

To use the SECPACKAGELIST system option, SECPACKAGE must be set to **negotiate**. The IOM server requires these two security package options to support single sign-on (SSO) to IOM servers. The connecting client should initialize with a security package name that matches what you have specified on the server. The **negotiate** value allows the client and server to negotiate a site-specific package to use.

Examples

Example 1

In the following example, the IOM server specifies either Kerberos or NTLM security for authenticating incoming client requests:

```
-sspi
-secpackage "negotiate"
-secpackagelist "Kerberos,NTLM"
```

Example 2

In the following example, the IOM server specifies Kerberos security only for authenticating incoming client requests:

```
-sspi
-secpackagelist "kerberos"
```

In the preceding example, SECPACKAGE does not have to be specified because it defaults to **negotiate**. The only protocol in the list to negotiate is Kerberos. Therefore, all clients that connect to the server must use Kerberos or fail the connection. It is important that the protocols of both the client and server match. The client is also forced to use Kerberos if the server displays only Kerberos in the package list.

Example 3

In the following example, the IOM server specifies NTLM security only for authenticating incoming Windows client requests:

```
-sspi
-secpackagelist "ntlm"
```

In the preceding example, SECPACKAGE does not have to be specified because it defaults to **negotiate**. The only protocol in the list to negotiate is NTLM. Therefore, all clients that connect to the server must use NTLM or fail the connection. It is important that the protocols of both the client and server match. The client is also forced to use NTLM if the server displays only NTLM in the package list.

See Also**System Options:**

- [“SECPACKAGE System Option” on page 148](#)
- [“SSPI System Option” on page 151](#)

Other SAS Documents:

- “Integrated Windows Authentication” in Chapter 10 of *SAS Intelligence Platform: Security Administration Guide*
- *SAS Companion for UNIX Environments*
- *SAS Companion for Windows*

SSPI System Option

Identifies support for the Security Support Provider Interface for SSO connections to IOM servers.

Valid in:	configuration file, SAS invocation, metadata
Category:	System Administration: Security
PROC OPTIONS GROUP=	SECURITY
Default:	NOSSPI
Restriction:	Windows and UNIX operating environments only
See:	SECPACKAGE System Option SECPACKAGELIST System Option

Syntax

`-sspi` | `-nosspi`

Syntax Description

`-SSPI`

Support the Security Support Provider Interface (SSPI).

`-NOSSPI`

(Default) Do not support SSPI.

Details

Use this option to identify support for the Security Support Provider Interface for single sign-on (SSO) connections to IOM servers.

Example

In the following example, the IOM server specifies Security Support Provider Interface for SSO connections. By default, the IOM server uses either Kerberos or NTLM (Windows only) security for authenticating incoming client requests:

```
-sspi
```

See Also

System Options:

- [“SECPACKAGE System Option” on page 148](#)
- [“SECPACKAGELIST System Option” on page 150](#)

Other SAS Documents:

- “Integrated Windows Authentication” in Chapter 10 of *SAS Intelligence Platform: Security Administration Guide*
- *SAS Companion for UNIX Environments*
- *SAS Companion for Windows*

Chapter 15

IOMOPERATE Procedure

Overview: IOMOPERATE Procedure	154
Concepts: IOMOPERATE Procedure	154
Syntax: IOMOPERATE Procedure	154
PROC IOMOPERATE Statement	155
CONNECT Statement	157
CONTINUE Statement	165
DISCONNECT Statement	165
FLUSH AUTHORIZATION CACHE Statement	165
LIST Statement	166
PAUSE Statement	171
QUIESCE Statement	171
QUIT Statement	172
REFRESH CONFIGURATION Statement	172
RESET PERFORMANCE Statement	173
SET Statement	173
STOP Statement	174
Using: IOMOPERATE Procedure	175
About the IOMOPERATE Procedure	175
Connecting to an IOM Server	175
Disconnecting from an IOM Server	175
Examples: IOMOPERATE Procedure	175
Example 1: Basic CONNECT Example	175
Example 2: CONNECT Example Using Credentials	176
Example 3: CONNECT Example Using a URI and a Class Identifier	176
Example 4: CONNECT Example Using a Logical Server Name	176
Example 5: CONNECT Example with a Server Name	177
Example 6: CONNECT Example with a URI and Explicit Server Name	177
Example 7: CONNECT and LIST Examples Using a Spawner	177
Example 8: CONNECT Example Using Kerberos	178
Example 9: Pausing an Object Spawner	178
Example 10: Continuing an Object Spawner	179
Example 11: Quiescing an Object Spawner	179
Example 12: LIST TYPES Example	179
Example 13: Stopping a Metadata Server	180
Example 14: DISCONNECT Example	181
Example 15: FLUSH AUTHORIZATION CACHE Example	181
Example 16: REFRESH CONFIGURATION Example	181
Example 17: RESET PERFORMANCE Example	181
Example 18: SET Example	182

Overview: IOMOPERATE Procedure

The IOMOPERATE procedure administers SAS servers that support the SAS IOM infrastructure.

Concepts: IOMOPERATE Procedure

With the exception of the workspace server, all the SAS IOM servers support the same IOM administrative interfaces. Various procedures, such as PROC METAOPERATE and PROC TSOPERATE, are used to administer the servers. However, these procedures are limited to a single server type. The IOMOPERATE procedure can be used to administer all IOM servers.

Three distinct types of commands can be issued via PROC IOMOPERATE:

- Those that perform some action on the server, such as: stopping, continuing, and setting server attribute values.
- Those that print information about the server, such as: listing sessions, listing clients, and listing UUIDs.
- Those that pertain only to spawners, such as: listing spawned servers and stopping spawned servers.

These commands do not pertain to all SAS IOM servers. The commands that a particular SAS IOM server supports depends on the underlying interfaces that the server supports. After you connect to a server, invoke LIST COMMANDS to determine which commands you can run on the server.

Syntax: IOMOPERATE Procedure

See: METAOPERATE, TSOPERATE

```
PROC IOMOPERATE;  
  CONNECT connect-options;  
  <CONTINUE (SERVER | CLUSTER); >  
  <DISCONNECT; >  
  <FLUSH AUTHORIZATION CACHE ;>  
  <LIST list-options; >  
  <PAUSE (SERVER | CLUSTER);>  
  <QUIESCE (SERVER | CLUSTER);>  
  QUIT;  
  <REFRESH CONFIGURATION; >  
  <RESET PERFORMANCE; >  
  <SET set-options; >  
  <STOP stop-options; >
```

Statement	Task	Example
PROC IOMOPERATE Statement	Administers SAS servers that support the SAS IOM infrastructure.	Ex. 1
CONNECT Statement	Connects to a SAS IOM server.	Ex. 1
CONTINUE Statement	Continues a paused server or cluster.	Ex. 10
DISCONNECT Statement	Disconnects from the current server.	Ex. 14
FLUSH AUTHORIZATION CACHE Statement	Clears the authorization cache on the connected server.	Ex. 15
LIST Statement	Lists to the console various attributes of the connected server.	Ex. 7, Ex. 12
PAUSE Statement	Suspends the connected server or cluster.	Ex. 9
QUIESCE Statement	Stops the connected server or cluster when all of the current work is finished.	Ex. 11
QUIT Statement	Terminates the IOMOPERATE procedure.	Ex. 1
REFRESH CONFIGURATION Statement	Causes the object spawner to reread its configuration from the metadata server.	Ex. 16
RESET PERFORMANCE Statement	Resets the performance metrics being tracked by the connected server.	Ex. 17
SET Statement	Modifies various attributes of the connected server.	Ex. 18
STOP Statement	Shuts down the connected server or cluster.	Ex. 13

PROC IOMOPERATE Statement

Administers SAS servers that support the SAS IOM infrastructure.

- Examples:**
- “Example 1: Basic CONNECT Example” on page 175
 - “Example 7: CONNECT and LIST Examples Using a Spawner” on page 177
 - “Example 4: CONNECT Example Using a Logical Server Name” on page 176
 - “Example 9: Pausing an Object Spawner” on page 178
 - “Example 11: Quiescing an Object Spawner” on page 179
 - “Example 12: LIST TYPES Example” on page 179
 - “Example 13: Stopping a Metadata Server” on page 180

Syntax

```
PROC IOMOPERATE;  
    CONNECT <connect-options>;  
    <optional-argument(s)>;  
    QUIT;
```

Required Arguments

CONNECT <connect-options>;
For more information, see [CONNECT Statement on page 157](#).

QUIT;
For more information, see [QUIT Statement on page 172](#).

Optional Arguments

<**CONTINUE (SERVER | CLUSTER)**>;
For more information, see [CONTINUE Statement on page 165](#).

<**DISCONNECT**>;
For more information, see [DISCONNECT Statement on page 165](#).

<**FLUSH AUTHORIZATION CACHE**>;
For more information, see [FLUSH AUTHORIZATION CACHE Statement on page 165](#).

<**LIST** <list-options>>;
For more information, see [LIST Statement on page 166](#).

<**PAUSE (SERVER | CLUSTER)**>;
For more information, see [PAUSE Statement on page 171](#).

<**QUIESCE (SERVER | CLUSTER)**>;
For more information, see [QUIESCE Statement on page 171](#).

<**REFRESH CONFIGURATION**>;
For more information, see [REFRESH CONFIGURATION Statement on page 172](#).

<**RESET PERFORMANCE**>; >;
For more information, see [RESET PERFORMANCE Statement on page 173](#).

<**SET** <set-options>>;
For more information, see [SET Statement on page 173](#).

<**STOP** <stop-options>>;
For more information, see [STOP Statement on page 174](#).

Details

Using PROC IOMOPERATE

The keywords PROC IOMOPERATE precede all statements.

A semicolon (;) ends each statement.

The keyword, QUIT terminates the IOMOPERATE procedure.

For a usage example, see “[Example 1: Basic CONNECT Example](#)” on page 175.

CONNECT Statement

Provides the required information to establish a session with a SAS IOM server.

See: [“SECPACKAGE System Option” on page 148](#), [“SECPACKAGELIST System Option” on page 150](#)

Examples: [“Example 1: Basic CONNECT Example” on page 175](#)
[“Example 2: CONNECT Example Using Credentials” on page 176](#)
[“Example 3: CONNECT Example Using a URI and a Class Identifier” on page 176](#)
[“Example 4: CONNECT Example Using a Logical Server Name” on page 176](#)
[“Example 5: CONNECT Example with a Server Name” on page 177](#)
[“Example 6: CONNECT Example with a URI and Explicit Server Name” on page 177](#)
[“Example 7: CONNECT and LIST Examples Using a Spawner” on page 177](#)
[“Example 8: CONNECT Example Using Kerberos” on page 178](#)

Syntax

CONNECT

```
<LOGICALSERVERNAME="logical-server-name">
<URI="uniform-resource-identifier">
<HOSTNAME="host-name">
<PORT=port>
<PROTOCOL=BRIDGE | COM>
<USERNAME="user-ID">
<PASSWORD="password">
<IOMOPTIONS="IOM-option, ...">
<CLSID=class-UUID>
<SERVERTYPE=type-name>
<SPAWNED="server-UUID">
<CLUSTER NAME= "cluster-name">
<CLUSTERSERVER ID="cluster-server-UUID">
;
```

Optional Arguments

LOGICALSERVERNAME=

specifies the logical server to connect to. SAS returns the first successful connection.

When you specify LOGICALSERVERNAME=, SAS ignores all other CONNECT options except for USERNAME= and PASSWORD=.

logical-server-name

is an alphanumeric string that is the logical name for the server that you want to connect to.

Alias: LOGICAL=

Default: The default metadata connection for the logical server.

Note: Enclose *logical-server-name* in quotes.

See: “[Example 4: CONNECT Example Using a Logical Server Name](#)” on page 176

Example:

```
logical="App Server - Logical Workspace Server"
```

URI=

specifies the server Uniform Resource Identifier (URI) that SAS uses to connect to the server.

When you specify URI=, SAS ignores HOST=, PORT=, PROTOCOL=, and IOMOPTIONS=. All connection options can be specified in the URI, but it can be useful to separate PASSWORD= (so that it will be blotted in the log) and CLSID= and SERVERTYPE= (so that the symbolic names can be used instead of the actual GUIDs).

"uniform-resource-identifier"

is an alphanumeric string that is the URI for the server that you want to connect to. A valid URI takes the form,

```
"iom://hostname:port"
```

Notes:

When URI= is specified, PORT= is not required.

Enclose *uniform-resource-identifier* in quotes.

The name-value pairs listed under IOMOPTIONS= on page 159 can be specified as part of URI.

See: “[Example 6: CONNECT Example with a URI and Explicit Server Name](#)” on page 177

Example:

```
uri="iom://myappserver1234:8581"
```

HOSTNAME=

specifies the machine or host to connect to. The value `localhost` can be used if the SAS session is connecting to a server on the same computer.

"host-name"

specifies the host name, fully qualified host name, or IP address of the machine hosting the server.

Alias: HOST=

Note: Enclose *host-name* in quotes.

See: “[Example 5: CONNECT Example with a Server Name](#)” on page 177

Example:

```
host="myhost.example.com"
```

PORT=

specifies the TCP port to which the specified server listens for connections (for example, PORT=2171). This option is not required if the COM protocol is specified. This port uses a default based on the specified CLSID= or SERVERTYPE=.

port

is a valid TCP port on the specified server.

Default: When a port is required but not specified, IOMOPERATE defaults to a value based on the CLSID= or SERVERTYPE=.

Note: Not required when you specify PROTOCOL=COM or URI=.

See: [“Example 5: CONNECT Example with a Server Name” on page 177](#)

Example:

```
port=2171
```

PROTOCOL=

specifies the network protocol for communicating with the SAS Server.

BRIDGE

specifies that the connection uses IOM Bridge protocol.

COM

specifies that the connection uses the COM protocol.

Default: BRIDGE

Note: When COM is specified, PORT= is not required.

See: [“Example 3: CONNECT Example Using a URI and a Class Identifier” on page 176](#)

Example:

```
protocol=com
```

USERNAME=

specifies a user ID that CONNECT uses to access the specified server. If `_PROMPT_` is specified without quotes, then SAS prompts for a user ID.

"user-ID"

is a valid user ID that has the required privileges to access the specified server.

Alias: USERID=, USER=, UID=

Note: Enclose *user-ID* in quotes.

See: [“Example 2: CONNECT Example Using Credentials” on page 176](#)

Example:

```
uid="sasdemo"
```

PASSWORD=

specifies the password associated with the user ID. If `_PROMPT_` is specified without quotes, then SAS prompts for a password.

"password"

is the password associated with the specified user ID.

Alias: PASSWD=, PASS=, PWD=, PW=

Note: Enclose *password* in quotes.

See: [“Example 2: CONNECT Example Using Credentials” on page 176](#)

Example:

```
pw="mypass"
```

IOMOPTIONS=

specifies a string of name-value pairs to use when connecting to the server.

"IOM-option"

is a valid name-value pairs consist of the following:

CLASSFACTORY=*class-UUID*

specifies the type of server being connected to.

This is an optional parameter and is required only when connecting to a SAS 9.2 or earlier server. When connecting to SAS 9.3 and later servers, this parameter can be omitted.

The [LIST TYPES command on page 171](#) displays the full name, short name, and UUID of the known server types. Any of the UUIDs can be used with CLASSFACTORY.

class-UUID

is a 36-character alphanumeric string that represents a valid class unique universal identifier (UUID) for the server.

Alias: CLSID

Note: Enclose the option in quotes. Separate multiple options with a comma.

See: “[Example 3: CONNECT Example Using a URI and a Class Identifier](#)” on page 176

Example:

```
opts="clsid=0E3B1810-6646-11D5-8863-00C04F48BC53,
      user=myacct,pass=mypwd"
```

DOMAIN=*name*

specifies the authentication domain in which to select credentials from the metadata to present at connect time.

name

is an alphanumeric string that equates to a valid domain name.

Note: The value of the DOMAIN never reaches the target IOM server.

Example:

```
opts='servername=myserver,domain=mydomain'
```

ENCR=*algorithm*

specifies the encryption algorithm to request when connecting to the peer identified in this IOM URI.

algorithm

is a valid encryption algorithm. Valid algorithms are: SASPROPRIETARY, SAS/SECURE, RC2, RC4, DES, TripleDES, and AES.

Note: Enclose the option in quotes. Separate multiple options with a comma.

Example:

```
opts='encr=aes,encrlvl=1,user=myusername,
      pass=mypswd'
```

ENCRLVL=*encryption-level*

specifies the level of encryption requested by the client when connecting to the peer identified in this IOM URI.

encryption-level

is a valid encryption level. Valid encryption levels are: none, credentials, and everything.

- none: the client requests no encryption.
- credentials: the client requests to encrypt only credentials.
- everything: the client requests to encrypt every packet.

Note: Enclose the option in quotes. Separate multiple options with a comma.

Example:

```
opts='encr=aes,encrlvl=1,user=myusername,
      pass=mypswd'
```

INTERFACEIID=*UUID*

UUID

is a 36-character alphanumeric string that represents a valid UUID of the desired interface within the object acquired.

Alias: IID

Note: Enclose the option in quotes. Separate multiple options with a comma.

Example:

```
opts='iid=0E3B1810-6646-11D5-8863-00C04F48BC53,
      user=myusername,pass=mypswd'
```

LOCALE=*locale-name*

The value for the LOCALE option specifies the locale of the connecting peer.

locale-name

is a valid locale.

Note: Enclose the option in quotes. Separate multiple options with a comma.

Example:

```
opts='locale=french,
      servername=SASApp - Stored Process Server'
```

MAJOR=*major-version-number*

specifies the major portion of the bridge protocol version to use.

major-version-number

is a number that represents the major (tenths) portion of the bridge protocol version.

Note: Enclose the option in quotes. Separate multiple options with a comma.

Example: For SAS 9.3 this would be MAJOR=3 MINOR=0.

MINOR=*minor-version-number*

specifies the minor portion of the bridge protocol version to use.

minor-version-number

is a number that represents the minor (hundredths) portion of the bridge protocol version. When a minor version number does not exist, use the number zero.

Note: Enclose the option in quotes. Separate multiple options with a comma.

Example: For SAS 9.3 this would be MAJOR=3 MINOR=0.

NOREDIRECT

disables any load-balancing redirection that would be performed by the server. Use this option with caution. It should be used only when attempting to get an administrative connection to a load-balanced server.

Example:

```
opts='noredirect,
      servername=SASApp - Stored Process Server'
```

USER=*user-ID*

specifies the identity to use when connecting to the peer identified in this IOM URI.

user-ID

is a valid metadata identity with permission to access the peer.

Note: Enclose the option in quotes. Separate multiple options with a comma.

Example:

```
opts='user=myusername,pass=mypswd,
servername=SASApp - Stored Process Server'
```

PASS=*password*

specifies the password for the identity to use when connecting to the peer identified in this IOM URI.

password

is a valid password for the metadata identity used to access the peer.

Note: Enclose the option in quotes. Separate multiple options with a comma.

Example:

```
opts='user=myusername,pass=mypswd,
servername=SASApp - Stored Process Server'
```

SECURITYPACKAGE=*package-name* | NEGOTIATE

identifies the security package that the IOM server uses to authenticate incoming client connections.

package-name

specifies the security package that the IOM server should use to authenticate incoming client connections. Valid values are Kerberos and NTLM.

NEGOTIATE

enables the server to present a set of valid security packages (through the SECPACKAGELIST system option) that the server uses to find a match with an incoming client connection. If the client specifies a security package in the list, then the server attempts to authenticate the client using the matched security package.

Default: NEGOTIATE

Note: Enclose the option in quotes. Separate multiple options with a comma.

See: [“SECPACKAGE System Option” on page 148](#)

Example:

```
opts='sspi,securitypackage=negotiate,securitypackagelist=Kerberos,NTLM'
```

SECURITYPACKAGELIST=*package-name*, ...

specifies the security authentication packages used by the server.

Note: Enclose the option in quotes. Separate multiple options with a comma.

To use the SECPACKAGELIST system option, SECPACKAGE must be set to **negotiate**.

See: [“SECPACKAGELIST System Option” on page 150](#)

Example:

```
opts='sspi,securitypackage=negotiate,securitypackagelist=Kerberos,NTLM'
```

SERVERNAME=*server-name*

specifies the name of the server to which you want to connect.

server-name

is a valid server name or a unique URI that is stored in metadata.

Note: Enclose the option in quotes. Separate multiple options with a comma.

Example:

```
opts='servername=SASApp - Stored Process Server,timeout=3600'
```

SPN=*name*

specifies the service principal name that the client wishes to use with this IOM Server instance.

name

is a valid service principal name.

Note: Enclose the option in quotes. Separate multiple options with a comma.

Example:

```
opts='spn=SAS [Config-Lvl] SASMeta - Metadata Server,timeout=3600'
```

SSPI

identifies support for the Security Support Provider Interface for single sign-on (SSO) connections to IOM servers.

Note: Use SSPI in conjunction with SECURITYPACKAGE and SECURITYPACKAGELIST.

See: [SECURITYPACKAGE=package-name | NEGOTIATE on page 162](#) , [SECURITYPACKAGELIST=package-name, on page 162](#)

Example:

```
opts='sspi,securitypackage=negotiate,securitypackagelist=Kerberos,NTLM'
```

TIMEOUT=*milliseconds*

specifies the time-out, in milliseconds, of all outcall activity.

milliseconds

a number that represents the number of milliseconds to wait before abandoning the server connection.

Note: Enclose the option in quotes. Separate multiple options with a comma. The value of the TIMEOUT never reaches the target IOM server.

Example:

```
opts='timeout=3600,servername=SASApp - Stored Process Server'
```

TRUSTEDSAS

indicates that the owner of the current IOM Server is to be used as the identity when connecting to the peer identified in this IOM URI.

Alias: IOMOPTS=, OPTS=

Note: Enclose *IOM-option* in quotes. Separate multiple options with a comma.

See: [“Example 8: CONNECT Example Using Kerberos” on page 178](#)

Example:

```
opts='servername=SASApp - Stored Process Server,timeout=3600'
```

CLSID=

specifies the type of server being connected to.

This is an optional parameter and is required only when connecting to a SAS 9.2 or earlier server. When connecting to SAS 9.3 and later servers, this parameter can be omitted.

The [LIST TYPES command on page 171](#) displays the full name, short name, and UUID of the known server types. Any of the UUIDs can be used with CLSID.

class-UUID

is a 36-character alphanumeric string that represents a valid class UUID for the server.

Note: When used as a connection parameter, enclose *class-UUID* in quotes.

See: [“Example 3: CONNECT Example Using a URI and a Class Identifier” on page 176](#)

Example:

```
clsid="0E3B1810-6646-11D5-8863-00C04F48BC53"
```

SERVERTYPE=

specifies the type of server being connected to.

This is an optional parameter and is required only when connecting to a SAS 9.2 or earlier server. When connecting to SAS 9.3 and later servers, this parameter can be omitted.

The [LIST TYPES command on page 171](#) displays the full name, short name, and UUID of the known server types. Any of the types can be used with **SERVERTYPE**.

"type-name"

an alphanumeric string that represents a valid type for the server.

Alias: TYPE=

See: [“Example 5: CONNECT Example with a Server Name” on page 177](#)

Example:

```
type=storedprocess
```

SPAWNED=

specifies the UUID of a spawned server.

This parameter is valid only when connecting to an object spawner. It has the effect of connecting through the spawner to the spawned server.

"UUID"

is a 36-character alphanumeric string that represents a valid UUID (unique universal identifier) for the spawner.

Alias: LAUNCHED=

Note: Enclose *UUID* in quotes. Because you are connecting to the object spawner, any other connection options that follow (like **PORT=**, **TYPE=**, and so on) refer to the spawner.

See: [“Example 7: CONNECT and LIST Examples Using a Spawner” on page 177](#)

Example:

```
spawned="E3D5D132-5DC0-46CB-A4FD-713098B1EE95"
```

CLUSTER NAME=

specifies the name of a cluster to connect to on the given server. This is useful when connecting to a load balanced server.

"cluster-name"

is the name of the cluster.

Alias: CLUSTER=

Note: Enclose *cluster-name* in quotes.

Example:

```
cluster="SASApp - Logical Stored Process Server"
```

CLUSTERSERVER ID=

specifies the UUID of the server in the given cluster to connect to.

"cluster-server-UUID"

a 36-character alphanumeric string that represents a valid UUID for a server in the cluster.

Alias: CLUSTERSERVER=

Requirement: You must also use the **CLUSTER NAME=** argument.

Note: Enclose *cluster-server-UUID* in quotes.

Example:

```
clusterserver="15931e31-667f-11d5-8804-00c04f35ac8c"
```

CONTINUE Statement

Continues a paused server or cluster.

Requirement: You must be the owner of the server or cluster process or have the administer permission in SAS metadata for the server or cluster.

Example: [“Example 10: Continuing an Object Spawner” on page 179](#)

Syntax

CONTINUE

SERVER | **CLUSTER**;

Required Arguments

SERVER

continues a paused server.

CLUSTER

continues a paused cluster.

DISCONNECT Statement

Terminates the session with the currently connected server.

See: [QUIT Statement on page 172](#)

Example: [“Example 14: DISCONNECT Example” on page 181](#)

Syntax

DISCONNECT;

Without Arguments

terminates the session with the currently connected server.

FLUSH AUTHORIZATION CACHE Statement

Clears the authorization cache on the connected server.

Requirement: You must be the owner of the server or cluster process or have the administer permission in SAS metadata for the server.

Example: [“Example 15: FLUSH AUTHORIZATION CACHE Example” on page 181](#)

Syntax

FLUSH AUTHORIZATION CACHE;

Without Arguments

clears the authorization cache on the connected server.

Alias: FLUSH AUTH

LIST Statement

Lists to the console various attributes of the connected server.

Examples: “Example 7: CONNECT and LIST Examples Using a Spawner” on page 177
“Example 12: LIST TYPES Example” on page 179

Syntax**LIST**

```
<ABANDONED SERVERS <FILTER=string><OUT=data-set-name>>;
<ATTRIBUTES <CATEGORY=category><NAME=name><FILTER=string><VERBOSE>>;
<CATEGORIES <OUT=data-set-name>>;
<CLASSIDS <OUT=data-set-name>>;
<CLIENTS <OUT=data-set-name>>;
<CLUSTERS <OUT=data-set-name>>;
<COMMANDS <OUT=data-set-name>>;
<CONTROL INFORMATION <OUT=data-set-name>>;
<DEFINED SERVERS <VERBOSE><FILTER=string><OUT=data-set-name>>;
<DNSNAME <OUT=data-set-name >>;
<INFORMATION <OUT=data-set-name>>;
<LOG <FILTER="filter">< OUT=data-set-name>>;
<LOG CONFIGURATION <OUT=data-set-name>>;
<NAME <OUT=data-set-name>>;
<PERFORMANCE <OUT=data-set-name>>;
<SERVERTIME <OUT=data-set-name>>;
<SESSIONS <OUT=data-set-name>>;
<SPAWNED SERVERS <FILTER= string>< OUT=data-set-name>>;
<STATE <OUT=data-set-name>>;
<TYPES <OUT=data-set-name>>;
<UNIQUEID <OUT=data-set-name>>;
```

Optional Arguments**ABANDONED SERVERS**

lists servers that were abandoned by the object spawner. You can use ABANDONED SERVERS only when you are connected to an object spawner.

ABANDONED SERVERS returns the logical name, server component name, server class UUID, process owner user ID, and the unique ID of the server as known by the spawner.

FILTER=

specifies a substring that is matched against logical name and server component name.

The LIST statement returns only those servers with logical or component names matched by the filter.

string

is an alphanumeric string that attempts to match server logical or component names.

Note: The filter does not support wildcard characters. The search is case-insensitive.

OUT=

writes LIST statement output to a SAS data set.

data-set-name

is the name of a valid SAS data set or the form **libname.data-set-name**. *data-set-name* can be a regular data set name or take the form:

libname.datasetname.

SAS creates the data set if it does not exist and overwrites it if it does exist. There is no requirement on location. If you can successfully reference the data set in a SAS DATA STEP statement, then you can use the data set here.

Requirement: You must have the proper permissions to write the data set.

Note: There is no requirement about where the data set is located.

Alias: ABANDONED

ATTRIBUTES

lists the attributes of the connected server.

CATEGORY=

specifies an attribute category used to match a list of server attributes

category

is an alphanumeric, case-sensitive string that represents a valid attribute category.

Alias: CAT=

Default: Appenders, Counters, Information, Loggers, and Properties.

Note: You must also specify a category when using NAME=.

Example:

```
list attrs cat = "Counters";
```

NAME=

specifies a name used to match a specific attribute name.

name

is an alphanumeric, case-sensitive string that represents a valid attribute name.

Note: You must also specify a category when using NAME=.

Example:

```
list attrs cat = "Information" name = "Server.ProcessIdentifier";
```

FILTER=

specifies a string used to match a specific attribute name.

string

is an alphanumeric, case-sensitive string that attempts to match server attribute categories.

Notes:

For all categories, the filter only returns a match if the specified string ends with a period. For example, the filter string **App** will return a match for **App.Program**.

FILTER attempts to match attributes across all categories, unless you also specify CATEGORY= to narrow the match to one category.

Example:

```
list attrs filter = "Server.";
```

VERBOSE

indicates that type, description, and the writability of the attribute should be returned in addition to attribute name and value.

Alias: ATTRS

Requirement: You must be the owner of the server or cluster process or have the administer permission in SAS metadata for the server to see all attributes.

CATEGORIES

lists the attribute categories that are supported by the connected server.

CLASSIDS

lists the class UUIDs supported by the connected server.

Alias: CLSIDS

CLIENTS

lists the clients that are currently connected to the server.

CLUSTERS

lists the load-balanced clusters defined in the connected server.

COMMANDS

lists all of the commands that are available based on the IOM interfaces supported by the connected server.

CONTROL INFORMATION

lists some of the operations supported by the connected server.

Alias: CONTROL INFO

DEFINED

lists the servers defined in the connected server. If connected to an object spawner, this argument lists the servers that the spawner can launch.

FILTER=

specifies a substring that is matched against logical name and server component name.

string

is an alphanumeric string that attempts to match server logical or component names.

Note: The filter does not support wildcard characters. The search is case-insensitive.

Alias: SERVERS, DEFINED SERVERS

DNSNAME

lists the domain name service (DNS) facility of the connected server.

INFORMATION

lists some information about the connected server.

Alias: INFO

LOG

attempts to list the current log from the connected server.

FILTER=

specifies the filter with which a match is attempted.

If no filter is specified, then "columns=message" is used as the default.

"filter"

is a series of name-value pairs that are delimited by spaces. Valid name-value pairs are as follows:

name=logger-name

specifies a logger.

Default: When **name=** is not specified, all loggers are listed.

Example:

```
list log filter='name=App.Server.Logger'
```

start=ddmmmyyyy:hh:mm:ss

specifies the starting date and time of log entries to be listed.

Example:

```
list log filter="start=01aug2011:08:00:00 name=App.Server.Logger"
```

end=ddmmmyyyy:hh:mm:ss

specifies the ending date and time of log entries to be listed.

Default: When **end=** is not specified, the current date and time is used.

Example:

```
list log filter='end=31aug2011:17:00:00 name=App.Server.Logger'
```

level=<TRACE>, <DEBUG>, <INFO>, <WARN>, <ERROR>, <FATAL>

specifies the log level of log entries to list.

Default: When **level=** is not specified, all levels are listed.

Example:

```
LIST LOG filter="level=warn,error name=App.Server.Logger"
```

threshold=<TRACE> | <DEBUG> | <INFO> | <WARN> | <ERROR> | <FATAL>

specifies the minimum level of log entries to list.

Note: The value **level=** takes precedence over **threshold=**. When **threshold=** is not used, all levels are listed.

Example:

```
list log filter="threshold=error name=App.Server.Logger"
```

"columns=(<name>, <level>, <message>, <datetime>)"

specifies one or more logger columns to list.

Default: When no column name is specified, **message** is the default.

Example:

```
list log filter='columns=(name,datetime) name=App.Server.Logger'
```

LOG CONFIGURATION

lists the baseline configuration being used by the logging system in the connected server.

LOG CONFIGURATION does not list modifications that might have been made by code or by submitting a partial configuration file.

Alias: LOG CONFIG, LOG CFG

Note: The configuration is transcoded into the current session locale and, if it is printing to the log, the output is in XML format.

NAME

lists the name of the connected server.

PERFORMANCE

lists the following performance metrics for the connected server:

TotalRealExecutionTime	total time spent on calls.
TotalSystemCPUTime	total system CPU time spent on calls.
TotalUserCPUTime	total user CPU time spent on calls.
TotalCalls	total calls to the server.
ElapsedObjectTime	elapsed time since the server object was launched.
ElapsedServerTime	elapsed time since the server object was last started.
Status	current state of server. The potential states are:

NotStarted	0	Server is still being configured by an administrator.
StartPending	1	Server has received start command but has not finished processing it.
Running	2	Server has been started and is accepting all requests.
PausePending	3	Server has received pause command but has not finished processing it.
Paused	4	Server has been paused and is not accepting new work.
ContinuePending	5	Server has received continue command but has not finished processing it.
DeferredStop	6	Server is not accepting new sessions and will stop when all client work is complete.
StopPending	7	Server is not accepting new work and is in the process of shutting down.
Stopped	8	Server is down.

Alias: PERF

SERVETIME

lists the current time as it is known by the connected server.

Alias: STIME

SESSIONS

lists the active sessions in the connected server.

SPAWNED SERVERS

lists the servers that have been spawned by this spawner. This option is supported only when the specified server is connected to an object spawner.

FILTER=

specifies a substring that is matched against logical name and server component name. The LIST statement returns only those servers with logical or component names matched by the filter.

string

is an alphanumeric string that attempts to match server logical or component names.

Note: The filter does not support wildcard characters. The search is case-insensitive.

Alias: SPAWNED, LAUNCHED

STATE

lists the state of the connected server.

TYPES

lists the server types known to IOMOPERATE.

LIST TYPES prints the server name, short name, and class UUID for each server known by IOMOPERATE. Any of these can be passed as the [TYPE=](#) on page 164 parameter when connecting to a server.

Note: LIST TYPES does not require a connection. It is a client-side operation and only lists the server types that PROC IOMOPERATE knows about.

UNIQUEID

lists the UUID for the server.

PAUSE Statement

Suspends the connected server or cluster.

Requirement: You must be the owner of the server or cluster process or have the administer permission in SAS metadata for the server or cluster.

Example: [“Example 9: Pausing an Object Spawner” on page 178](#)

Syntax

PAUSE

[SERVER](#) | [CLUSTER](#);

Required Arguments

SERVER

suspends the connected server.

CLUSTER

suspends the connected cluster.

QUIESCE Statement

Stops the connected server or cluster once all of the current work is finished.

Requirement: You must be the owner of the server or cluster process or have the administer permission in SAS metadata for the server or cluster.

Note: When QUIESE is executed, the server or cluster does not interrupt work currently being done, but no new client requests are accepted.

Example: [“Example 11: Quiescing an Object Spawner” on page 179](#)

Syntax

QUIESCE

[SERVER | CLUSTER](#);

Required Arguments

SERVER

stops the connected server when all of the current work is finished.

CLUSTER

stops the connected cluster when all of the current work is finished.

QUIT Statement

Terminates the IOMOPERATE procedure.

See: [DISCONNECT Statement on page 165](#)

Example: [“Example 1: Basic CONNECT Example” on page 175](#)

Syntax

QUIT;

Without Arguments

causes SAS to execute an automatic server disconnect.

Note: It is not necessary to explicitly disconnect from the server before terminating the procedure.

REFRESH CONFIGURATION Statement

Causes the object spawner to reread its configuration from the metadata server.

Example: [“Example 16: REFRESH CONFIGURATION Example” on page 181](#)

Syntax

REFRESH CONFIGURATION;

Without Arguments

causes the spawner to reread its configuration from the metadata server.

Alias: REFRESH CONFIG

Note: Available only when connected to an object spawner.

RESET PERFORMANCE Statement

Resets the performance metrics being tracked by the connected server.

Example: [“Example 17: RESET PERFORMANCE Example” on page 181](#)

Syntax

RESET PERFORMANCE;

Without Arguments

resets the performance metrics being tracked by the connected server.

Alias: RESET PERF

SET Statement

Modifies various attributes of the connected server.

Example: [“Example 18: SET Example” on page 182](#)

Syntax

SET

<ATTRIBUTE CATEGORY=*category* NAME=*name* VALUE=*value*>;

<LOG CONFIGURATION=*file-reference* | "*path*">;

Optional Arguments

ATTRIBUTE CATEGORY=

sets the value of an attribute of the connected server.

category

is an alphanumeric string that represents a valid category to which the attribute belongs.

Alias: ATTR CAT=

NAME=

sets the value of the name attribute for the connected server.

name

is an alphanumeric string that represents the valid name for the attribute being set.

VALUE=

This command sets the value of an attribute of the connected server.

value

is an alphanumeric string that represents the value to which the attribute is being set.

Alias: VAL=

LOG CONFIGURATION=

is used to set a SAS logging facility configuration file on the connected server. This file can be a partial configuration or a complete replacement configuration. For more information, see *SAS Logging: Configuration and Programming Reference*.

file-reference

is a valid logging appender reference.

"path"

is a valid path.

Alias: LOG CONFIG=, LOG CFG=

STOP Statement

Shuts down the connected server or cluster.

Requirement: You must be the owner of the server or cluster process or have the administrator permission in SAS metadata for the server or cluster.

Example: [“Example 13: Stopping a Metadata Server” on page 180](#)

Syntax**STOP**

<(SERVER | CLUSTER);>

<SESSION ID=*session-ID* <CLUSTERSERVER=*cluster-server-ID*>>;

<SPAWNED SERVER ID=*server-ID*>;

Optional Arguments**SERVER**

stops the currently connected server.

CLUSTER

stops the currently connected cluster.

SESSION ID=

stops a session in the currently connected server.

session-ID

is a UUID that is a valid session identifier.

CLUSTERSERVER ID=

stops a session in the currently connected cluster.

cluster-server-UUID

is a valid UUID (unique universal identifier) for the cluster.

Alias: CLUSTERSERVER=

SPAWNED SERVER ID=

stops a spawned server.

server-UUID

is a valid UUID (unique universal identifier) for the server.

Alias: SPAWNED ID=, LAUNCHED ID=

Note: This option is supported only when connected to an object spawner.

Using: IOMOPERATE Procedure

About the IOMOPERATE Procedure

The IOM infrastructure provides basic administration interfaces that can be used by server implementations. These interfaces are common across many (if not all) of the SAS IOM servers.

Connecting to an IOM Server

You establish a connection to a running SAS IOM Server by providing connection information about the PROC IOMOPERATE statement or by running the CONNECT command. All other IOMOPERATE commands require a connection to an IOM Server. Once connected, all subsequent commands apply to that server until a DISCONNECT or STOP SERVER command is executed.

Disconnecting from an IOM Server

After you have disconnected from the server, the only other command that you can execute is the CONNECT command to connect to another (or the same) server. When the IOMOPERATE procedure is terminated with the QUIT command, an automatic server disconnect is executed. It is not necessary to explicitly disconnect from the server before terminating the procedure.

Examples: IOMOPERATE Procedure

Example 1: Basic CONNECT Example

Basic CONNECT Example

This is one example of how you might connect to a SAS object spawner:

```
PROC IOMOPERATE;
  CONNECT uri='iom://itsvista16234:8571;Bridge;
           USER=sasiom1,
           PASS=mypassword'
           servertype=OBJECTSPAWNER;
  LIST COMMANDS;
QUIT;
```

Example 2: CONNECT Example Using Credentials

CONNECT Example Using Credentials

The following example demonstrates how to include a user ID and its password to connect to a SAS object spawner:

```
PROC IOMOPERATE;
  CONNECT host='itsvista16234'
         port=8571
         user='sasiom1'
         pass='mypassword'
         servertime=OBJECTSPAWNER;
  LIST COMMANDS;
QUIT;
```

Example 3: CONNECT Example Using a URI and a Class Identifier

CONNECT Example Using a URI and a Class Identifier

In the following example, a URI and class identifier is specified to identify the server to connect to:

```
PROC IOMOPERATE
  uri='iom://itsvista16234:8571;Bridge;
  CLSID="0E3B1810-6646-11D5-8863-00C04F48BC53",
  USER=sasiom1,
  PASS=mypassword'
  LIST COMMANDS;
QUIT;
```

Example 4: CONNECT Example Using a Logical Server Name

CONNECT Example Using a Logical Server Name

The following example shows how you can use a logical server name to identify a server to connect to:

```
options metaserver="visat64"
       metaport=8561
       metauser="sasadm@saspw"
       metapass="*****";
proc iomoperate logical="9.3 WIN App Server - Logical Workspace Server";
  list info;
quit;
```

Example 5: CONNECT Example with a Server Name

CONNECT Example with a Server Name

The following example shows how you can use a server name to identify the server to connect to. An explicit server name is required when there are multiple servers listening on the same spawner port:

```
PROC IOMOPERATE;
    CONNECT host='itsvista16234'
           port=8591
           user='sasiom1'
           pass='mypassword'
           servertype=STOREDPROCESS
           iomoptions='SERVERNAME=SASApp - Stored Process Server';
    LIST COMMANDS;
QUIT;
```

Example 6: CONNECT Example with a URI and Explicit Server Name

CONNECT Example with a URI and Explicit Server Name

The following example uses a URI and a server name to identify the server to connect to:

```
PROC IOMOPERATE
    uri="iom://itsvista16234:8591;Bridge;
    USER=sasiom1,
    SERVERNAME=SASApp - Stored Process Server"
    pass='mypassword'
    servertype=STOREDPROCESS;
    LIST COMMANDS;
QUIT;
```

Example 7: CONNECT and LIST Examples Using a Spawner

CONNECT and LIST Examples Using a Spawner

The following example demonstrates how you might use a combination of the LIST and CONNECT statements to connect to a SAS object spawner.

You can use LIST to obtain a list of the SAS servers that have been spawned on the local host:

```
PROC IOMOPERATE uri="iom://localhost:8581;Bridge;
                USER=mydomain\admin,PASS=*****" type=OBJECTSPAWNER;
    LIST SPAWNED SERVERS;
QUIT;
```

SAS returns output similar to the following:

```
Server 0
  Logical name      : SASApp - Logical Stored Process Server
  Server component  : SASApp - Stored Process Server
  Server class     : 15931E31-667F-11D5-8804-00C04F35AC8C
  Process owner    : admin@MYDOMAIN
  Server id        : E3D5D132-5DC0-46CB-A4FD-713098B1EE95
```

From the output, you can use the server ID to identify the server, and the TYPE= argument to specify that you want to connect to the object spawner:

```
PROC IOMOPERATE
  uri="iom://localhost:8581;Bridge;
  USER=mydomain\admin,PASS=*****"
  type=OBJECTSPAWNER
  spawned="E3D5D132-5DC0-46CB-A4FD-713098B1EE95";
LIST LOG;
LIST INFORMATION;
/* Other administration actions here */
QUIT;
```

Example 8: CONNECT Example Using Kerberos

CONNECT Example Using Kerberos

The following example demonstrates how you can use Kerberos to make a secure connection.

```
PROC IOMOPERATE;
  CONNECT host='myhost.example.com'
         port=4001;
QUIT;
Note: host - metadata host name, port - workspace port
/* Using security package and securitypackagelist for kerberos */
PROC IOMOPERATE;
  CONNECT host='myhost.example.com'
         port=4001
         IOMOPTS='SECURITYPACKAGE=Negotiate,SECURITYPACKAGELIST=Kerberos';
QUIT;
```

Example 9: Pausing an Object Spawner

Pausing an Object Spawner

The following command pauses the object spawner located on machine, myserver03.

```
PROC IOMOPERATE;
  CONNECT host='myserver03'
         port=8581
```

```

        user='sasadmin'
        pass='xxxxxxx'
        servertype=OBJECTSPAWNER;
    PAUSE SERVER;
QUIT;

```

Example 10: Continuing an Object Spawner

Continuing an Object Spawner

The following command continues the paused object spawner located on machine, myserver03.

```

PROC IOMOPERATE;
    CONNECT host='myserver03'
           port=8581
           user='sasadmin'
           pass='xxxxxxx'
           servertype=OBJECTSPAWNER;
    CONTINUE SERVER;
QUIT;

```

Example 11: Quiescing an Object Spawner

Quiescing an Object Spawner

The following command quiesces the object spawner located on machine, myserver03.

```

PROC IOMOPERATE;
    CONNECT host='myserver03'
           port=8581
           user='sasadmin'
           pass='xxxxxxx'
           servertype=OBJECTSPAWNER;
    QUIESCE SERVER;

```

Example 12: LIST TYPES Example

LIST TYPES Example

The following command:

```

PROC IOMOPERATE;
    LIST TYPES;
QUIT;

```

Returns output similar to the following:

```

1 PROC
1 !      IOMOPERATE;
2      LIST TYPES;

DataFlux Authentication Server
  Short type name : DataFluxAuth
  Class identifier : 2d1bcdbf-f900-4ca9-85f6-95ecdbaf2122

SAS Metadata Server
  Short type name : Metadata
  Class identifier : 0217e202-b560-11db-ad91-001083ff6836

SAS Object Spawner
  Short type name : ObjectSpawner
  Class identifier : 0e3b1810-6646-11d5-8863-00c04f48bc53

SAS OLAP Server
  Short type name : OLAP
  Class identifier : f3f46472-1e31-11d5-87c2-00c04f38f9f6

SAS Pooled Workspace Server
  Short type name : PooledWorkspace
  Class identifier : 620963ee-32bf-4128-bf5f-4b0df8ff90eb

SAS Stored Process Server
  Short type name : StoredProcess
  Class identifier : 15931e31-667f-11d5-8804-00c04f35ac8c

SAS Workspace Server
  Short type name : Workspace
  Class identifier : 440196d4-90f0-11d0-9f41-00a024bb830c
NOTE: The LIST TYPES command completed.
3 QUIT;

```

Example 13: Stopping a Metadata Server

Stopping a Metadata Server

The following command shuts down the metadata server named metadata01.

```

PROC IOMOPERATE;
  CONNECT host='metadata01'
          port=8561
          user='sasadmin'
          pass='xxxxxxx'
  STOP SERVER;
QUIT;

```

Example 14: DISCONNECT Example

DISCONNECT Example

The following example shows how using DISCONNECT enables you to embed several commands to multiple servers in one PROC IOMOPERATE statement:

```
proc iomoperate;

    connect uri="iom://myserver01.example.com:5555;Bridge";
    list defined servers;

    disconnect;

    connect uri="iom://myserver02.example.com:1111;Bridge";
    list attrs;

quit;
```

Example 15: FLUSH AUTHORIZATION CACHE Example

FLUSH AUTHORIZATION CACHE Example

The following example shows how to flush a server's authorization cache:

```
proc iomoperate uri="iom://myserver01.example.com:5555;Bridge";
USER=myuserid,PASS=mypassword";
flush authorization cache;
quit;
```

Example 16: REFRESH CONFIGURATION Example

REFRESH CONFIGURATION Example

The following example shows how to refresh a server's configuration:

```
proc iomoperate uri="iom://myserver01.example.com:8581;Bridge";
USER=myuserid,PASS=mypassword";
refresh configuration;
quit;
```

Example 17: RESET PERFORMANCE Example

RESET PERFORMANCE Example

The following example shows how to reset a server's performance metrics:

```
proc iomoperate uri="iom://myserver01.example.com:8581;Bridge;  
USER=myuserid,PASS=mypassword";  
reset performance;  
quit;
```

Example 18: SET Example

SET Example

The following example shows how to set the Audit.Authentication logger level to DEBUG.

```
proc iomoperate uri="iom://myserver01.example.com:8581;Bridge;  
USER=myuserid,PASS=mypassword";  
set attribute category="Loggers" name="Audit.Authentication" value="DEBUG";  
quit;
```

Part 5

Appendixes

<i>Appendix 1</i>	
Object Spawner and SAS OLAP Server Messages	185

Appendix 1

Object Spawner and SAS OLAP Server Messages

Object Spawner Messages	185
Load Balancing Error Messages	199

Object Spawner Messages

Here are messages that might be reported by the object spawner and explanations to correct their cause. You will find some of these messages only in the object spawner log. Some are error messages, and some are informational messages. For more information about changing the spawner logging level, see [“Configure Object Spawner Logging” on page 108](#). If you are unable to resolve any errors, contact SAS Technical Support.

- The object spawner is currently running in session <sessionID>.

Explanation:
The object spawner was started from a command line and is not able to launch processes because of Windows security constraints.

Resolution:
Run the spawner as a service.
- The launch of the server process failed with an unknown status code.

Explanation:
The process returned an unknown error code.

Resolution:
Contact SAS Technical Support.
- The launch of the server process failed due to an invalid or unaccessible SYSIN file.

Explanation:
The server was started with the SYSIN option, and the spawner either could not access or locate the SYSIN file.

Resolution:
Check the path to make sure that it is correct, or remove the SYSIN option.
- The launch of the server process failed due to an invalid or unaccessible AUTOEXEC file.

Explanation:
The autoexec file specified for the process is invalid or cannot be accessed.

Resolution:

Check that the autoexec file exists. Check the permissions on the file, and make sure that the server identity has access to the file.

- The launch of the server process failed due to an invalid or unaccessible LOG file.

Explanation:

The server was started with a -log option that contained a path or a file that could not be accessed by the server.

Resolution:

Check the path and make sure that it is correct and that the server identity has access to it.

- The launch of the server process failed due to a problem with the processing of the SAS logging facility configuration file (LOGCONFIGLOC).

Explanation:

The file specified for the -logconfigloc option cannot be processed. Either it is invalid, or it cannot be accessed.

Resolution:

Make sure that the server identity has access to this file, and that the filename is correct.

- The launch of the server process failed because the process ran out of memory.

Explanation:

The server has run out of memory during initialization.

Resolution:

If the server has been started with the -memsize option, either remove it, or choose a higher value.

- The launch of the server process failed because the SAS kernel could not be loaded.

Explanation:

For some reason, the SAS kernel cannot be loaded.

Resolution:

Contact SAS Technical Support.

- The launch of the server process failed due to an invalid or unaccessible PRINT file.

Explanation:

The server was started with a PRINT file that cannot be accessed.

Resolution:

Check the option to make sure that the file specified is accessible to the server identity.

- The launch of the server process failed because of a SAS kernel initialization failure.

Explanation:

For some reason, the SAS kernel cannot initialize.

Resolution:

Check the command line of the server to make sure that there are no invalid command line options. Also check any configuration files that are used to make sure that they are correct.

- The launch of the server process failed because of an invalid or unaccessible SASHELP library.

Explanation:

The SASHELP library cannot be accessed as configured.

Resolution:

Check the command line or configuration files where this library is specified and make sure that the library is accessible to the server identity.

- The launch of the server process failed because of an invalid or unaccessible SASUSER library.

Explanation:

The SASUSER library cannot be accessed as configured.

Resolution:

Check the command line or configuration files where this library is specified and make sure that the library is accessible to the server identity.

- The launch of the server process failed because of an invalid or unaccessible WORK library.

Explanation:

The WORK library cannot be accessed as configured.

Resolution:

Check the command line or config files where this library is specified and make sure that the library is accessible to the server identity.

- The launch of the server process failed because either SASUSER, SASHELP, or WORK was not defined.

Explanation:

The SASUSER, SASHELP, or WORK library was not defined.

Resolution:

Check the command line or configuration files and ensure that these libraries are defined.

- The launch of the server process failed because of the failure to load required formats.

Explanation:

SAS cannot load formats.

Resolution:

Check the configuration containing the format specification, or contact SAS Technical Support.

- The launch of the server process failed due to a site validation problem.

Explanation:

There is a problem with the server SETINIT.

Resolution:

Contact SAS Technical Support.

- The launch of the server process failed due to an invalid command line option.

Explanation:

There is an invalid command line option on the server.

Resolution:

Check the command and correct the option.

- The launch of the server process failed due to an I/O initialization failure.

Explanation:

A subsystem failed to initialize, preventing the process from handling input and output functions.

Resolution:

Contact SAS Technical Support.

- The launch of the server process failed due to the YZL subsystem failing to initialize.

Explanation:

The YZL subsystem failed to initialize.

Resolution:

Contact SAS Technical Support.

- The launch of the server process failed due to the YX subsystem failing to initialize.

Explanation:

The YX subsystem failed to initialize.

Resolution:

Contact SAS Technical Support.

- The launch of the server process failed because of an invalid or unaccessible CONFIG file.

Explanation:

The configuration file specified for the server is not accessible.

Resolution:

Make sure that the file exists and that it is accessible by the server identity.

- Spawned servers must allow for at least two threads. The specified value of **1** will be ignored, and a value of **2** will be used for THREADSMAX.

Explanation:

Servers that are spawned by the object spawner, including workspace, stored process, and pooled workspace servers must have at least two threads in order to function correctly. If the server starts with a threadsmax option of less than **2**, the value will be forced to be **2**.

Resolution:

Either remove the threadsmax option, or to set it to a value higher than **2**.

- The version of server *ServerName (MajorVersion.MinorVersion)* as specified in metadata does not match the current running version (*MajorVersion.MinorVersion*). Some features might not be available.

Explanation:

There is a mismatch between the compiled version of the software and the major and minor values stored in the **ServerName** component.

Resolution:

This message does not indicate a problem, but is meant to inform you of this version mismatch. To resolve this message, update your SAS software with the latest version.

- The server *ServerName (ServerOMR-ID)* cannot be started with the **-encryptfips** option, because this version (*MajorVersion.MinorVersion*) does not support this option.

Explanation:

The specified server is an earlier version of SAS and does not support the **-encryptfips** option. The server will start, but **-encryptfips** is ignored. The object spawner still requires that the server use AES.

Resolution:

Update your SAS software with the latest version.

- The configuration source value conflicts with the previously specified configuration source value.

Explanation:

Multiple configuration files have been specified for the object spawner.

Resolution:

Check your configuration and ensure that only a single configuration option is used for your object spawner.

- Unable to retrieve the name definition or definitions.

Explanation:

The object or objects specified could not be found.

Resolution:

Check your configuration and command line and try again.

- The server named name is not unique to the listen port. Therefore only the first occurrence of this server will be used.

Explanation:

Two or more servers have been defined to use the same listen port. The object spawner is unable to distinguish between these two servers that are defined on the same port. Therefore, only the first server defined on this port will be used by the object spawner.

Resolution:

Check your configuration and change the port for the server listed in this message.

- Objspawn was unable to initiate the listen on port.

Explanation:

The object spawner could not listen on the specified port. In most cases this means that the port is already in use.

Resolution:

Check to see that no other application is currently using the specified port. Alternatively, update the port that the object spawner is trying to use to a port that is not currently being used.

- Port is reserved for objspawn administration and cannot be used as a server listen port.

Explanation:

The object spawner cannot use the same port for administration, also known as the operator port, and as a server listen port.

Resolution:

Check your configuration and ensure that the operator port for the object spawner and the ports for all servers are different.

- Port is reserved for objspawn to launched server communication. Therefore this server definition will not be included.

Explanation:

The specified port is being used by the object spawner for internal communication and cannot be used as a server listen port.

Resolution:

Change the specified port for this server to a different port.

- Port is reserved for UUID Generator requests. Therefore this server definition will not be included.

Explanation:

The specified port is being used by the object spawner to fulfill UUID generator requests. It cannot be used as a server listen port.

Resolution:

Update the port for the server or the UUID generator so that these services are defined on different ports.

- No configuration was specified.

Explanation:

No configuration information can be found for the object spawner.

Resolution:

Check the command line and make sure that some configuration option was specified for the object spawner.

- The SAS Metadata Server configuration file failed to process.

Explanation:

There is a problem in the object spawner's configuration file

Resolution:

Check the configuration file to make sure it is correct. Contact Tech Support for assistance.

- The metadata returned from the SAS Metadata Server was empty and failed to process. This could be due to there being no spawner definition in the metadata for this machine. Or, if the spawner name was specified, the name specified did not match any spawner definitions in the metadata. A spawner definition for this machine did not exist in the processed configuration. Objspawn cannot continue without a spawner definition. The name spawner definition did not exist in the processed configuration. Objspawn cannot continue without a spawner definition.

Explanation:

No object spawner definition can be found in the metadata.

Resolution:

Ensure that an object spawner definition exists in the metadata. If command `-sasspawnercn` or `-sscoption` is used on the command line, ensure that an object spawner definition of that name exists in the metadata.

- An unknown option name was specified

Explanation:

The specified option is unknown to the object spawner.

Resolution:

Check your command line or configuration for this option and remove it. Contact Tech Support for assistance.

- The attribute name was specified more than once. The previously specified value will be used.

Explanation:

The attribute or option was found multiple times in the configuration source. The first instance will be used.

Resolution:

Check your configuration for this attribute and only specify it once.

- The value (value) specified for the attribute name is invalid. The value must be corrected for Objspawn to initialize properly.

Explanation:

The attribute has an invalid value.

Resolution:

Correct the attribute's value and restart the object spawner.

- Objspawn encountered errors during results processing.

Explanation:

Errors occurred during the object spawner's initialization.

Resolution:

Check the object spawner's log for information about what caused the problem.
Correct the problem and restart the object spawner.

- Objspawn cannot proceed with the specified Telnet administrator password. The password must be corrected for Objspawn to initialize properly. Objspawn does not allow the administrator password to be all asterisks. The password must be corrected for Objspawn to initialize properly. Objspawn does not allow the administrator password to be all asterisks. If the password is not recorded as all asterisks in the SAS Metadata Server, then the identity Objspawn used to connect to the SAS Metadata Server does not have permission to view the password value.

Explanation:

The object spawner found an operator or administrator password in its configuration that it cannot use.

Resolution:

Check the password that has been specified for the object spawner Telnet access. If this object spawner is defined in metadata, this password belongs to the user that is specified on the **Initialization** tab of the object spawner definition. Contact Tech Support for assistance.

- The credentials specified for the name (ID) server definition failed to authenticate. Therefore this server definition will not be included. A failure occurred during the retrieval of the authenticated credentials for the name (ID) server definition.

Explanation:

The specified credentials for the specified server cannot be authenticated. In order to avoid further errors and a possible account lockout, the object spawner will not use this definition.

Resolution:

Check the credentials that are specified for this server, and make sure that they are correct. Ensure that the user name is a valid user name for the system that you are using and that the password is valid for that user. After correcting the problem, restart your object spawner.

- The name (ID) server definition is not defined for this machine and will be ignored.

Explanation:

The server definition specified is not defined for the machine that the object spawner is currently running on. This server will not be used by the object spawner.

Resolution:

If the specified server is needed on the machine that this object spawner is running on, make sure that the server definition has this machine's name as part of its definition.

- Objspawn was unable to locate a server definition. Objspawn is exiting.

Explanation:

No valid server definition can be found by the object spawner. This might mean that some servers have configuration issues, or that the object spawner does not have any servers configured to be used.

Resolution:

Check the object spawner log and look for errors or warnings pertaining to servers that might be defined. Correct these issues and restart your object spawner. If no errors or warnings are found, make sure that your object spawner has servers associated with it.

- The cluster name (ID) is configured to use server side pooling. However, no port bank exists in this spawner to use with server side pooling.

Explanation:

The specified cluster contains a Pooled Workspace Server. For a Pooled Workspace Server to work correctly, the object spawners that are associated with the Pooled Workspace Server must contain a set of ports known as a port bank.

Resolution:

Check your object spawner definition and go to the Operator connection. From the **Options** tab, you will be able to add ports to the object spawner's port bank. Save your changes and restart your object spawner.

- Objspawn is unable to contact the metadata server.

Explanation:

The object spawner cannot connect to the specified metadata server in the configuration file or on the command line.

Resolution:

Make sure that the configuration file or the command line points to the correct metadata server. Also verify that the metadata server is running.

- The cluster definition name (ID) is invalid. Therefore this cluster definition will not be included.

Explanation:

Errors were found during the processing of the specified cluster.

Resolution:

Check the object spawner log for other errors related to the specified cluster. Correct these errors and restart your object spawner.

- The configuration for this spawner is invalid. Please check the configuration and try again.

Explanation:

Errors were found during the processing of the object spawner's configuration.

Resolution:

Check the object spawner log for other errors related to the spawner's configuration processing. Correct these errors and restart your object spawner.

- The machine name cannot be found. This machine will be ignored.

Explanation:

The specified machine name cannot be found. The machine might not exist, or the machine name might not be entered correctly.

Resolution:

Check the name of the machine to ensure that it is correct.

- Objspawn does not allow Telnet administrator access when the administrator password is the ASCII character specified. The Telnet administrator access will be disabled. Objspawn does not allow Telnet administrator access when the administrator password begins with the ASCII string "SASI". The Telnet administrator access will be disabled.

Explanation:

The specified password for Telnet administrative access cannot be used. The Telnet administrative interface will be disabled.

Resolution:

Change the administrative password of the object spawner, and restart the object spawner to enable Telnet access.

- The name (ID) server definition did not specify a command to use. Therefore this server definition will not be included.

Explanation:

No command was specified for the object spawner to use for this server. The server will not be used by the object spawner.

Resolution:

Specify a command for the object spawner.

- The name (ID) server definition did not contain a port or TCP/IP service name. The default port for this server class identifier will be used.

Explanation:

The specified server does not have a port or service defined. The default port for this type of server will be used.

Resolution:

Each type of server has its own default port. If a specific port is needed for this server, then specify one in the configuration.

- The name TCP/IP service name specified in the name server definition resolves to a port that differs from port specified in the name server attribute definition. Therefore this server definition will not be included.

Explanation:

There is a conflict between the port that is defined for the server and the service that is defined for the server.

Resolution:

Correct the conflict by either specifying only port or service, or by ensuring that the service resolves to the same port as defined by the port attribute.

- The name TCP/IP service name specified in the name server definition is invalid. Therefore this server definition will not be included.

Explanation:

The specified service name cannot be resolved.

Resolution:

Correct or remove the service name specification and restart your object spawner.

- The launched server definition was not found for the server with activity. The server might have completed its processing and exited. The requested server was not found. The server might have completed its processing and exited.

Explanation:

The object spawner has attempted to access a server that has already shut down.

Resolution:

None. This error might occur during the object spawner's server management. It is handled internally by the object spawner.

- Objspawn was unable to perform the requested action due to an active configuration refresh.

Explanation:

A request was made to the object spawner while it was attempting to refresh its configuration. Once the refresh is complete, requests will be allowed to be performed.

Resolution:

Once the refresh has completed, resubmit the request that received this error.

- Objspawn was unable to remove its existing configuration. The status received was status. Objspawn is terminating.

Explanation:

The object spawner was attempting to refresh itself when an error occurred. The object spawner will shut down.

Resolution:

Check the object spawner log to determine the error that caused the object spawner to shut down. It is very likely that any error that prevented the refresh from succeeding will also prevent the object spawner from restarting.

- Objspawn has lost connectivity with the server (child ID) launched on behalf of user ID.

Explanation:

A server that the object spawner has started has shut down unexpectedly.

Resolution:

This might happen sometimes due to administrative requests that are made against the servers that the object spawner has started. If these errors persist, then contact Tech Support for assistance.

- The launch of server name for user ID failed. The server failed to start. Failed to start the server.

Explanation:

The specified server failed to start. Other messages in the log might point to the cause of this failure. The typical reason for these failures is a permissions issue that is related to the server command line.

Resolution:

Depending on what operating system you are using, you might be able to find extended error information. On Windows systems, consult the Windows Event log. On z/OS systems, check the system log. On UNIX or Linux systems,

messages might be written to the console, or they might exist in the `objspawn_console.log`. Correct the errors that are reported, and restart your object spawner if necessary.

- The peer for launched server child ID is no longer defined. The launched server will exit.

Explanation:

The peer (client) that requested the server is no longer present, and might have disconnected unexpectedly. The server that was started for this client will exit.

Resolution:

No resolution. If the message persists, there might be a problem with the client application. Contact Tech Support for assistance.

- The server definition associated with the launched server no longer exists. This might be due to an objspawn configuration refresh.

Explanation:

The object spawner is in the process of refreshing itself when it receives a request or activity for a server that it is in the process of refreshing.

Resolution:

Once the refresh has completed, the message should no longer be seen.

- The operator password specified by name is invalid.

Explanation:

An invalid password was specified to the Telnet administrative interface.

Resolution:

Use the password as defined in metadata.

- Objspawn was unable to verify that the password due to an active configuration refresh.

Explanation:

The object spawner was in the process of refreshing its configuration when a client attempted a connection to the Telnet administrative interface.

Resolution:

Once the refresh has completed, the Telnet interface will be available again.

- Objspawn was unable to generate UUIDs due to an active configuration refresh.

Explanation:

The object spawner was in the process of refreshing its configuration when a client attempted to request a UUID.

Resolution:

Once the refresh has completed, UUID generation will be available again.

- Client name does not have permission to use server name (ID).

Explanation:

The client that has connected does not have permission to use the specified server.

Resolution:

Check with your system administrator. A client must have the proper permissions on the server in order to use the server.

- Objspawn is unable to find server named name. Objspawn was unable to find the specified server name.

Explanation:

A client requested a server by name from the object spawner. The requested server is not a server that the object spawner is currently configured to use.

Resolution:

Make sure that the client application is connecting to the correct machine and is using the correct server. Also make sure that the object spawner configuration has the correct servers associated with it.

- The server requested is not configured correctly. The client request cannot be completed.

Explanation:

There was a problem with the server that the client requested, and it cannot be used by the object spawner. Check the object spawner log for errors or warnings that indicate the issue with the server.

Resolution:

Correct the server configuration and restart the object spawner. Otherwise, select a different server to use.

- There is a problem with the Negotiate Package List. Please verify that the property is set correctly. There is a problem with the Security Package. Please verify that the property is set correctly. There is a problem with the Service Principal Name. Please verify that the property is set correctly.

Explanation:

There is a problem with one of the Integrated Windows Authentication properties. These properties have certain values. Check the applicable documentation for information about these properties.

Resolution:

Correct the problems with these security properties, and restart your object spawner.

- Credentials could not be generated for user ID because spawner is not connecting to the metadata server as a trusted user. This might prevent the launched server from contacting the metadata server.

Explanation:

Servers that are launched by the object spawner attempt to connect to the metadata server in order to get further configuration information, including pre-assigned libraries and security information. In order to facilitate this, the object spawner generates credentials for the server to use to connect to the metadata server. If the object spawner is not connected to the metadata server as a trusted user, then it cannot generate credentials for other users.

Resolution:

Check the object spawner's configuration file or command line. Ensure that the user that is specified there is a trusted user. Check the security documentation for more information.

- No host credentials exist to start this server. Either the client needs to send in host credentials, or credentials need to be specified for the server. No host credentials exist to start server name (ID). Either the client needs to send in host credentials, or credentials need to be specified for the server.

Explanation:

The specified server has been configured in metadata to allow SAS Token Authentication. These servers must have credentials specified for them in metadata in order for the object spawner to use to launch them.

Resolution:

Ensure that credentials are set for the server on the Options tab. After setting the credentials, restart your object spawner.

- The spawner cannot accept a new client because it is currently in a paused state. The spawner cannot accept a new client because it is currently in a deferred shutdown state.

Explanation:

The object spawner is current in the specified state and cannot currently accept new clients.

Resolution:

If the object spawner is in a deferred shutdown state, the object spawner will soon shut down. At that time it will no longer accept clients, and must be restarted. A paused object spawner can be continued at any time. Once it is continued or resumed, it will accept clients again.

- A port could not be obtained to allow the client to be connected to this server.

Explanation:

This applies to Pooled Workspace Servers. When a client is redirected to a workspace server from a pool, the workspace server opens a port briefly for the client to use in order to connect to the server. This message indicates that at the time that a client request to the pool was made, no ports were available to have the server listen on.

Resolution:

Add more ports to the object spawner port bank.

- Objspawn encountered a failure while processing a Telnet request. The peer connection will be closed.

Explanation:

An error occurred during the processing of an administrative interface request. The interface will be closed.

Resolution:

Attempt to reconnect to the administrative interface. If the message continues, restart the object spawner, or contact Tech Support.

- Objspawn encountered a failure while processing a UUID generator request. The peer connection will be closed.

Explanation:

An error occurred during the generation of a UUID for the client.

Resolution:

Check the object spawner log for warnings or messages that can indicate the problem. Restart the object spawner if necessary. If the problem continues, contact Tech Support.

- Line line_number in the configuration file is not in LDIF format and is being skipped.

Explanation:

The specified line in the configuration file is not in the correct format and cannot be used.

Resolution:

Correct the specified line.

- The connection to the LDAP server failed.

Explanation:

The specified server in the LDAP configuration could not be accessed. The object spawner cannot obtain its configuration.

Resolution:

Check the specified LDAP server and ensure that it is running.

- The attribute_name attribute found on line line_number failed to process.

Explanation:

The specified attribute name is incorrect.

Resolution:

Correct the attribute and restart your object spawner.

- Two consecutive blank lines in an LDIF configuration file identify the logical end of the configuration. The remainder of the LDIF configuration file will be ignored.

Explanation:

The LDIF file contains two consecutive blank lines. This signifies the end of the configuration file.

Resolution:

If the rest of the file is needed, then remove one of the blank lines.

- The object class definition name contains more than one distinguished name definition. The object class definition will be ignored.

Explanation:

The specified definition contains multiple distinguished names. A definition can contain only one distinguished name.

Resolution:

Remove one of the distinguished names.

- The object class definition name contains conflicting object class identifier definitions. The object class definition will be ignored.

Explanation:

The specified definition contains multiple or incompatible class identifier definitions.

Resolution:

Correct the object class identifier in the specified definition.

- The object class definition that began on line line_number defines an unknown object class and is being ignored.

Explanation:

An unknown class identifier was used in the specified definition and must be corrected.

Resolution:

Correct the specified definition to have a known class identifier.

- The object class definition that began on line `line_number` contained one or more attribute resolution failures and is being ignored.

Explanation:

The specified object class definition contained one or more errors and will be ignored by the object spawner.

Resolution:

See the object spawner log and correct the errors reported. Restart the object spawner.

- Objspawn was unable to open the configuration file (name).

Explanation:

The configuration file that is specified to the object spawner cannot be located or opened.

Resolution:

Specify the correct configuration file.

- The name server definition did not specify a protocol. Therefore this server definition will not be included.

Explanation:

The server definition is missing the protocol attribute. The protocol must be set.

Resolution:

Correct the configuration by setting the protocol attribute and restart your object spawner.

Load Balancing Error Messages

Here are error messages that might be reported by the object spawner (or the OLAP server when load balanced) and explanations to correct their cause. You will find some of these messages only in the object spawner or OLAP server logs. For more information about changing the spawner logging level, see [“Configure Object Spawner Logging” on page 108](#) . For more information about the OLAP server, see [“Administering SAS OLAP Servers” on page 126](#) . If you are unable to resolve the error, contact SAS Technical Support.

- Objspawn was unable to launch the server `server_name` due to the server launch exceeding the specified wait time. The server did not start in the specified amount of time.

Explanation:

The specified server took longer than the configured Launch Timeout setting to start completely.

Resolution:

This message might appear if the system is in heavy use and cannot start a server in an efficient manner. You might need to increase the value that is specified in the Launch Timeout to allow sufficient time for your servers to start. You might want to also investigate the amount of work that this system is currently being sent and determine whether it is overloaded.

- Port is reserved for server name and cannot be used as a load balanced server instance listen port. Therefore this server definition will not be included. Port is reserved for a load balanced server instance of server name. Therefore this server definition will not be included.

Explanation:

There is a port conflict between a load-balanced server and a server that is not load-balanced. These different types of servers cannot use the same port.

Resolution:

Change one of the ports of the server listed so that they are different. Restart your object spawner.

- Load balanced server listen port is in use. This might cause future load balanced server launches to fail.

Explanation:

Some other application is already using the port that is specified for the server to use. Future failures might occur because servers of this kind that are launched for load balancing might fail.

Resolution:

Select a different port for the server in question.

- The credentials specified for the name (ID) cluster definition failed to authenticate. A failure occurred during the retrieval of the authenticated credentials for the name cluster definition.

Explanation:

Load balancing is unable to obtain or authenticate the specified credentials for the load-balanced logical server or cluster listed in the message. This will prevent load balancing from sharing the load among the other servers that are defined.

Resolution:

Verify that the correct credentials have been defined for the cluster or logical server. Credentials are set in the logical server on the **Load Balancing** tab of the Logical Server's property page.

- The name (ID) cluster does not contain any valid server definitions. Therefore this cluster definition will not be included. No valid cluster could be found for this configuration. Check your configuration and try again. No valid server definitions found for cluster name (ID). No valid server definitions were found in cluster name (ID).

Explanation:

One or more errors were found when trying to process the cluster and its server definitions. The object spawner will not monitor this cluster.

Resolution:

View the object spawner log to find other errors or warnings that have caused the cluster and servers to fail to process. Correct these issues and restart your object spawner.

- The name (ID) server definition is a load balanced server that requires credentials. No credentials were associated with the definition. Therefore this server definition will not be included.

Explanation:

Certain load-balancing servers require credentials in order to be launched by load balancing. The server that is specified here requires credentials, but none were found.

Resolution:

Specify credentials on the **Options** tab of the property page of the listed server definition. Restart your object spawner.

- There is no available server for this client to use. The Balance algorithm timed out before a server could be found. Unable to find an available server to redirect client.

Explanation:

There are no available servers at this time. All servers in the cluster are busy with clients and cannot handle any new clients at this time.

Resolution:

The client can try again later when the servers are less busy. The system administrator might want to determine whether the settings need to be adjusted to allow more clients in. Also, more servers might need to be added to handle the client load.

- More servers were requested to be started than were actually defined. Overflow requests will be ignored.

Explanation:

The Start Size property for this specified server a number of servers larger than the number allowed by the configuration.

Resolution:

The Start Size property is found in the Advanced Options of the **Options** tab of the server definition. This message is just a warning that only the number of configured servers will be started, and not more as specified by the Start Size property.

- An error occurred while server (server_name) was starting. Now attempting a different server. Unable to start the server name (ID).

Explanation:

The specified server failed to start for load balancing. Load balancing will now attempt to use a different server.

Resolution:

Check the object spawner log to determine whether any messages indicate why the server failed to start. A problem with the server or its command line might need to be corrected. Also, the system might have become too busy or overloaded to handle starting a new server at this time.

- Attempt to connect to peer failed. Attempt to connect to peer (name) for cluster name (ID) failed with exception message.

Explanation:

Load balancing was configured to use several machines to balance the client load, but was unable to connect to one of the other peers as defined in the cluster. Load Balancing will not be able to use that peer at this time. A peer can be another object spawner.

Resolution:

This message might be seen during initialization due to the fact the other peers or object spawners have not initialized yet. Once they initialize, load balancing will be able to communicate with them. Ensure that the other peers have initialized.

- The Max Clients property for this server is set to 0. No clients will be redirected to this server definition.

Explanation:

The server has been configured to allow a maximum of zero clients. Therefore, no clients will be sent to this server.

Resolution:

If the intent is to not allow clients in to this server, then there is not resolution. However, if this is not the intent, the Max Clients property can be updated in the Advanced Options dialog box on the **Options** Tab of the server's property page.

- The name (ID) cluster definition requires the use of multibrige connections and none were found. Therefore this cluster definition will not be included.

Explanation:

The specified cluster contains servers that require the use of a multibrige connection. None were defined for the servers in this cluster.

Resolution:

Correct this issue by adding multibrige connections to the servers that are members of the specified cluster. Multibrige connections can be added on the **Options** tab of the server's Bridge connection property page.

- The password obtained from the metadata indicates that it was obtained by a user that has unrestricted access. Please check your configuration.

Explanation:

The connection to the metadata server has been made by an unrestricted user. An unrestricted user cannot connect to the metadata server for load balancing, because this prevents load balancing from getting some necessary information.

Resolution:

Check your configuration file or command line. Ensure that the user is not an unrestricted user.

- The host name resolved to an address that matches a host name already found in this definition (name). Therefore, this host name will be skipped.

Explanation:

The specified host name has resolved to an address that the load balancer has already processed. To avoid potential confusion and conflicts, the load balancer will treat these two host names as the same machine.

Resolution:

Check and ensure that the machine names are configured correctly.

- The cluster name (ID) does not have a valid user name associated with it. Load balancing does not support anonymous connections. Specify credentials for you cluster and retry.

Explanation:

Load balancing has been configured to use multiple servers on multiple hosts in this particular cluster. To do so, load balancing must connect to the other hosts. However, no credentials were configured to make this connection with.

Resolution:

Specify credentials in the cluster definition. Credentials can be specified on the **Load Balancing** tab of the cluster's property page.

Glossary

access control entry

a set of identities and permissions that are directly associated with a particular resource. Each access control entry is directly associated with only one resource. More than one ACE can be associated with each resource. Short form: ACE.

ACE

See access control entry.

aggregation

a summary of detail data that is stored with or referred to by a cube.

API

See application programming interface.

application programming interface

a set of software functions that facilitate communication between applications and other kinds of programs or services. Short form: API.

Application Response Measurement

the name of an application programming interface that was developed by an industry partnership and which is used to monitor the availability and performance of software applications. ARM monitors the application tasks that are important to a particular business. Short form: ARM.

ARM

See Application Response Measurement.

authentication

See client authentication.

authentication domain

a SAS internal category that pairs logins with the servers for which they are valid. For example, an Oracle server and the SAS copies of Oracle credentials might all be classified as belonging to an OracleAuth authentication domain.

client authentication

the process of verifying the identity of a person or process for security purposes.

client-side pooling

a configuration in which the client application maintains a collection of reusable workspace server processes.

cube

See OLAP cube.

cube loading

the process of building a logical set of data that is organized and structured in a hierarchical, multidimensional arrangement.

daemon

a process that starts and waits either for a request to perform work or for an occurrence of a particular event. After the daemon receives the request or detects the occurrence, it performs the appropriate action. If nothing else is in its queue, the daemon then returns to its wait state.

dimension

a data element that categorizes values in a data set into non-overlapping categories that can be used to group, filter, and label the data in meaningful ways. Hierarchies within a dimension typically represent different groupings of information that pertains to a single concept. For example, a Time dimension might consist of two hierarchies: (1) Year, Month, and Date, and (2) Year, Week, and Day.

DNS name

a name that is meaningful to people and that corresponds to the numeric TCP/IP address of a computer on the Internet. For example, www.alphaliteairways.com might be the DNS name for an Alphalite Airways Web server whose TCP/IP address is 192.168.145.6.

drill-through table

a view, data set, or other data file that contains data that is used to define a cube. Drill-through tables can be used by client applications to provide a view from processed data into the underlying data source.

encryption

the act or process of converting data to a form that is unintelligible except to the intended recipients.

foundation services

See SAS Foundation Services.

grid

a collection of networked computers that are coordinated to provide load balancing of multiple SAS jobs, scheduling of SAS workflows, and accelerated processing of parallel jobs.

grid control server

the machine on a grid that distributes SAS programs or jobs to the grid nodes. The grid control server can also execute programs or jobs that are sent to the grid.

grid monitoring server

a metadata object that stores the information necessary for the Grid Manager plug-in in SAS Management Console to connect with the Platform Suite for SAS or other grid middleware to allow monitoring and management of the grid.

grid node

a machine that is capable of receiving and executing work that is distributed to a grid.

hierarchy

an arrangement of related objects into levels that are based on parent-child relationships. Members of a hierarchy are arranged from more general to more specific.

Integrated Object Model

the set of distributed object interfaces that make SAS software features available to client applications when SAS is executed as an object server. Short form: IOM.

Integrated Object Model server

See IOM server.

IOM

See Integrated Object Model.

IOM bridge

a software component of SAS Integration Technologies that enables Java clients and Windows clients to access an IOM server.

IOM server

a SAS object server that is launched in order to fulfill client requests for IOM services. Short form: IOM server.

level

an element of a dimension hierarchy. Levels describe the dimension from the highest (most summarized) level to the lowest (most detailed) level. For example, possible levels for a Geography dimension are Country, Region, State or Province, and City.

load balancing

for IOM bridge connections, a program that runs in the object spawner and that uses an algorithm to distribute work across object server processes on the same or separate machines in a cluster.

logical grid server

a metadata object that stores the command that is used by a grid-enabled SAS program to start a SAS session on a grid.

logical server

the second-level object in the metadata for SAS servers. A logical server specifies one or more of a particular type of server component, such as one or more SAS Workspace Servers.

MDX language

See multidimensional expressions language.

Measures dimension

a special dimension that contains summarized numeric data values (measures) that are analyzed. Total Sales and Average Revenue are examples of measures. For example, you might drill down within the Clothing hierarchy of the Product dimension to see the value of the Total Sales measure for the Shirts member.

member

an element of a dimension. For example, for a dimension that contains time periods, each time period is a member of the dimension.

metadata object

a set of attributes that describe a table, a server, a user, or another resource on a network. The specific attributes that a metadata object includes vary depending on which metadata model is being used.

MultiBridge connection

a specialized bridge connection that is used for stored process servers. Each MultiBridge connection represents a separate server process and runs on a specific port.

multidimensional expressions language

a standardized, high-level language that is used to query multidimensional data sources. The MDX language is the multidimensional equivalent of SQL (Structured Query Language). Short form: MDX language.

NWAY aggregation

the aggregation that has the minimum set of dimension levels that is required for answering any business question. The NWAY aggregation is the aggregation that has the finest granularity.

object spawner

a program that instantiates object servers that are using an IOM bridge connection. The object spawner listens for incoming client requests for IOM services. When the spawner receives a request from a new client, it launches an instance of an IOM server to fulfill the request. Depending on which incoming TCP/IP port the request was made on, the spawner either invokes the administrator interface or processes a request for a UUID (Universal Unique Identifier).

OLAP

See online analytical processing.

OLAP cube

a logical set of data that is organized and structured in a hierarchical, multidimensional arrangement to enable quick analysis of data. A cube includes measures, and it can have numerous dimensions and levels of data.

OLAP schema

a container for OLAP cubes. A cube is assigned to an OLAP schema when it is created, and an OLAP schema is assigned to a SAS OLAP Server when the server is defined in the metadata. A SAS OLAP Server can access only the cubes that are in its assigned OLAP schema.

online analytical processing

a software technology that enables users to dynamically analyze data that is stored in multidimensional database tables (cubes).

pool

a group of server connections that can be shared and reused by multiple client applications. A client-side pool consists of one or more puddles.

puddle

a group of servers that are started and run using the same login credentials. Each puddle can also allow a group of clients to access the servers.

Remote Library Services

a feature of SAS/SHARE and SAS/CONNECT software that enables you to read, write, and update remote data as if it were stored on the client. RLS can be used to access SAS data sets on computers that have different architectures. RLS also provides read-only access to some types of SAS catalog entries on computers that have different architectures. Short form: RLS.

RLS

See Remote Library Services.

SAS Application Server

a logical entity that represents the SAS server tier, which in turn comprises servers that execute code for particular tasks and metadata objects.

SAS ARM interface

an interface that can be used to monitor the performance of SAS applications. In the SAS ARM interface, the ARM API is implemented as an ARM agent. In addition, SAS supplies ARM macros, which generate calls to the ARM API function calls, and ARM system options, which enable you to manage the ARM environment and to log internal SAS processing transactions.

SAS batch server

in general, a SAS application server that is running in batch mode. In the SAS Open Metadata Architecture, the metadata for a SAS batch server specifies the network address of a SAS Workspace Server, as well as a SAS start command that will run jobs in batch mode on the SAS Workspace Server.

SAS Deployment Wizard

a cross-platform utility that installs and initially configures many SAS products. Using a SAS installation data file and, when appropriate, a deployment plan for its initial input, the wizard prompts the customer for other necessary input at the start of the session, so that there is no need to monitor the entire deployment.

SAS Foundation Services

a set of core infrastructure services that programmers can use in developing distributed applications that are integrated with the SAS platform. These services provide basic underlying functions that are common to many applications. These functions include making client connections to SAS application servers, dynamic service discovery, user authentication, profile management, session context management, metadata and content repository access, activity logging, event management, information publishing, and stored process execution.

SAS IOM workspace

in the IOM object hierarchy for a SAS Workspace Server, an object that represents a single session in SAS.

SAS Management Console

a Java application that provides a single user interface for performing SAS administrative tasks.

SAS Metadata Repository

a container for metadata that is managed by the SAS Metadata Server.

SAS Metadata Server

a multi-user server that enables users to read metadata from or write metadata to one or more SAS Metadata Repositories.

SAS OLAP Cube Studio

a Java interface for defining and building OLAP cubes in SAS System 9 or later. Its main feature is the Cube Designer wizard, which guides you through the process of registering and creating cubes.

SAS OLAP Server

a SAS server that provides access to multidimensional data. The data is queried using the multidimensional expressions (MDX) language.

SAS Open Metadata Architecture

a general-purpose metadata management facility that provides metadata services to SAS applications. The SAS Open Metadata Architecture enables applications to exchange metadata, which makes it easier for these applications to work together.

SAS Pooled Workspace Server

a SAS Workspace Server that is configured to use server-side pooling. In this configuration, the SAS object spawner maintains a collection of workspace server processes that are available for clients.

SAS Stored Process Server

a SAS IOM server that is launched in order to fulfill client requests for SAS Stored Processes.

SAS token authentication

a process in which the metadata server generates and verifies SAS identity tokens to provide single sign-on to other SAS servers. Each token is a single-use, proprietary software representation of an identity.

SAS Workspace Server

a SAS IOM server that is launched in order to fulfill client requests for IOM workspaces.

SAS/CONNECT server

a server that provides SAS/CONNECT services to a client. When SAS Data Integration Studio generates code for a job, it uses SAS/CONNECT software to submit code to remote computers. SAS Data Integration Studio can also use SAS/CONNECT software for interactive access to remote libraries.

SAS/CONNECT spawner

a program that runs on a remote computer and that listens for SAS/CONNECT client requests for connection to the remote computer. When the spawner program receives a request, it invokes a SAS session on the remote computer.

Scalable Performance Data Engine

a SAS engine that is able to deliver data to applications rapidly because it organizes the data into a streamlined file format. Short form: SPD Engine.

schema

a map or model of the overall data structure of a database. A schema consists of schema records that are organized in a hierarchical tree structure. Schema records contain schema items.

server component

in SAS Management Console, a metadata object that specifies information about how to connect to a particular kind of SAS server on a particular computer.

server context

a SAS IOM server concept that describes how SAS Application Servers manage client requests. A SAS Application Server has an awareness (or context) of how it is being used and makes decisions based on that awareness. For example, when a SAS Data Integration Studio client submits code to its SAS Application Server, the server determines what type of code is submitted and directs it to the correct physical server for processing (in this case, a SAS Workspace Server).

server-side pooling

a configuration in which a SAS object spawner maintains a collection of reusable workspace server processes that are available for clients. The usage of servers in this pool is governed by the authorization rules that are set on the servers in the SAS metadata.

service

one or more application components that an authorized user or application can call at any time to provide results that conform to a published specification. For example, network services transmit data or provide conversion of data in a network, database services provide for the storage and retrieval of data in a database, and Web services interact with each other on the World Wide Web.

single sign-on

an authentication model that enables users to access a variety of computing resources without being repeatedly prompted for their user IDs and passwords. For example, single sign-on can enable a user to access SAS servers that run on different platforms without interactively providing the user's ID and password for each platform. Single sign-on can also enable someone who is using one application to launch other applications based on the authentication that was performed when the user initially logged on.

spawner

See object spawner and SAS/CONNECT spawner.

SPD Engine

See Scalable Performance Data Engine.

SSO

See single sign-on.

star schema

tables in a database in which a single fact table is connected to multiple dimension tables. This is visually represented in a star pattern. SAS OLAP cubes can be created from a star schema.

thread

a single path of execution of a process that runs on a core on a CPU.

threaded I/O

I/O that is performed by multiple threads in order to increase its speed. In order for threaded I/O to improve performance significantly, the application that is performing the I/O must be capable of processing the data rapidly as well.

threading

a high-performance technology for either data processing or data I/O in which a task is divided into threads that are executed concurrently on multiple cores on one or more CPUs.

wizard

an interactive utility program that consists of a series of dialog boxes, windows, or pages. Users supply information in each dialog box, window, or page, and the wizard uses that information to perform a task.

workspace

See SAS IOM workspace.

Index

Special Characters

.bat file 43

A

abnormal OLAP server shutdowns 103

accessibility features 4

application server 3

See also SAS Application Server

server components 5, 6

ARM log files 39

authentication

IOM server 148

security authentication packages 150

B

batch servers 4, 7, 23

SAS DATA Step Batch Server 23, 24

SAS Generic Batch Server 24

SAS Java Batch Server 23, 25

build files

for OLAP cubes, on SAS Workspace Servers 102

C

catalogs

for servers on multiple machines 93

cleaning up temporary files

after abnormal OLAP server shutdowns 103

client assignment (load balancing)

to OLAP servers 35

to workspace and stored process servers 34

client-side pooling 55

application properties 67

assigning libraries to server 76

choosing 61

concepts 62

configuration overview 62

configuring 69

configuring across multiple machines 69

configuring server properties 65

configuring workspace server for row-level security 77

connection process 58

converting logical workspace server to 64

creating restricted workspace server 74

creating Web Report Studio deployment 77

how it works 57

metadata identities 63

metadata identities and logins for puddle access 63

puddles for logical pooled server 63

Restricted Client-side Pool

Administrator 73

Restricted Pool Puddle Login group 73

sensitive data resources 77

verifying, for SAS Web Report Studio 69

coalescing cubes 131

compute services 16

Cost algorithm 44, 45

parameters 46

SAS Stored Process Server example 48

SAS Workspace Server example 46

cube cache

tuning 135

cube viewers 126

D

data access

for servers on multiple machines 93

in database management systems 93

data resources

securing sensitive data resources 77

- data transfer services 16
- database management systems
 - data access in 93
- E**
- e-mail settings
 - for SAS Application Servers 93
- encoding 97
- encryption
 - FIPS 119
- error messages
 - for load balancing 199
 - for object spawner 185
- F**
- FIPS 119
- flattened rows 136
- formats
 - updating references for user-defined formats 92
- G**
- Grid algorithm 45, 50
- grid computing
 - SAS/CONNECT and 18
- Grid Monitoring Server 28
- grid server 27
- group definitions 72
- group objects 73
- H**
- hosts
 - for load-balancing clusters 39
- I**
- Integrated Object Model (IOM) server 139
 - security package for authentication 148
 - Security Support Provider Interface for SSO connections 151
 - start-up parameters 142
- Integrated Object Model (IOM) workspace interface 6
- IOMOPERATE procedure 154
 - examples 175
- L**
- launch command
 - adding system options to (workspace server) 101
- Least Recently Used algorithm 45, 51
- libraries
 - assigning, for client-side pooling 76
 - updating, for servers on multiple machines 92
- load balancing 12, 39
 - See also* load-balancing algorithms
 - See also* load-balancing clusters
 - adding or deleting servers 44
 - client assignment to OLAP servers 35
 - client assignment to workspace and stored process servers 34
 - edits for UNIX and z/OS 43
 - error messages 199
 - initial setup for stored process servers 37
 - installing and configuring server software 43
 - installing server and spawner software 41
 - log files 39
 - MultiBridge connections 36
 - planning, installation, and configuration 36
 - security for 38
 - starting or refreshing object spawners or OLAP servers 43
 - stopping and restarting servers 44
- load-balancing algorithms 38, 51
 - choosing 39
 - Cost 44, 45
 - Grid 45, 50
 - Least Recently Used 45, 51
 - Most Recently Used 45, 51
 - Response Time 45, 50
- load-balancing clusters 34
 - algorithm for 39
 - converting logical server to load balancing 40
 - converting server instance to load balancing 41
 - editing metadata 41
 - hosts and port numbers 39
 - metadata for 41
 - metadata for new servers in cluster 41
 - of OLAP servers 128
 - reading metadata at initialization 34
 - workspace servers in SAS Management Console 36
- locale 97
 - configuring workspace servers for 100
- log files
 - for load balancing 39
- logconfig.xml 42
- logging
 - configuring for object spawner 108

- SAS OLAP Servers 134
- logging level
 - SAS/CONNECT spawner 21
- logical grid server 27
 - configuration files 29
 - metadata object 29
- logical pooled servers
 - puddles for client-side 63
- logical servers 8
 - adding servers to 85
 - adding to SAS Application Server 84
 - converting logical workspace server to client-side pooling 64
 - converting to load balancing 40
 - modifying server definitions 88
 - puddles for client-side logical pooled server 63
 - removing 89
- logical workspace servers
 - converting to client-side pooling 64

M

- metadata
 - editing for load-balancing clusters 41
 - for load-balancing clusters 41
 - for new servers in load-balancing clusters 41
 - reading load-balancing cluster metadata at initialization 34
 - refreshing cube metadata for calculated members and name sets 138
- metadata configuration file 13
- metadata connection 13
- metadata connection and security options
 - object spawner 121
- metadata identities
 - for puddle access 63
- metadata logins
 - for puddle access 63
- metadata objects
 - for SAS/CONNECT 19
 - hierarchy for defining SAS Application Server 8
 - logical grid server 29
- migration
 - OLAP cubes 127
- Most Recently Used algorithm 45, 51
- MultiBridge connections 36

O

- object spawners
 - adding connections to 106
 - administering with Telnet 111
 - changing ports 107

- configuring and starting on z/OS 113
- configuring logging 108
- configuring to accept single sign-on connections 109
- defining 106
- error messages 185
- general invocation options 117
- invocation options 117
- metadata connection and security
 - options 121
- modifying definitions 107
- refreshing 110
- service options 122
- starting or refreshing for load-balancing servers 43
- updating a Windows service for 111
- OBJECTSERVER system option 139
- OBJECTSERVERPARMS system option 142
- OLAP cubes
 - authorizing access to cubes and cube data 133
 - building 129
 - changing permissions 133
 - coalescing 131
 - configuring storage for build files on SAS Workspace Servers 102
 - cube viewers 126
 - deleting 132
 - disabling and enabling 129, 133
 - migrating 127
 - refreshing metadata for calculated members and name sets 138
 - tuning the cube cache 135
 - updating 130
 - updating in place 130
 - updating incrementally 131
- OLAP schemas 126, 138
- OLAP servers 3
 - cleaning up, after abnormal shutdowns 103
 - client assignment to, for load balancing 35
 - starting or refreshing for load-balancing servers 43
- OLAPOPERATE procedure 126
- OLAPServerSSCU.ini 43

P

- performance
 - tuning workspace servers 99
- permissions
 - for OLAP cubes 133
- permissions configuration
 - row-level 72

- pooled workspace servers 3, 12
 - changing SAS options with 98
 - pooling 55
 - See also [server-side pooling](#)
 - client-side 55, 57, 58
 - port bank 56
 - port numbers
 - for load-balancing clusters 39
 - ports
 - changing object spawner-managed server ports 107
 - PROC IOMOPERATE 154
 - examples 175
 - procedure
 - IOMOPERATE 154
 - IOMOPERATE, examples 175
 - puddle login 57
 - puddles 57, 62
 - defining 76
 - for client-side logical pooled server 63
 - metadata identities and logins for access 63
- Q**
- queries
 - managing OLAP queries 134
 - query thread pool
 - tuning 136
- R**
- remote data access
 - with SAS Data Integration Studio 82
 - remote library services 17
 - Response Time algorithm 45, 50
 - Restricted Client-side Pool Administrator 73
 - restricted client-side pooling workspace server 74
 - Restricted Pool Puddle Login group 73
 - row-level security
 - for client-side pooling workspace server 77
 - permissions configuration 72
 - rows, flattened 136
- S**
- SAS Application Server 3, 6
 - accessing remote data with SAS Data Integration Studio 82
 - adding 81
 - adding logical server to 84
 - adding or modifying e-mail settings 93
 - as collection of server components 5
 - defining multiple 83
 - hierarchy of metadata objects for
 - defining 8
 - modifying server definitions 88
 - multiple, with SAS Web Report Studio 83
 - removing logical servers 89
 - SASMeta application server and 8
 - server components 6
 - server context 6
 - structure of 8
 - system options for 139
 - SAS Data Integration Studio
 - accessing remote data 82
 - SAS/CONNECT and 17
 - SAS DATA Step Batch Server 23, 24
 - SAS Deployment Wizard 77
 - SAS Enterprise Miner
 - SAS/CONNECT and 17
 - SAS Generic Batch Server 24
 - SAS Grid Manager 18
 - SAS Grid Server 4, 7, 27
 - initial configuration 29
 - role of 28
 - SAS Java Batch Server 23, 25
 - SAS Management Console
 - clustered workspace servers in 36
 - in OLAP implementation 126
 - SAS Metadata Server 126
 - SAS Object Spawners 14
 - balancing the server workload 14
 - configuration file for metadata connection 13
 - initiating the operator interface 14
 - requesting a server 14
 - tasks 14
 - SAS OLAP Cube Studio 126
 - changing permissions and disabling cubes 133
 - SAS OLAP Server Monitor 127
 - SAS OLAP Servers 7
 - adding 128
 - adding a load-balancing cluster of 128
 - administrative overview 126
 - authorizing access to 132
 - authorizing access to cubes and cube data 133
 - building cubes 129
 - coalescing cubes 131
 - connecting to 128
 - deleting cubes 132
 - disabling and enabling cubes 129
 - flattened row values 136
 - implementation components 126
 - installing and configuring 128
 - logging 134

- managing sessions and queries 134
- migrating OLAP cubes 127
- monitoring 133
- OLAP schemas 138
- refreshing cube metadata for calculated members and name sets 138
- starting 128
- stopping, pausing, and resuming 129
- tuning cube cache 135
- tuning query thread pool 136
- tuning subquery caches 135
- tuning with advanced server options 134
- updating cubes 130
- SAS options
 - changing with pooled workspace server 98
 - changing with workspace server 98
- SAS Pooled Workspace Server 6, 12
- SAS Stored Process Server 11
 - Cost algorithm example 48
 - MultiBridge connections 36
- SAS Web Report Studio
 - creating deployment for client-side pooling 77
 - multiple SAS Application Servers with 83
 - verifying client-side connection pooling 69
- SAS Workspace Server 6, 12
 - cleaning up, after abnormal OLAP server shutdowns 103
 - configuring storage for OLAP cube build files on 102
 - Cost algorithm example 46
 - in OLAP implementation 126
- SAS/CONNECT 15, 17
 - accessing remote data 17
 - compute services 16
 - configuration files 21
 - data transfer services 16
 - grid computing and 18
 - overview of services 16
 - remote library services 17
 - SAS Data Integration Studio and 17
 - SAS Enterprise Miner and 17
- SAS/CONNECT server 3, 7
 - accessing remote data 17
 - configuration files 21
 - initial configuration 21
 - metadata objects 19
- SAS/CONNECT spawner
 - changing logging level 21
- SAS/SECURE 119
- SASApp - Logical Workspace Server object 8
- SASApp object 5
- SASMain object 5
- SASMeta application server 8
- sasv9_usermods.cfg 42
- scheduling 23
- SECPACKAGE system option 148
- SECPACKAGELIST system option 150
- security
 - for load balancing 38
 - row-level, for client-side pooling workspace server 77
 - securing sensitive data resources 77
 - security package for IOM server authentication 148
- security authentication packages 150
- Security Support Provider Interface (SSPI) 151
- server components 5, 6
- server context 6
- server instance
 - converting to load balancing 41
- Server Manager 8, 126
- server pooling
 - See pooling
- server ports
 - changing object spawner-managed 107
- server-side pooling 55
 - connection process 56
 - how it works 56
- service options
 - object spawner 122
- sessions
 - managing OLAP sessions 134
- shortcuts.ini 43
- single sign-on connections
 - configuring the object spawner for 109
- spawner software
 - for load-balancing servers 41
- SSO connections to IOM servers
 - Security Support Provider Interface for 151
- SSPI system option 151
- Stored Process Server 6
- stored process servers 3, 11
 - client assignment to, for load balancing 34
 - default 12
 - encoding and locale information 97
 - initial setup for load balancing 37
 - moving 97
 - on multiple machines 93
 - pooling 55
- stored processes 11
- subquery caches
 - tuning 135
- system options

- adding to workspace server launch command 101
- for SAS Application Server components 139

T

- Telnet
 - administering object spawner 111
- temporary files
 - cleaning up, after abnormal OLAP server shutdowns 103
- tuning
 - cube cache 135
 - query thread pool 136
 - SAS OLAP Servers 134
 - subquery caches 135
 - workspace servers 99

U

- UNIX
 - load-balancing servers 43
- user accounts 72
- user definitions 72
- user objects 73
- user-defined formats
 - updating references, for servers on multiple machines 92

W

- Windows services

- updating for object spawner 111
- workspace servers 3, 11
 - adding system options to launch command 101
 - changing SAS options with 98
 - client assignment to, for load balancing 34
 - clustered in SAS Management Console 36
 - configuration options for pool 65
 - configuration tasks 103
 - configuring for a locale 100
 - configuring for row-level security 77
 - converting logical server to client-side pooling 64
 - creating restricted client-side pool 74
 - default 12
 - encoding and locale information 97
 - moving 97
 - on multiple machines 93
 - pooling 55
 - tuning 99

Z

- z/OS
 - configuring and starting object spawner 113
 - load-balancing servers 43