
SilverStream eXtend Application Server
Administrator's Guide

Version 4

July 2002

SilverStream[®]

Copyright ©2002 SilverStream Software, Inc. All rights reserved.

SilverStream software products are copyrighted and all rights are reserved by SilverStream Software, Inc.

SilverStream and jBroker are registered trademarks and SilverStream eXtend is a trademark of SilverStream Software, Inc.

Title to the Software and its documentation, and patents, copyrights and all other property rights applicable thereto, shall at all times remain solely and exclusively with SilverStream and its licensors, and you shall not take any action inconsistent with such title. The Software is protected by copyright laws and international treaty provisions. You shall not remove any copyright notices or other proprietary notices from the Software or its documentation, and you must reproduce such notices on all copies or extracts of the Software or its documentation. You do not acquire any rights of ownership in the Software.

Jakarta-Regexp Copyright ©1999 The Apache Software Foundation. All rights reserved. Ant Copyright ©1999 The Apache Software Foundation. All rights reserved. Xalan Copyright ©1999 The Apache Software Foundation. All rights reserved. Xerces Copyright ©1999-2000 The Apache Software Foundation. All rights reserved. Jakarta-Regexp, Ant, Xalan and Xerces software is licensed by The Apache Software Foundation and redistribution and use of Jakarta-Regexp, Ant, Xalan and Xerces in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notices, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>)." Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear. 4. The names "The Jakarta Project", "Jakarta-Regexp", "Xerces", "Xalan", "Ant" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org <<mailto:apache@apache.org>>. 5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of The Apache Software Foundation. THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright ©1996-2000 Autonomy, Inc.

Copyright ©2000 Brett McLaughlin & Jason Hunter. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the disclaimer that follows these conditions in the documentation and/or other materials provided with the distribution. 3. The name "JDOM" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact license@jdom.org <<mailto:license@jdom.org>>. 4. Products derived from this software may not be called "JDOM", nor may "JDOM" appear in their name, without prior written permission from the JDOM Project Management (pm@jdom.org <<mailto:pm@jdom.org>>). THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Sun Microsystems, Inc. Sun, Sun Microsystems, the Sun Logo Sun, the Sun logo, Sun Microsystems, JavaBeans, Enterprise JavaBeans, JavaServer Pages, Java Naming and Directory Interface, JDK, JDBC, Java, HotJava, HotJava Views, Visual Java, Solaris, NEO, Joe, Netra, NFS, ONC, ONC+, OpenWindows, PC-NFS, SNM, SunNet Manager, Solaris sunburst design, Solstice, SunCore, SolarNet, SunWeb, Sun Workstation, The Network Is The Computer, ToolTalk, Ultra, Ultracomputing, Ultraserver, Where The Network Is Going, SunWorkShop, XView, Java WorkShop, the Java Coffee Cup logo, Visual Java, and NetBeans are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

The software licensed hereby contains modified Sun NetBeans software which is available, together with the Sun Public License pursuant to which such NetBeans software may be used, at <http://www.silverstream.com/workbenchdownload>. Terms of this Agreement that differ from the terms of the Sun Public License are offered solely by SilverStream.

SilverStream eXtend Workbench software contains Sun NetBeans software that has been modified by SilverStream. The source code for such software may be found at <http://www.silverstream.com/workbenchdownload> together with the Sun Public License that governs the use of such modified software. The Original Code is NetBeans. The Initial Developer of the Original Code is Sun Microsystems, Inc. Portions Copyright 1997-2000 Sun Microsystems, Inc. All Rights Reserved. The Contributor to Covered Code is SilverStream Software, Inc.

IBM Jikes™ and Bean Scripting Framework (BSF) Copyright ©2001, International Business Machines Corporation and others. All Rights Reserved. This software contains code in executable form obtained pursuant to, and the use of which is subject to, the IBM Public License, a copy of which may be obtained at <http://oss.software.ibm.com/developerworks/opensource/license10.html>. Source code for Jikes™ is available at <http://oss.software.ibm.com/developerworks/opensource/jikes/>. Source code for BSF is available at <http://oss.software.ibm.com/developerworks/projects/bsf>.

Copyright ©2001 Extreme! Lab, Indiana University License. <http://www.extreme.indiana.edu>. Permission is hereby granted, free of charge, to any person obtaining a copy of the Indiana University software and associated Indiana University documentation files (the "IU Software"), to deal in the IU Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the IU Software, and to permit persons to whom the IU Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the IU Software. THE IU SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE IU SOFTWARE OR THE USE OR OTHER DEALINGS IN THE IU SOFTWARE.

Graph Layout Toolkit and Graph Editor Toolkit ©1992 - 2001 Tom Sawyer Software, Oakland, California, All Rights Reserved.

This Software is derived in part from the SSLava™ Toolkit, which is Copyright ©1996-1998 by Phaos Technology Corporation. All Rights Reserved.

SearchServer © 2000 Hummingbird Communications, Inc.

Copyright © 1994-2002 W3C® (Massachusetts Institute of Technology, Institut National de Recherche Informatique et en Automatique, Keio University), all Rights Reserved. <http://www.w3.org/consortium/legal>. This W3C work (including software, documents, or other related items) is being provided by the copyright holders under the following license. By obtaining, using and/or copying this work, you (the licensee) agree that you have read, understood, and will comply with the following terms and conditions: Permission to use, copy, modify, and distribute this software and its documentation, with or without modification, for any purpose and without fee or royalty is hereby granted, provided that you include the following on ALL copies of the software and documentation or portions thereof, including modifications, that you make: 1. The full text of this NOTICE in a location viewable to users of the redistributed or derivative work. 2. Any pre-existing intellectual property disclaimers, notices, or terms and conditions. If none exist, a short notice of the following form (hypertext is preferred, text is permitted) should be used within the body of any redistributed or derivative code: "Copyright © [date-of-software] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved. <http://www.w3.org/Consortium/Legal/>" 3. Notice of any changes or modifications to the W3C files, including the date changes were made. (We recommend you provide URIs to the location from which the code is derived.) THIS SOFTWARE AND DOCUMENTATION IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE OR DOCUMENTATION WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS. COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE SOFTWARE OR DOCUMENTATION. The name and trademarks of copyright holders may NOT be used in advertising or publicity pertaining to the software without specific, written prior permission. Title to copyright in this software and any associated documentation will at all times remain with copyright holders.

Contents

About This Book xv

- Purpose xv
- Audience xv
- Organization xv

Chapter 1 Administration Quick Reference 1

- SMC panels 1
 - Configuration options 2
 - Security options 4
 - Monitor options 6
 - Deployment options 6
- Administration tasks 7
 - Data source configuration 7
 - General server management 7
 - Security 8
 - Tuning and performance 8
 - Load balancing and failover 8
 - Using the Server Administration API 9
 - Troubleshooting 9

PART I ADMINISTRATION BASICS

Chapter 2 Administration Overview 13

- The SilverStream eXtend Application Server 13
 - Three-tiered communications 14
 - Application server environments 15
- SilverStream administration 16
 - Client tier administration 17
 - Server administration 18
 - Data tier administration 20
- The SilverStream Management Console (SMC) 20
 - Running the SMC 21
 - The SMC user interface 23
 - Logging in 25
 - Logging out 26
 - Online help 26

Chapter 3 Server Configuration 27

- Server configurations 27
 - Production environment 28
 - SilverStream classic development environment 29
- Firewalls and proxy servers 31
 - Configuration with a firewall and proxy server 33
- Network configurations 34
 - Simple intranet configuration 34
 - Intranet cluster configuration 36
 - Simple Internet configuration 38
 - Internet cluster configuration 40
 - Demilitarized Zone (DMZ) Internet configuration 42
- HTTP server and Web basics 44
 - Uniform Resource Locators (URLs) 44
 - SilverStream resources 45
 - HTTP communications 46
- Session management 48
 - Cookies 48
 - URL rewriting 49
 - How session tracking works 49

Chapter 4 Data Source Configuration 51

- About data source configurations 51
 - J2EE configuration 51
 - Classic SilverStream configuration 52
 - Data source configuration tasks 53
- Preparing a database for access 53
 - Database access 55
 - Adding JDBC driver JARs to the server classpath 56
 - Testing your database connections 57
- Configuring the SilverMaster database 58
 - SilverMaster functions 58
 - Moving the SilverMaster database 59
- Configuring classic application databases 59
 - Adding a database to the server 59
 - Removing a database from the server 65
 - Configuring a database 66
- Configuring connection pools 69
 - Adding a JDBC connection pool 69
 - Adding a Connector connection pool 81
 - Removing a connection pool 85
 - Maintaining a connection pool 85

PART II ADMINISTERING THE SERVER

Chapter 5 Running the Server 91

- Starting the SilverStream server 91
 - Starting the server 92
 - Using startup options 92
 - Specifying the VM to use 98
 - Starting the server on a specific IP address or hostname 99
- Shutting down the SilverStream server 99
- Restarting the SilverStream server 100
- Maintaining SilverStream processes running as services 101
 - Using SilverServiceUtil 102
- Setting up separate ports 106
 - Using separate ports with your firewall 107
 - About enabling ports 108
 - When to use the administration port 108
 - Port types 109
- Specifying general server properties 110
- Using server logging 112
- Specifying ORB settings 116
- Running multiple servers on one host 119
 - Specifying unique ports 119
 - Host-based properties 120
- Specifying character set encoding 122
- Running the JMS (jBroker MQ) server 123
 - Starting the JMS server 124
 - Using JMS servers in clusters 125
 - Displaying JMS debug messages 125

Chapter 6 Setting Up Users and Groups 127

- About Silver Security users and groups 127
 - About your administrator account 129
- Managing Silver Security users and groups 130
 - Adding Silver Security users 130
 - Editing user properties 131
 - Adding Silver Security groups 132
- Using the Locksmith privilege 134

Chapter 7	Maintaining the Server	135
	Administering a SilverStream server remotely	135
	Managing licenses	136
	About licenses	136
	Setting the AGCLASSPATH variable	138
	Maintaining deployed J2EE objects	139
	Using the Deployed Objects panel	140
	Managing J2EE transactions	144
	Monitoring server activity	146
	Displaying charts of server activity	147
	Displaying logs	151
	Displaying views of server statistics	152
	Integrating with existing Web servers	159
	Setting up mail on the server	159
	Setting Fulcrum full-text properties	161
Chapter 8	Using the Web Server Integration Modules	165
	About the WSI modules	165
	How the WSI modules work	166
	WSI module examples	166
	Using a WSI in a cluster	171
	Planning your WSI installation	172
	Installing and configuring the WSI module	172
	About the WSI configuration file	174
	Configuration file settings	175
	Enabling the WSI module for IIS for Windows	182
	Enabling the WSI module for iPlanet for Windows and UNIX platforms	184
	WSI for iPlanet sample configuration files	186
	Redirecting requests to multiple SilverStream servers	188
	A sample configuration file redirecting to different SilverStream servers	188
	About connection pooling	190
	Authentication and security integration considerations	190
	Using IIS NTLM authentication with the WSI module	191
	The AgWSIUser utility	191
	HTTPS client certificate authentication issues	192
	enCommerce getAccess integration	192
Chapter 9	Setting Up Security	199
	Security configuration	200
	Types of encryption used for authentication	202
	Security functions	203

About authentication	203
Establishing a secure connection to the server	204
Establishing a secure connection between a Java client and the SilverStream server	204
Establishing a secure connection between an HTML client and the SilverStream server	205
Establishing a secure (SSL) connection between an EJB client and the SilverStream server	206
Accessing security provider systems	208
Adding security provider access	209
Using NT security	211
Using LDAP security	213
Using NIS+ security	220
Accessing users and groups	220
Using security provider login formats	221
Overriding defaults for login name components	223
Using certificates	226
About certificates	227
Creating and installing server certificates using the SMC	229
Creating and installing server certificates from the command line	240
Viewing server certificates	250
Enabling RSA/DSA ports	251
Turning off HTTP communications	252
Restricting SSL cipher suites	253
Managing Certificate Authorities	255
Installing and managing client certificates	256
Verifying SSL server certificates for Java clients	264
Enabling authentication	266
Using Cryptographic Hardware Integration	268
Managing trusted clients	269

Chapter 10 Using Security 271

Authorization and access control	271
Authorization and access control	271
Permission types	272
Administrative server permissions	272
Database object permissions	273
How access works	275
Changing access	276
Restricting permissions	278

Making secure application objects executable	285
Typical permissions for classic applications	285
Typical permissions for J2EE application objects	287
Default server and object security	288
Default object security	288
Default group permissions	288
Locking down servers, clusters, and applications	289
Ways to lock down a server	290
Using the SMC to lock down the server or an application	290
Using SilverCmd to lock down the server, an application, or a cluster	291
Securing the production server	292
Security checklist	292
Step 1: Design firewalls	292
Step 2: Set up a unique database account	292
Step 3: (Optional) Set up SSL	292
Step 4: (Optional) Set up unique ports	293
Step 5: Set up users, groups, and security providers	293
Step 6: Require authentication at the server	293
Step 7: Restrict the directory listing on the server	294
Step 8: Secure the administration resource	295
Step 9: Secure the SilverMaster database	297
Step 10: Secure your application databases	298
Step 11: Map J2EE security roles	299
Securing the development server	299
Excluding robots	301

Chapter 11 Tuning the Server 303

Setting performance parameters	303
Managing client connections	306
Client sessions and threads	306
Client connection parameters	307
Managing the server content cache	311
Managing connection pools for J2EE applications	314
About connection pool connections	314
Setting the number of connection pool connections	315
Managing database connections for classic applications	318
About database connections and performance	319
Setting the maximum and minimum number of database connections	320
Using prefetch buffers	325

Chapter 12	Administering a Cluster	327
Server clustering		327
Cluster components		328
The Cache Manager		330
The Load Manager		332
The Dispatcher		333
Component failover		336
Persistent failure		337
Setting up a server cluster		337
Starting the clustering components		338
Installing cluster servers		340
Creating the cluster profile		342
Restarting the clustered servers		347
Administering a server cluster		348
About properties in a clustered environment		348
Setting cluster-level properties		349
Setting server-level properties in a cluster		352
Specifying a server's relative load weight		353
Managing failover		354
Cache Manager properties		354
Load Manager properties		355
Dissolving a cluster		356
Changing the clustering components' properties		357
Changing hosts		357
Changing ports		357
Installing certificates in a cluster		358
About the server certificates		358
How to do it		359
Setting up Fulcrum in a cluster		360
Setting up Fulcrum on Windows NT		360
Setting up Fulcrum on UNIX		365
Chapter 13	Using the Server Administration API	367
Introduction to the Administration API		368
How the Administration API is organized		370
How server objects are organized		371
More about containers and elements		374
Getting started with the Administration API		376
Obtaining a server object from a client application		377
Obtaining a server object from a J2EE application		378

Obtaining a server object from a classic SilverStream page or business object	379
Getting a server's properties	379
Working with server elements	380
Common tasks	386
Sample code	387
Enumerating security providers, users, and groups	387
Identifying users	399
Customizing the logging class	400
Chapter 14 Troubleshooting	407
Using error logging	407
Low-level debugging	408
Setting JDBC/ODBC tracing	409
Using the server's command shell	411
Using the Watcher	411
Common problems starting the SilverStream server	413
System resource problems	413
Business object generating errors	413
Database not synchronized	414
Using SilverMonitor	414
Using the SilverMasterInit program	416
Command-line options	417
Using SilverMasterInit to recreate or refresh SilverMaster	426
Regaining access to SilverMaster	428
Handling a stack overflow	429
About stacks	430
What to do if you get a stack overflow	430
Changing the stack size	430
Changing the Java stack size	431
Example	431
Miscellaneous issues	432
Browser issues	432
Server appears to be hung	432
Socket exceptions	433

PART III APPENDIXES

Appendix A The httpd.props File 437

- About the httpd.props file 437
- Server properties 438

Appendix B The SilverStream SNMP Agent 445

- About SNMP 445
- SNMP implementation overview 446
 - How the SilverStream components work 447
 - Process flow and terminology 448
- Setting up SNMP for the SilverStream server 450
 - Installing SNMP as a service 450
 - Installing the SilverStream server 451
 - Deploying the AgSNMPGetStats servlet 452
 - Testing the program 452
 - Setting up access from your SNMP Management node 453

Appendix C SilverStream System Tables and URLs 455

- Where application and system data is stored 455
- SilverStream internal system tables 455
- SilverStream database URLs 457

About This Book

Purpose

This guide explains how to administer the SilverStream eXtend Application Server.

Audience

This guide is for the SilverStream eXtend Application Server administrator.

Organization

This section provides a brief summary of the Quick Reference (the first chapter) and each of the book's three parts.

Part	Description
Chapter 1, "Administration Quick Reference"	A listing of cross-references to help you get to the information you want as fast as possible.
Part I, "Administration Basics"	This part: <ul style="list-style-type: none">• Introduces you to the SilverStream eXtend Application Server's three-tiered architecture and outlines your tasks as a SilverStream system administrator. It also introduces the SilverStream Management Console (SMC), which you will use to perform many administrative tasks.• Describes the basic hardware configurations for the SilverStream eXtend Application Server and explains how the server operates in the Web environment.• Describes configuration settings for SilverStream-supported databases and other data sources. It also describes the SilverMaster database, and briefly outlines general database administration.

Part	Description
Part II, “Administering the Server”	<p>This part provides instructions for common administrative tasks that you will need to perform, including:</p> <ul style="list-style-type: none">• Running the server, including how to start and stop the server and how to log server activity• Setting up SilverStream users and groups• Performing common maintenance tasks such as adding databases to the server, setting up mail on the server, managing J2EE deployed objects, and managing licenses• Using Web Server Integration modules to integrate a SilverStream server with an external Web server• Setting up and using security in both the HTTP and HTTPS environments• Tuning the server for optimum performance, including managing connections to clients and databases• Administering a cluster to provide load balancing and failover• Using the Server Administration API to programmatically administer the server• Troubleshooting problems
Part III, “Appendixes”	<p>This part describes:</p> <ul style="list-style-type: none">• The <code>httpd.props</code> file, which you can use to edit selected server properties• How Simple Network Management Protocol (SNMP) is implemented in SilverStream, and how to run SNMP on a SilverStream server• SilverStream system tables and SilverMaster database URLs

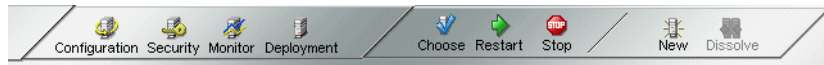
1

Administration Quick Reference

Use this Quick Reference to quickly get to the following information:

- SMC panels
- Administration tasks

SMC panels



The SMC is divided into three main areas:




- Configuration options
- Security options
- Monitor options
- Deployment options





This section describes the panels in each of the areas.

NOTE The panels are different if you are managing a clustered environment. For a quick reference to the SMC panels for a cluster, see “Administering a server cluster” on page 348.

Configuration options




Configuration options consist of the following panels.




Panel	Description / Where to go for more information
General	<p>General, server logging, and ORB/RMI settings. Use this panel to configure separate ports for different types of users and operations.</p> <p> See:</p> <ul style="list-style-type: none"> • General: “Specifying general server properties” on page 110 • Ports: “Setting up separate ports” on page 106 • Logging: “Using server logging” on page 112 • ORB/RMI: “Specifying ORB settings” on page 116
Advanced	<p>Debugging, performance, server cache, and J2EE transactions.</p> <p> See:</p> <ul style="list-style-type: none"> • Debug: “Low-level debugging” on page 408 • Performance: “Setting performance parameters” on page 303 • Cache: “Managing the server content cache” on page 311 • Transactions: “Managing J2EE transactions” on page 144
Pools	<p>Adding, removing, and maintaining JDBC and Connector connection pools.</p> <p> See:</p> <ul style="list-style-type: none"> • Managing JDBC connection pools: “Configuring connection pools” on page 69 • Managing Connector connection pools: “Configuring connection pools” on page 69

Panel	Description / Where to go for more information
Databases	<p>Adding and removing databases. Configuring how the server communicates with databases, including user name and password information, JDBC connectivity information, and the minimum and maximum number of database connections. You can also use this panel to synchronize database schema and delete idle connections.</p> <p> See:</p> <ul style="list-style-type: none"> • Adding and remove databases: “Configuring classic application databases” on page 59 • Minimum and maximum number of connections: “Managing database connections for classic applications” on page 318 • All other configuration tasks: “Configuring classic application databases” on page 59
Connections	<p>Managing client connection settings.</p> <p> See “Client connection parameters” on page 307</p>
Licenses	<p>Managing server licenses.</p> <p> See “Managing licenses” on page 136</p>
Classic	<p>Setting up e-mail accounts that work with SilverStream business objects triggered on mail receipt, and setting up Fulcrum full-text settings.</p> <p> See:</p> <ul style="list-style-type: none"> • Setting up mail: “Setting up mail on the server” on page 159 • Full text: “Setting Fulcrum full-text properties” on page 161

Security options




Security options consist of the following panels.

Panel	Description / Where to go for more information
General	<p>General security settings.</p> <p> See:</p> <ul style="list-style-type: none"> • Require user authentication, Disable HTML directory listing, and Allow users to modify own account: “Enabling authentication” on page 266 • Security resource timeout: “Resetting the security resource timeout” on page 210 • Default security realm and authority: “Overriding defaults for login name components” on page 223
Advanced	<p>HTTPS client certificate levels, accelerator settings, and trusted clients.</p> <p> See:</p> <ul style="list-style-type: none"> • Client certificate level in HTTPS: “Enabling and installing client certificates” on page 256 • Accelerator settings: “Using Cryptographic Hardware Integration” on page 268 • List of trusted clients: “Managing trusted clients” on page 269
Permissions	<p>Controlling access to any SilverStream object. You can set up access control at the cluster, server, database, or individual object level.</p> <p> See “Authorization and access control” on page 271</p>

Panel	Description / Where to go for more information
Users & Groups	<p>Adding Silver Security and certificate users. Adding Silver Security groups. Viewing users in external security providers. Editing user properties.</p> <p> See:</p> <ul style="list-style-type: none"> • Adding Silver Security users and groups: “Managing Silver Security users and groups” on page 130 • Adding certificate users: “Manually installing client certificates” on page 260 • Viewing users and groups: “Accessing users and groups” on page 220 • Using external security providers: “Accessing security provider systems” on page 208 • Editing user properties: “Editing user properties” on page 131
Certificates	<p>Viewing certificates that have been installed on the server. Viewing recognized Certificate Authorities.</p> <p> See:</p> <ul style="list-style-type: none"> • Creating and installing server certificates: “Creating and installing server certificates using the SMC” on page 229 • Viewing server certificates: “Viewing server certificates” on page 250 • Managing Certificate Authorities: “Managing Certificate Authorities” on page 255 • DSA and RSA port properties: “Enabling RSA/DSA ports” on page 251
Security Providers	<p>Configuring SilverStream to recognize external security providers, including Windows NT directory services, LDAP, NIS+, and certificate issuers.</p> <p> See “Accessing security provider systems” on page 208.</p>





Monitor options

Monitor options consist of the following panels.

Panel	Description / Where to go for more information
Charts	Displaying real-time charts of various server statistics.  See “Displaying charts of server activity” on page 147
Logs	Displaying logs if you have enabled server logging.  See “Displaying logs” on page 151
Statistics	Displaying tabular views of server statistics related to sessions and threads, as well as summary statistics.  See “Displaying views of server statistics” on page 152

Deployment options

Deployment options consist of the following panels:

Panel	Description / Where to go for more information
Deployed objects	Viewing and managing J2EE objects. Use this panel to enable, disable, and shut down EJB JARs that have been deployed on the server.  See “Maintaining deployed J2EE objects” on page 139
JNDI tree	 See “Maintaining deployed J2EE objects” on page 139
Manage URLs	 See “Maintaining deployed J2EE objects” on page 139
Resource Adapters	 See “Maintaining deployed J2EE objects” on page 139

Administration tasks

This section provides a quick reference to common administration tasks.

Data source configuration

- “Configuring the SilverMaster database” on page 58
- “Preparing a database for access” on page 53
- “Configuring classic application databases” on page 59
- “Configuring connection pools” on page 69

General server management

- “Starting the SilverStream server” on page 91
- “Using startup options” on page 92
- “Shutting down the SilverStream server” on page 99
- “Restarting the SilverStream server” on page 100
- “Setting up separate ports” on page 106.
- “Specifying general server properties” on page 110
- “Using server logging” on page 112
- “Specifying ORB settings” on page 116
- “Running multiple servers on one host” on page 119
- “Managing Silver Security users and groups” on page 130
- “Using the Locksmith privilege” on page 134
- “Administering a SilverStream server remotely” on page 135
- “Setting up mail on the server” on page 159
- “Managing licenses” on page 136
- “Setting the AGCLASSPATH variable” on page 138
- “Maintaining deployed J2EE objects” on page 139
- “Setting Fulcrum full-text properties” on page 161
- “Monitoring server activity” on page 146
- “Installing and configuring the WSI module” on page 172
- “Setting up SNMP for the SilverStream server” on page 450

Security

- “Establishing a secure connection to the server” on page 204
- “Accessing security provider systems” on page 208
- “Using certificates” on page 226
- “Enabling RSA/DSA ports” on page 251
- “Enabling authentication” on page 266
- “Restricting permissions” on page 278
- “Excluding robots” on page 301
- “Locking down servers, clusters, and applications” on page 289
- “Security checklist” on page 292

Tuning and performance

- “Managing database connections for classic applications” on page 318
- “Setting performance parameters” on page 303
- “Managing client connections” on page 306
- “Managing the server content cache” on page 311

Load balancing and failover

- “Setting up a server cluster” on page 337
- “Installing cluster servers” on page 340
- “Creating the cluster profile” on page 342
- “Administering a server cluster” on page 348
- “Restarting the clustered servers” on page 347
- “Specifying a server’s relative load weight” on page 353
- “Managing failover” on page 354
- “Dissolving a cluster” on page 356
- “Installing certificates in a cluster” on page 358
- “Setting up Fulcrum in a cluster” on page 360

Using the Server Administration API

- “Obtaining a server object from a client application” on page 377
- “Obtaining a server object from a classic SilverStream page or business object” on page 379
- “Working with server elements” on page 380



Also see “Sample code” on page 387

Troubleshooting

- “Using error logging” on page 407
- “Low-level debugging” on page 408
- “Setting JDBC/ODBC tracing” on page 409
- “Using the Watcher” on page 411
- “Using SilverMonitor” on page 414
- “Handling a stack overflow” on page 429

Part I Administration Basics

This part describes the basics of administering the SilverStream eXtend Application Server

- Chapter 2, “Administration Overview”
- Chapter 3, “Server Configuration”
- Chapter 4, “Data Source Configuration”

2

Administration Overview

This chapter introduces the SilverStream eXtend Application Server architecture and outlines administrative tasks in the SilverStream server environment.

It contains sections on:

- The SilverStream eXtend Application Server
- SilverStream administration
- The SilverStream Management Console (SMC)

The SilverStream eXtend Application Server

The SilverStream eXtend Application Server is a multithreaded J2EE application server implemented in Java. Client communications are conducted through the HyperText Transfer Protocol (HTTP), the most common protocol for the World Wide Web.

NOTE There are two situations when clients do not use HTTP but instead use RMI (Java's Remote Method Invocation) to communicate with the server: (1) when calling an Enterprise JavaBean and (2) when an external client using AgRuntime establishes a connection using `connectRMI()`.

The SilverStream server provides business logic processing and access to new and legacy data.

Three-tiered communications

The SilverStream server provides a three-tiered architecture that consists of a **client tier**, a **middle tier**, and a **data tier**.

Tier	Description
Client	Includes Web browsers, the SilverJ2EEClient and SilverJRunner applications, and the SilverStream Designer. The presentation can be Struts, JSP pages, servlets, or any combination of J2EE technology, or SilverStream's classic technology (forms and pages).
Middle	Includes the SilverStream eXtend Application Server. The middle tier includes two runtime environments: <ul style="list-style-type: none"> • Web container—(which primarily contains JSP pages and servlets) provides support for receiving and responding to client requests. • EJB container—provides built-in support for transaction management (among other things). Containers also provide built-in support for accessing enterprise information systems, such as supporting JDBC to access relational databases.
Data	Includes data from relational databases, SAP, Peoplesoft, Notes, and J2EE Connectors. For a list of supported databases and connectors, see the <i>Release Notes</i> .






Three-tiered communications provide the following benefits:

Benefit	Description
Security management	The SilverStream server mediates all communication to the database, enforcing the security you set up.
Code management	By encapsulating your business logic into server-side objects, it is easier to manage and maintain code, especially in a large-scale development environment.
Data validation	Because the business logic is contained in one tier, you can protect data by controlling access and operations from one central point.

Application server environments

As a SilverStream administrator, you'll set up and support these environments:

Environment	Description
Production	<p>The SilverStream production environment consists of one or more application servers, one or more database servers or enterprise information systems (EIS), and clients.</p> <p>Clients include browsers (running HTML or applets) and Java clients (a SilverStream client using SilverJRunner, SilverJ2EEClient, or a non-SilverStream Java client).</p> <p> For more information about production environment configurations, see Chapter 3, "Server Configuration".</p>
Development	<p>The SilverStream development environment consists of one or more development servers, one or more database servers or EIS systems, SilverStream eXtend Workbench (for developing J2EE applications), and the SilverStream Designer (for developing classic applications).</p> <p> For more information about Workbench, see the Workbench help.</p> <p> For more information about the Designer, see the server's Classic Development Help.</p>




Environment	Description
Deployment	<p>The deployment environment consists of one or more application servers, one or more database servers or EIS systems, and the deployment tools.</p> <p>For J2EE applications—Deployment tools include SilverCmd or the deployment tools provided by SilverStream eXtend Workbench. You might be responsible for deploying J2EE applications to the server. Deployment responsibilities can include mapping role references to users and groups in the security system and mapping resource references to entities in the data tier. For more information, see the chapter on J2EE archive deployment in the <i>Facilities Guide</i>.</p> <p>For SilverStream classic applications—Deployment tools are provided by the SilverStream Designer. For more information, see the server’s Classic Development Help.</p>

SilverStream administration

In addition to administering different application server environments, you have administrative responsibilities for each of the three tiers in the server architecture: the client tier, the server tier, and the data tier.

Client tier administration






Specific requirements for running each of the SilverStream client types are as follows:







Client	Requirements
Browser	<p>The browser requirements depend on the kind of HTML applications being run.</p> <p> For more information, see the chapter on deployment in the <i>Programmer's Guide</i> of the server's Classic Development Help.</p> <p>NOTE Browser administration is not covered in this guide. See your browser documentation for more information.</p>
Java-based clients	<p>SilverStream provides two programs that can be deployed to user machines for running Java-based clients:</p> <ul style="list-style-type: none"> • SilverJ2EEClient is used to host J2EE application clients on user machines. • SilverJRunner is used to host SilverStream forms on user machines. <p> For more information, see the chapter on SilverJ2EEClient and SilverJRunner in the <i>Facilities Guide</i>.</p>
SilverStream Designer	<p>The SilverStream Designer is installed using the SilverStream installation program. The Designer is the SilverStream integrated development environment (IDE) for application developers. Each developer using the SilverStream IDE should have a Designer installed on their local workstation.</p>
Non-SilverStream Java client	<p>The SilverStream server also supports non-SilverStream Java clients, such as a command-line Java application, a Java GUI application using AWT or Swing written with a third-party IDE, or an externally written applet.</p> <p> For more information, see the chapter on external clients in the <i>Programmer's Guide</i> of the server's Classic Development Help.</p>


Server administration

In the SilverStream environment, server administration is your major responsibility. Most server administration is accomplished through the SilverStream Management Console (SMC), a standalone administration tool that is described later in this chapter.

These are the major areas of SilverStream administration:

Administration area	Server environment	Description
Installation	Setup	Use the SilverStream eXtend Application Server installation program to install the product.  See the <i>Installation Guide</i> for instructions. See the <i>Release Notes</i> for the latest system requirements.
Data sources	Development/ Production	You need to specify how the server will communicate with one or more databases or connection pools, and how to maintain connections.  See Chapter 4, “Data Source Configuration”.
Licenses	Production	The server requires a valid license to run. You receive a license when you purchase the product. You can obtain additional licenses from SilverStream.  See “Managing licenses” on page 136.
Statistics	Production	Once a production server is up and running, you can monitor statistics in order to tune performance and schedule maintenance activity.  See Chapter 7, “Maintaining the Server”.
Logging	Development/ Production	The SilverStream server can log different types of system information to either a database or a file.  See “Using server logging” on page 112.

Administration area	Server environment	Description
Certificates	Production	<p>Certificates are used in Secure Sockets Layer (SSL) connections for the server to authenticate itself to clients and for the clients to authenticate themselves to the server. You can install RSA and DSA certificates on the SilverStream server.</p> <p> See Chapter 9, “Setting Up Security”.</p>
Security	Production	<p>SilverStream offers different levels of security: server security, application security, and database row-level security.</p> <p> See Chapter 10, “Using Security”.</p>
Server performance	Production	<p>The server has several settings that define its behavior when busy or under light load. You can define the number of connections required for the server to be in a specific state.</p> <p> See Chapter 11, “Tuning the Server”.</p>
Mail	Development/ Production	<p>The server can get mail from a POP3 or IMAP server for processing by business objects. You can configure the mail accounts it reads and how often it checks for mail.</p> <p> See “Setting up mail on the server” on page 159.</p>
Load balancing	Production	<p>Load balancing lets you use multiple servers (clusters) in a large-scale production environment.</p> <p> See Chapter 12, “Administering a Cluster”.</p>
Troubleshooting	Development/ Production	<p> See Chapter 14, “Troubleshooting”.</p>

 For a quick reference of administration tasks, see “Administration tasks” on page 7.

Data tier administration

In the data tier, your administrative duties include:

- Configuring, adding, and removing databases
- Configuring, adding, and removing connection pools
- Managing data source connection pools

One of the key features of the SilverStream environment is the SilverMaster database catalog, which is created at installation as a relational database. Among other important functions, the SilverMaster maintains a catalog of databases managed by the server.



For more information, see Chapter 4, “Data Source Configuration”.

The SilverStream Management Console (SMC)

The SilverStream Management Console (SMC) is a standalone administration tool that you can use for most of your SilverStream administration tasks.

You can perform the following administrative tasks using the SMC:

- Maintain the SilverStream server environment
- Monitor the environment
- Modify configuration options to improve performance
- Set up authentication
- Set up and maintain server clusters for load balancing

You can administer multiple servers from the same SMC window.

A few of the SMC settings affect entries in the `httpd.props` file, which you can edit directly. However, SilverStream recommends that whenever possible you use the SMC to change server settings. For more information about the `httpd.props` file, see Appendix A, “The `httpd.props` File”.

NOTE SilverStream also offers a full administration API that Java programmers can use to perform administration tasks programmatically. This API requires users to have the correct access privileges. For more information, see Chapter 13, “Using the Server Administration API”.

Running the SMC

There are several ways to run the SMC.

Using ports The SilverStream server supports separate **runtime**, **design**, and **administration** ports. During installation all three HTTP ports are configured to whatever port number you specified as the default. The default port number is 80 for an NT server and 8080 for a UNIX server. If you have configured separate server ports, you must specify your administration port number when starting the SMC.



For more information, see “About enabling ports” on page 108.

➤ To run the SMC:

- Use any of the following methods:
 - From Windows NT, choose **Programs>SilverStream version>SilverStream Management Console** from the Start menu. You need to update your NT program shortcut used to launch the SMC if you change the port your server is listening on from the port you installed the server on.

OR

- At a command prompt, type the following command:

```
SilverStreamInstallDir\bin\smc
```

The **smc** command can take the following command-line option:

Command-line option	Description
-ss_noconsole	Suppress the Java console at startup
-ss_username <i>username</i>	Login using the specified username
-ss_password <i>password</i>	Login using the specified password
-ss_nosplash	Do not display the SMC splash screen
+Dssw.ssl.nocachecheck	Do not verify self-signed server certificates
-?or -help	Lists options

OR

- From the SilverStream Designer, choose **File>Manage Server**.
If you have configured separate ports, make sure you have first added the server to the Designer using the Administration port.

Creating a secure connection You can establish a secure (SSL) connection between the SMC and the SilverStream server. For information, see “Establishing a secure connection to the server” on page 204.

The SMC properties file The SMC properties file (smc.props located in the Resources directory) contains information about:

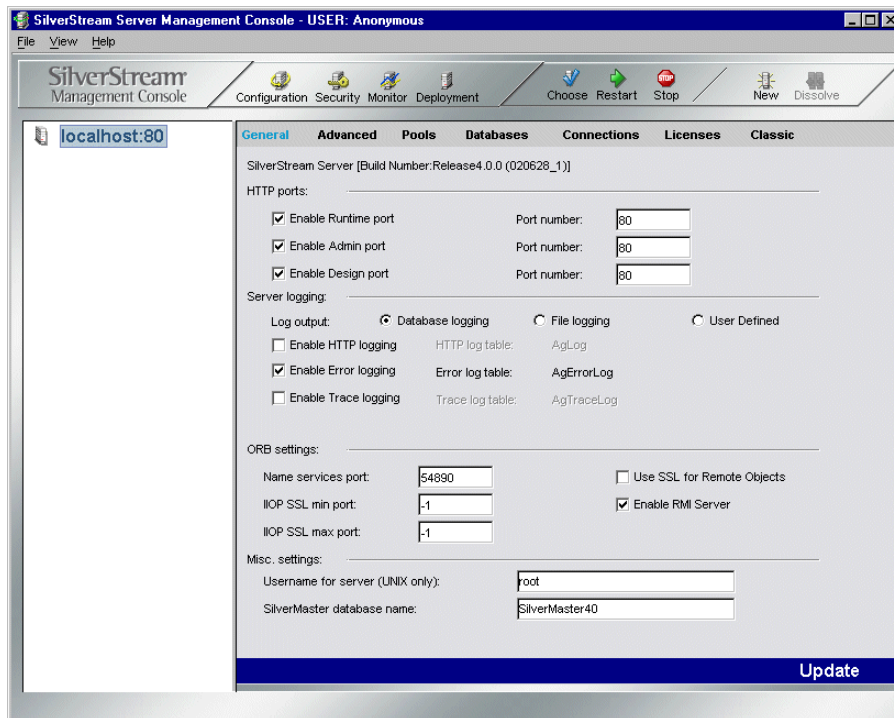
- The list of servers that have been added to the SMC through the SMC
- The property that specifies the settings for charting
- The property that specifies whether to display Classic settings

If you pass a server name on the command line, it is **not** added to the list of servers. If you supply the user name and password on the command line but not the server, the command is ignored (since the SMC cannot determine which server the parameters apply to).

The smc.props file is updated when you use the SMC to make changes to these properties and when you close the SMC. You should not edit smc.props manually while the SMC is running, because none of the changes will be saved.

The SMC user interface

The SMC consists of a series of panels that you can use to administer the server.



NOTE The SMC displays different options if you are running the server in a clustered environment. For more information, see Chapter 12, “Administering a Cluster”.

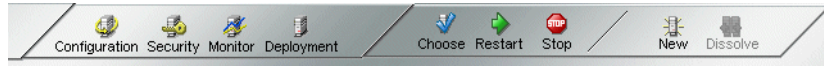
ABout SMC panels

The administrative options are divided into panels, such as General, Advanced, and so on.

 For a quick reference to the SMC panels, see “SMC panels” on page 1.

About the toolbar

The toolbar at the top of the console displays icons that allow you to perform actions.



Icon	Description
Configuration	Provides access to configuration options, such as general server options, database options, and client connection options
Security	Provides access to security options, such as users and groups, the use of user authentication, certificates, and security providers
Monitor	Provides access to charts of server statistics, server logs, and views of statistics
Deployment	Provides access to J2EE objects deployed on the server, the server's JNDI tree, settings for the server's or database's default URL, and listings of RARs deployed to the server.
Choose (server)	Adds a server on your network to administer using the SMC; you can administer multiple servers from one SMC console
Restart (server)	Restarts the selected server after changing parameters
Stop (server)	Shuts down the selected server
New (cluster)	Creates a cluster profile for load balancing
Dissolve (cluster)	Dissolves a load balancing server cluster and deletes the cluster profile (applies to server clustering only)

Menu

The menu at the top of the console provides another way to perform many of the same functions that the toolbar provides. It also lets you perform these additional tasks:

Menu option	Description
File>Login	Allows you to log in to the SMC. See “Logging in” on page 25 for more information.
View>Server console	Displays the server console.
View>Display Classic Settings	Some features of the SMC apply only to classic SilverStream applications (not J2EE applications). These settings are only displayed when this menu item is checked.

Logging in

You must start the SilverStream server before you can log in to the SMC or the SilverStream Designer. If you start the SMC without providing a user name or password, you are connected as Anonymous **only** if the server was installed in unrestricted mode. By default, the SilverStream server is installed in restricted mode. When the server is restricted, which is recommended for production environments, all users need to log in.

If you log in to the SilverStream Designer and then start the SMC from the Designer, you are logged in to the SMC under the same user name.

Separate ports If you have configured separate ports and require user authentication (which is the default), all users will need to log in to (and out of) each port of the Designer or the SMC individually. With separate server ports, you must specify your administration port number when logging in to the SMC. If you add databases in the Designer, you will not see them until you click **Refresh** (or press F5) for each configured port.

➤ To log in:

1. Select **File>Login**.

The Enter Password dialog displays.

NOTE The SilverStream server installs a predefined group named Administrators, which initially contains only the server administrator.

2. Enter your server administrator user name and password, then click **OK**.

Your server administration account name and password is whatever you specified when you installed the SilverStream server. For more information, see “About your administrator account” on page 129.

You now have all administration permissions. The SMC shows the name of the user in its window title.

Logging out

➤ To log out:

1. In the left panel of the SMC, select a server other than the one you started the SMC from.
2. Select **File>Logout**.

You are now connected to that server as Anonymous (as shown in the window title). If you want, you can log back in as a user.

NOTE If you are logged in to multiple ports, you will need to log out of each port individually.

Online help

Here’s how to access the administration documentation in the server’s Core Help:

From here	Do this
SMC	Press F1 or select Help>Help Topics . The Administration Quick Reference displays in your browser. From there you can access the entire <i>Administrator’s Guide</i> and the rest of the Core Help.
Windows	From the Start menu, select Programs>SilverStream eXtend>AppServerN.N>Server Help . Then select <i>Administrator’s Guide</i> in the left (help contents) frame.



For more information, see Using Help and Documentation in the *Essentials* book.

3 Server Configuration

This chapter describes basic hardware configurations for the SilverStream server and explains how the server operates in the Web environment. It contains sections on:

- Server configurations
- Firewalls and proxy servers
- Network configurations
- HTTP server and Web basics
- Session management

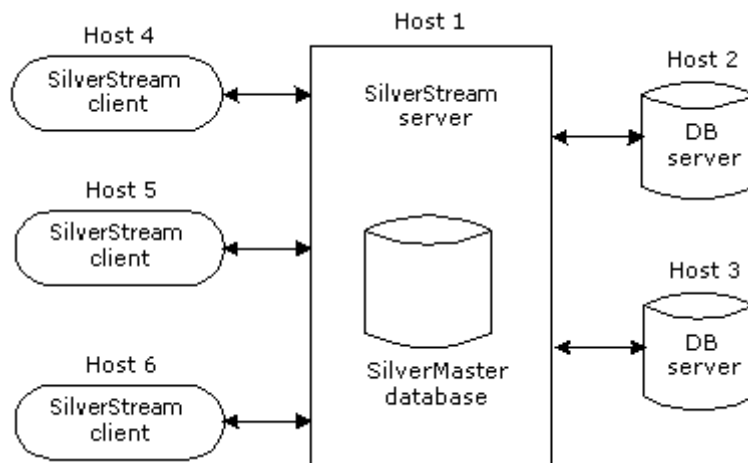
Server configurations

HTTP server configurations differ according to the server environment. This section describes the recommended SilverStream configurations for production and development environments. For simplicity, the descriptions assume a single (standalone) SilverStream server.

Production environment

In a production environment, it is best to configure your SilverStream server and database server(s) on separate machines. (This is called a **multiple-host configuration**.)

The figure below shows the preferred SilverStream server configuration with two database server connections.



The SilverMaster database (shown above with the SilverStream server) is a master database catalog for the entire system and is created when the SilverStream server is installed. For a description of the SilverMaster, see “Configuring the SilverMaster database” on page 58.


NOTE Having another Web server in this configuration would have little impact on the SilverStream server. The SilverStream server can coexist with Web servers as long as you change SilverStream listening port from the default port 80 to another port. For more information, see “Specifying general server properties” on page 110.

Benefits Configuring the SilverStream server and database servers on separate machines results in the following benefits:

- The SilverStream server does not compete with the database servers for CPU and memory resources.
- The machine that hosts each database server can be configured to match that server’s memory requirements.
- Database servers can be optimized and tuned without affecting the SilverStream server.

- You can run your database servers on operating system platforms other than the one the SilverStream server is running on. For example, you can run UNIX database servers and the SilverStream server on Windows NT.

Drawback One drawback to configuring SilverStream servers and database servers on separate machines is that you must maintain extra machines.

 For more detailed information about possible configurations for your production SilverStream environment, see “Network configurations” on page 34.

SilverStream classic development environment

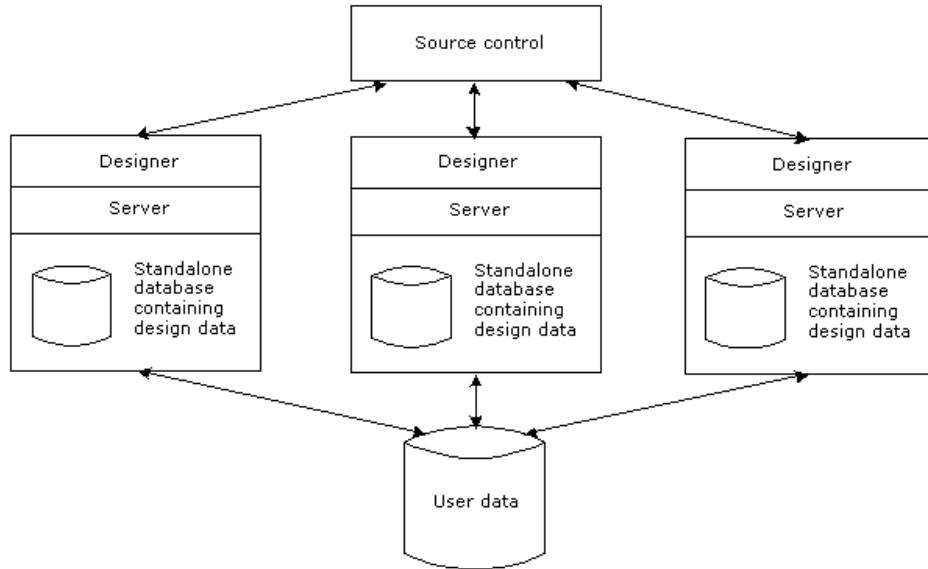
NOTE This section describes the classic SilverStream development environment only. J2EE applications are developed using SilverStream eXtend Workbench. See the SilverStream eXtend Workbench help system for more information.

You can set up your development environment using **multiple independent development environments** (the preferred configuration) or using a **shared development environment**.

Multiple independent development environments


In a multideveloper environment, it is best for each developer to have a SilverStream server and SilverStream Designer client running on their own machine (so that each development environment is independent). Ideally, each developer would also have a standalone database on their machine or their own user account in the database, so that each server can store its own version of the design data. You should also install a source control system to ensure that changes made by developers are not overwritten by other developers.

Preferred classic development environment The figure below shows the preferred development environment configuration. Here each designer communicates with the source control system, while user data (the data used by the application) is accessed from a separate machine.



The chief benefit of configuring one SilverStream server for each Designer is that each developer can have a **sandbox** environment—a space to develop, test, and build applications independently. A developer is not affected by changes made by other developers on a shared server until that developer chooses to get those changes from source control.

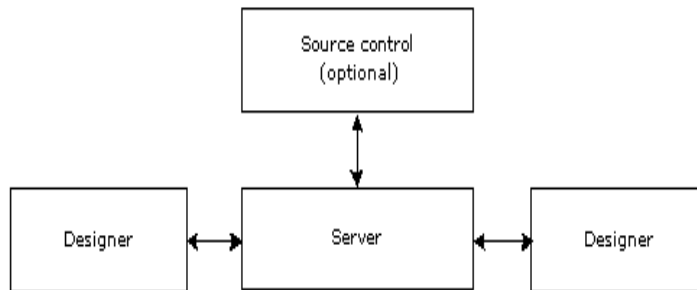
A variation of this configuration is to have on each developer's workstation a standalone database containing the user data; this database matches the production database. This configuration gives each developer a completely independent sandbox environment. But this configuration is not always practical. Sometimes it is not possible to maintain a copy of the production database on a developer's workstation.

 For a description of a development environment coexisting with a production environment, see "SilverStream classic development environment" on page 29.

Shared development environment

In a shared development environment, two or more SilverStream Designers share a single SilverStream server. The database may exist on the same machine as the server, on one of the Designers, or (more commonly) on a separate machine. The single-host configuration is typically used in three situations:

- When you are developing an application in a small group
- When it is impractical for each developer to have their own database access
- When you do not plan to use source control.



In this configuration, there is no sandbox environment—which means that developers cannot necessarily rely on independent application development. Also, debugging a shared server causes the entire server process to be interrupted for all developers.

Firewalls and proxy servers

Firewalls are critical for regulating network access. Before deploying your site you must make many decisions about how you will use firewalls, how the SilverStream server will communicate with database servers, and what if any access you will allow Anonymous users through the firewall.

In a typical large-scale Web environment, a static traffic routing service is placed between the network service provider's router and the internal network. The traffic routing service may be implemented at an IP level using screening rules in a router—or at an application level, using proxy gateways and services.

About proxy servers A **proxy server** is an application that mediates traffic between a protected network and the Internet. Proxy servers are used primarily to consolidate Internet connections, provide users a general level of anonymity (by shielding information normally passed from the browser to the Web server), and enforce enhanced security about Web traffic (for example, what sites users can access).

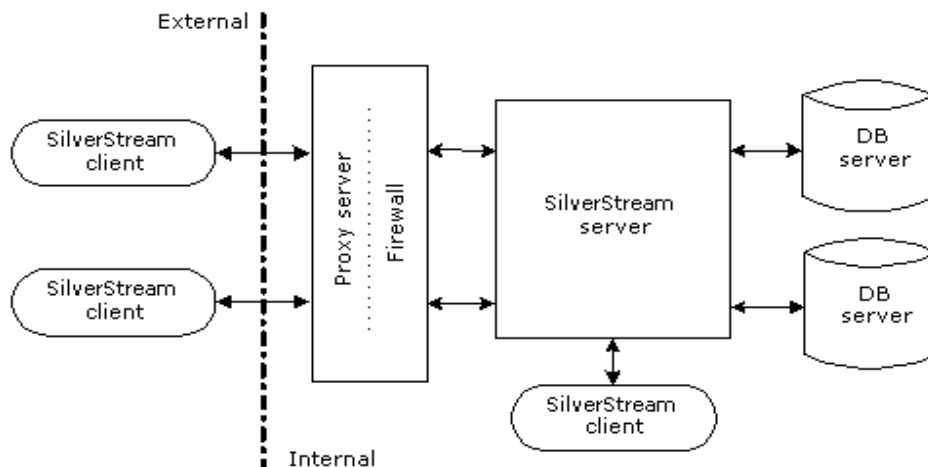
Many proxies contain extra logging or support for user authentication. Since proxies must understand the application protocol being used, they can also implement protocol-specific security. The proxy machine provides a higher level of audit and security, but it also increases configuration costs and reduces the level of service—because a proxy needs to be developed for each desired service.

NOTE The proxy server software you use with the SilverStream server should support HTTP 1.1, such as the Microsoft Proxy Server or Netscape Proxy Server.

About firewalls A **firewall** is a hardware or software facility used to regulate access to a network. Firewalls are traditionally used to protect the company's intranet from the public Internet traffic. **Policies** are configured on the firewall to allow only certain traffic to pass through. The actual mechanism involved varies, but in principle the firewall can be thought of as two mechanisms: one that exists to block traffic and another that exists to permit traffic. Administrators can configure a firewall to notify them of security breaches and monitor overall traffic.

Configuration with a firewall and proxy server

The SilverStream server should run **inside** any firewalls your site has, with the HTTP requests from extranet customers to the SilverStream server either allowed through the firewall or proxied. This way, the database connections need not go through the firewall. The figure below shows how a SilverStream server might be configured with a firewall and proxy server.



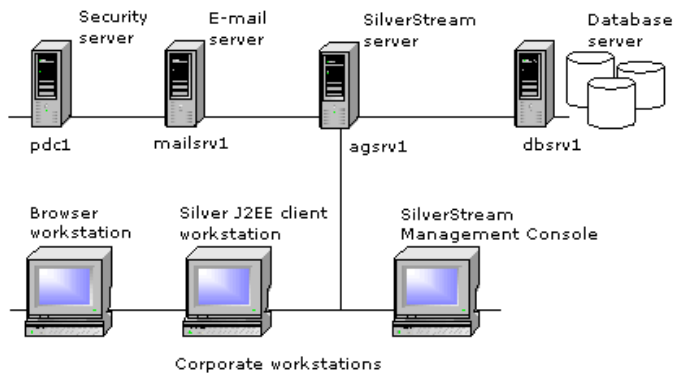
Network configurations

This section presents several possible ways to configure your network, based on your application's needs.

- Simple intranet configuration
- Intranet cluster configuration
- Simple Internet configuration
- Internet cluster configuration
- Demilitarized Zone (DMZ) Internet configuration

Simple intranet configuration

Small companies, departments, and small teams of developers can work against a single SilverStream server. The following figure shows a simple network configuration with the SilverStream server (**agsrv1**) hosting a simple Web application serving users on a local area network (LAN). The application server leverages an existing Windows NT security domain (on **pdc1**) and e-mail server (**mailsrv1**) for user authentication/access control and pushing application data to users via e-mail.



The SilverStream server maintains its master catalog (SilverMaster) in the database server (**dbsrv1**) where the line-of-business database resides. The application database also resides on the database server.

When this configuration makes sense This type of configuration is suitable when:

- The number of users is relatively low (under 50).
- The amount of data returned to the clients is small (for example, a standard departmental application accessing a standard DBMS database).
- Failover capabilities are not required. In some cases it may be acceptable to take the server down for infrequent administrative tasks such as a hardware upgrade or tape backup. So a clustered server arrangement may not be a requirement.
- All users are authenticated against a single, existing security model. A department or site may have a preexisting server listing of users and groups (for example, a Windows NT domain), so this directory can be leveraged and used by the SilverStream server.

Benefits of this configuration This configuration has several benefits:

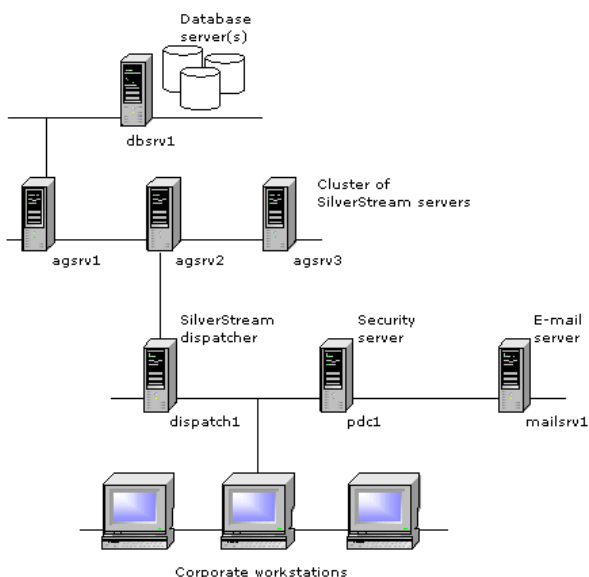
Benefit	Description
Simple administration	Administration of a single application server machine is easier than maintaining a group of computers hosting a cluster of servers.
Simple network topology	Because the number of users is small and the application complexity is low, there is no additional network configuration to make beyond ensuring proper TCP/IP connectivity.

Limitations of this configuration While this configuration may be suitable for small companies or departmental applications, there are some limitations to this approach:

Area	Limitation
Load balancing and failover	This solution offers no provision for maintaining application availability in the event of server downtime. A hardware failure on agsrv1 , for example, would mean that no users could access the application until the problem was resolved.
Internet use	While suitable for smaller intranet applications, this scenario provides no security mechanism for external use by Internet users. No firewall is provided to protect unauthorized access to unsecured LAN resources. And the Intranet security server (pdcl) is not used to authenticate external Internet or extranet users.

Intranet cluster configuration

In order to provide basic load balancing and failover capabilities, the SilverStream server provides a Dispatcher, Load Manager, and Cache Manager. The figure below shows a typical network diagram of several SilverStream servers (**agsrv1**, **agsrv2**, and **agsrv3**) in a cluster with traffic directed by the SilverStream Dispatcher (**dispatch1**). The Cache Manager and Load Manager can reside on virtually any machine in the network, though it is preferable to have them on the same physical subnet as the cluster of SilverStream servers.



In this scenario, a browser on one of the corporate workstations would access the application by connecting to the SilverStream Dispatcher (**dispatch1**) using the Web browser, SilverJRunner, or SilverJ2EEClient. Depending on the load plan, the Dispatcher would reply with an HTTP redirection to one of the available servers in the cluster.

In order to establish a connection, the client needs to resolve the TCP/IP host name of the target server using standard means. On Windows workstations, for example, the client would request the TCP/IP address of the target server from the WINS (Windows Internet Naming Service) or DNS (Domain Naming Service) server, or perform a NBT (NetBIOS over TCP/IP) broadcast to resolve the name and address. Once they are resolved, the client would then access the server directly. No subsequent trips to the Dispatcher would be made.

HTML application example A corporate user opens a browser to `http://dispatch1/Accounting/default.html` in order to log in to the company's accounting HTML application. The SilverStream Dispatcher (**dispatch1**) returns an HTTP redirect signal back to the client, which in turn establishes a connection directly to `http://agsrv2/Accounting/default.html`. Notice that not only was the browser redirected to the SilverStream server (**agsrv2**); the full URL address information (database name, Accounting, and page name) was also passed along.

The next user to access the application would be directed in round-robin fashion to the next available server according to the load plan: `http://dispatch1/Accounting/default.html` would be redirected to `http://agsrv3/Accounting/default.html`.

Java client example A Windows NT workstation launches SilverJRunner with the three parameters **dispatch1 Accounting fmMain**. Once connected to the dispatcher, the SilverJRunner session is redirected to **agsrv1 Accounting fmMain**, according to the load balance plan. As in the previous example, the next user would also be automatically redirected to the next available server, **agsrv3 Accounting fmMain**.

Benefits of this configuration This configuration has several benefits:

Benefit	Description
Server redundancy	In this load-balanced scenario, administrators are free to take down one or even two of the SilverStream servers for maintenance, because the other servers would be available for incoming requests provided that the remaining servers could accommodate the load.
Ease of administration	Setting up the cluster is extremely easy: initial configuration is wizard-based, and every aspect of server administration is done using the SilverStream Management Console (SMC). And there's no additional hardware or software (such as a third-party dispatcher or firewall) required to install and maintain this configuration.
Load balancing	This configuration is flexible: as the number of users grows, the number of servers can expand to accommodate them. The distribution of load across servers means that no one user can cause the server to be a bottleneck for other users in the organization.

Limitations of this configuration This configuration does have limitations:

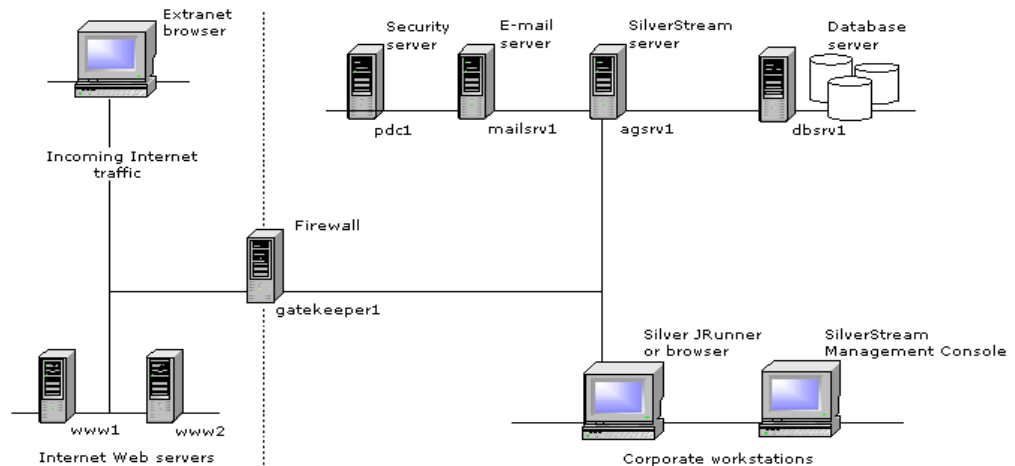
- No firewall or additional security mechanism is provided to protect unauthorized access to unsecured LAN resources.
- While this configuration could be used for Internet applications, there is no provision for DNS masking, such as the kind needed with a more advanced dispatcher. As such, all SilverStream servers and Dispatchers would need to be DNS-registered on the Internet.

To learn more For more information about clusters and load balancing, see Chapter 12, “Administering a Cluster”.

Simple Internet configuration

The figure below shows how a single SilverStream server can be used to provide extranet Web application functionality for both internal users (running a Java application using SilverJRunner or SilverJ2EEClient) and external business partners accessing the HTML application over the Internet.

In this scenario, the SilverStream server (**agsrv1**) provides Web application services in conjunction with existing static content served from the corporate Web site servers (**www1** and **www2**). The SilverStream server (**agsrv1**) is DNS-registered, so when an extranet user is linked from the Web site to the application logon page (hosted on the SilverStream server), the browser knows what route to take in order to connect to the SilverStream server. In this case, Internet clients must pass through the firewall (**gatekeeper1**) in order to gain access to the SilverStream server.



To facilitate this connection, the firewall (**gatekeeper1**) has been configured so that only HTTP traffic on TCP/IP port 80 can pass through to the SilverStream server. This way system administrators are assured that the application-sensitive data will not be intercepted by someone other than the end user, and that incoming traffic cannot access other corporate resources.

The user accesses the application from a link on the corporate Web site (**www1** and **www2**). A SilverStream Web server integration (WSI) module has been installed and configured on both Web servers and offers redirection capabilities to the logon page on **agsrv1**. Once redirected, browsers will establish a connection to the SilverStream server.

This process can be summarized as follows:

1. The user accesses a SilverStream server URL from one of the Web servers outside the firewall.
2. The WSI module responds to the browser with an HTTP redirection to the SilverStream server.
3. The browser automatically requests the URL directly from the SilverStream server, through the firewall.

For user authentication, upon connecting through the firewall to the SilverStream server (**agsrv1**), the user is prompted to log on to the application. A listing of extranet users is maintained in the server's master catalog, the SilverMaster database. This database, like the database serving the e-commerce application, is maintained on **dbsrv1**. The user enters the logon credentials and is logged on to and can commence using the application.

Internal to the company, corporate users interact with extranet users using a Java application that runs on Windows NT workstations and SilverJRunner or SilverJ2EEClient. For administrative and development purposes, corporate IT uses the SilverStream SMC and SilverStream Designer on HTTP port 80.

Benefits of this configuration This configuration has several benefits:

Benefit	Description
Secure e-commerce application	HTTP traffic between extranet users and the SilverStream server can pass safely over the Internet through the firewall. Administrators can log all logon activity either using the firewall or with a logon/logoff business object.
HTML and Java clients	Both groups of users can take advantage of the user interface options from SilverStream: Web and Java. Intranet users, for example, are running Windows NT and Java applications using SilverJRunner or SilverJ2EEClient. Extranet users, on the other hand, are accessing the e-commerce application using a browser client.
Ease of administration	Configuring the SilverStream server for use with the existing network was a simple case of adding a policy to the firewall configuration (for example, allow HTTP traffic to pass to agsrv1 on TCP/IP 80 and log all activity).

Limitation of this configuration This configuration has the following limitation:

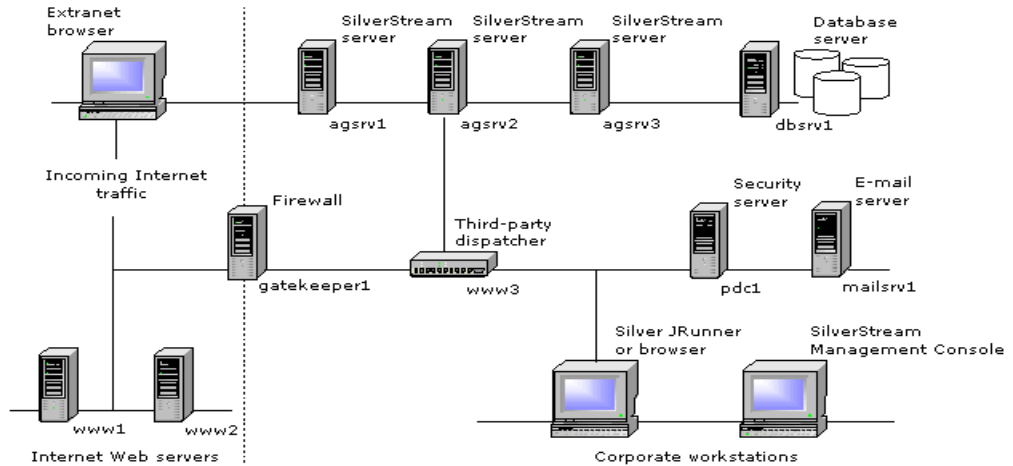
- **Load balancing and failover** Given that users inside the company and external business partners both use this application, the lack of load balancing and failover capabilities means that server downtime results in users not accessing the application.

To learn more For more information about WSI modules, see Chapter 8, “Using the Web Server Integration Modules”.

Internet cluster configuration

Larger-scale e-commerce applications usually require a very high degree of functionality, throughput, and availability. This requires an underlying system architecture that is more robust and complex than those previously shown.

The figure below shows an example of a large-scale Internet application served from a cluster of SilverStream servers. Internet users access the application using links from the two Web servers (**www1** and **www2**) located outside the firewall (**gatekeeper1**).



In order to implement transparent session-level failover and reduce overall DNS and firewall administration, the system administrators install a third-party hardware dispatcher that supports DNS masking, as opposed to using SilverStream Dispatchers. This way traffic to all SilverStream servers can be localized to a single TCP/IP address and host name on the Intranet (**www3**). In addition, with this type of device only one TCP/IP address and host name have to be DNS registered, as opposed to four machines using the SilverStream Dispatcher (**dispatch1**, **agsrv1**, **agsrv2**, and **agsrv3**).

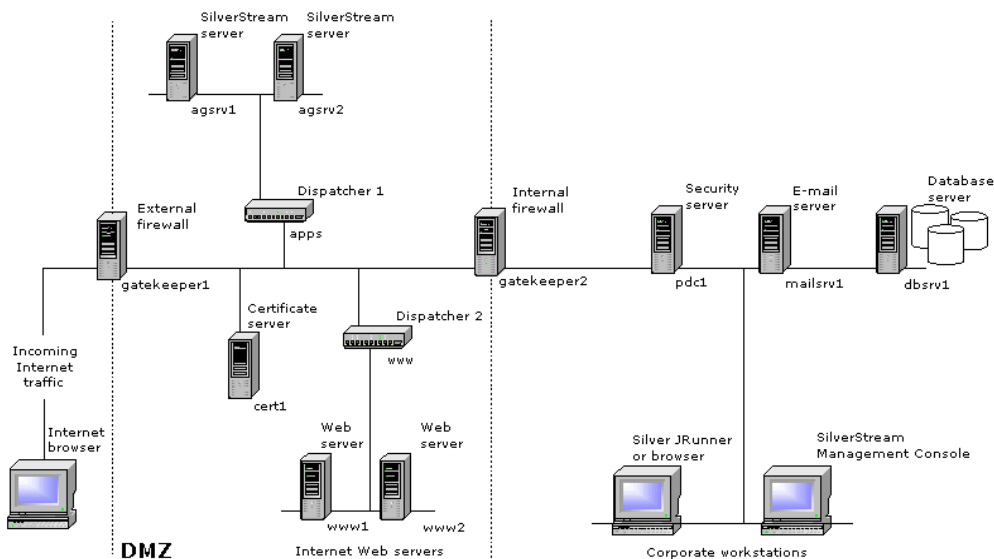
When any incoming requests are linked to the Web application itself (on **www3**), the browser establishes a connection through the firewall to the Web dispatcher. Based on its own load plan, the dispatcher connects the browser to an available server in the cluster. Unlike the SilverStream Dispatcher, the hardware dispatcher controls the flow of all HTTP traffic. In the event that the server goes down, the dispatcher can automatically route the browser session to a different server in the cluster. Since the dispatcher uses DNS masking, the failure is completely transparent to the end user.

To learn more For more information about clusters and load balancing, see Chapter 12, “Administering a Cluster”.

Demilitarized Zone (DMZ) Internet configuration

The complexity of Internet security and network infrastructure is often related to the size of a company. Larger companies with complex e-commerce Internet and extranet applications, for example, may have a two-tiered approach to firewall security.

The following figure shows such an example. All Internet traffic is routed through an Internet firewall (**gatekeeper1**). This firewall allows only Web traffic and Internet mail through to the Demilitarized Zone (DMZ), the area between the two firewalls. All Web and application servers reside in the DMZ for security purposes.



It would have been possible to add a third network card to the firewall and have it protect Intranet traffic as well. However, for security reasons, this company decided to use separate devices.

DNS-masking hardware dispatchers (**www** and **apps**) are used to route traffic in a load-balanced fashion. It is also possible to use one device configured for multiple TCP/IP addresses and route traffic to both clusters. For redundancy purposes, however, two separate devices are used.

The Intranet firewall (**gatekeeper2**) allows e-mail traffic and database connections from the SilverStream servers (**agsrv1** and **agsrv2**) to pass through. This way, the system administrators can be assured that only e-mail traffic and database calls from the secured DMZ (the SilverStream servers) can access corporate information.

External users can be authenticated by obtaining a browser certificate from the certificate server (**cert1**). The SilverStream servers can authenticate these users based on their certificates and encrypt the network traffic from the browser to the application server.

Finally, since there are separate development and deployment environments, business logic has been completely secured in two ways. First, the Java classes that make up the business logic have been **published** from the development to the deployment environments without source code. This is an option developers can use when moving code from one environment to another. Second, security has been placed on the database itself, such that only authorized Intranet users can overwrite or delete the application objects stored in the database.

Benefits of this configuration This configuration is beneficial where there are the following requirements:

Benefit	Description
Security	System administrators have carefully designed this architecture to ensure that traffic from the outside world can only pass into the DMZ. For example, this multitier approach to security allows for tighter control over the origin of database access.
Availability	Server clusters are used for both static Web content and application services. Even the hardware dispatchers provide redundancy among themselves.
Session-level failover	The DNS-masking capabilities of the hardware dispatchers allow for this e-commerce application to run continuously even in the event that a server fails, because the user is automatically rerouted to another server.
High volume	The scalability of a multicluster server arrangement means that user load can be distributed among many servers. This is especially beneficial during peak periods of application usage.

This architecture is complex and more difficult to maintain than the average intranet site. However, you might want to set up this type of architecture to ensure the benefits listed above. Downtime often equates to loss of business. Maintaining a well-designed network infrastructure often pays for itself very quickly.

To learn more For more information about clusters and load balancing, see Chapter 12, “Administering a Cluster”.

HTTP server and Web basics

This section provides an overview of the HTTP communications protocol, describing in some detail how clients communicate with the SilverStream server. It is provided for background information.

Uniform Resource Locators (URLs)

SilverStream clients access server objects through HTTP Uniform Resource Locators (URLs) (except in the case of clients accessing Enterprise JavaBeans or some non-SilverStream clients, which use RMI). The accessed object (resource) can be of any type, from a static HTML page to an executable program. Resources that have URLs can be located and served to browsers regardless of the resource type. URLs are used to obtain information or to place information at a specific location. An URL is composed of four parts:

- **Access protocol** The access protocol specifies the mechanism the browser uses to communicate with the server. SilverStream responds to URLs with the HTTP protocol. The protocol tag is always followed by a colon. For example:
`http:`
- **Host name** The name of the host machine where the resource resides. For example:
`//www.silverstream`
- **Port number** The default port number for HTTP communications is 80 and is normally not specified. When specified, the port number follows the host name and a colon. For example:
`//sun.java.com:80`
- **Access path** The final destination in the address. For example:
`http://www.silverstream.com/products/updates.html`
`http://java.sun.com:80/doc/tutorial.html`


SilverStream resources

What is a resource? A **resource** is a network data object or service that can be identified by an URL. Resources provide an object-oriented mechanism for extending SilverStream server behavior. A SilverStream resource consists of the following parts:

Part	Description
Name	A Uniform Resource Locator (URL)
Resource manager	The Java class that handles HTTP requests (GET, PUT, POST, DELETE, and so on) for its URL
Attributes	An exact set defined by the class—for example: last modified time

Where resource information is stored Resource information is stored in two SilverStream system tables:

Table	Contents
AgResources	The name and location of each resource
AgContents	The actual contents of the resource, which includes the following types: <ul style="list-style-type: none"> • A generic file (static HTML page, GIF, Java class) • An active HTML presentation page • An applet-to-database communication endpoint • A designer communication endpoint • A servlet resource

 For more information about resources, see Appendix C, “SilverStream System Tables and URLs” or “Default group permissions” on page 288.

HTTP communications

HTTP is a request/response protocol used for communications between clients and servers on the Web. The client sends a request to the server. The target server responds to the client only after it receives a request.

HTTP 1.1 uses persistent connections as the default. This means that the client and server maintain connections and send their responses and requests back and forth until the connection is explicitly closed.

Request components The client request has five components:

- **Method** Specifies what the client intends to do with the requested data. Recognized methods include GET, PUT, POST, and DELETE.
- **URL** The location (or target) of the requested resource.
- **Query string** An optional component of the URL that further specifies the request. A query string displays in the URL after the ? character. For example, the following query requests all employees for the month of April:

```
http://localhost/employee/active.html?query=month%30"april"
```
- **Header** This is a required component. The header supplies information about the request or the requestor; for example, the type and version of the client program or the date of the request.
- **Content** For PUT and POST methods, the content is the actual data. If the request is a GET method, the content is empty and will be provided by the response.

Response components The server response has the following components:

Component	Description
URL	The location of the requesting client to which the response is directed.
Status	The message protocol version and a success or error code.
Header	For GET requests, contains a MIME-type , which describes the content of a page that the server is providing to the client. MIME-types include text/plain, text/html, and image/gif.
Content	For GET requests, the information (such as a form or a page) that the client (browser) received from the target URL.

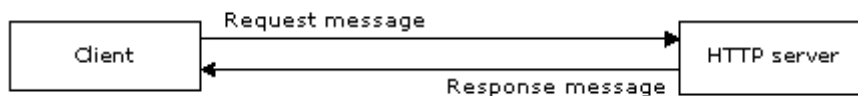
The SilverStream server constructs many of the HTTP components for you automatically. For example, when a SilverStream form requests data from the database, SilverStream constructs the URL for the database resource and submits a GET request to the SilverStream server. The SilverStream server then locates the data and returns it to the requesting form.

Response chains

These are the three general types of HTTP request-response communication scenarios:

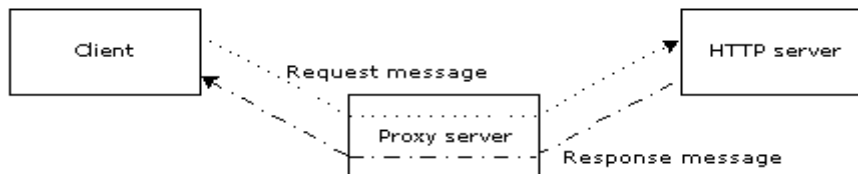
- **User agent (client) directly to server**

This is the simplest and most common scenario in a developer environment.



- **User agent (client) to proxy or cache agent**

This scenario is often used with firewalls where the local-area network (LAN) is isolated from the Internet. A client makes a request to the proxy, then the proxy makes a request to the server on behalf of the client. The server relays to the proxy and the proxy relays to the client.



- **Intermediate caching agent**

Another approach is to use an intermediate caching agent instead of a proxy. (Most proxy agents are also caching agents.) The caching agent tries to serve the request through its internal cache of resources before it forwards the request. The cache itself saves any response it receives, if appropriate. This shortens the response chain, improves response time, and reduces network load.

Session management

The SilverStream server stores information about each client connection in a **session object**. A session is initiated when a client first connects to the server. Information in the session object includes user authentication, SilverStream download information, and database access activity. SilverStream applications can also store application-specific data in the session object.

The SilverStream server can use either cookies or URL rewriting to keep track of the state of multiple Web browser clients. Both cookies and URL rewriting use session IDs. All calls to the server within a browser session will operate under the same session ID. For secure data, authentication occurs once per active session for sessions requiring user authentication.

Authentication is the process through which the server and client verify one another's identities. Authentication is described in "Enabling authentication" on page 266.

SilverStream sessions containing forms and views use SilverJRunner or SilverJ2EEClient (which pings the server) to keep the session alive. If a session containing a page is idle for more than five minutes (the default), it is terminated. Pages are run from the browser, which does not automatically keep the session alive.

Cookies

The SilverStream server uses cookies to track sessions if the user's browser supports them. A **cookie** is a piece of information sent by a Web server to a Web browser. The browser client stores the cookie and sends it back to the server whenever an additional request is made to the server. The SilverStream server uses the cookie as a session ID. When the server receives a request from a client that includes a cookie, it is able to use the information stored in the cookie to reconnect with the session.


Important: SilverStream server cookies are kept in memory and are never written to disk. There is no personal user or tracking information in the cookie.

If you or your users are concerned about the contents of cookies, you can set your browser not to accept cookies or to warn on cookies. See your browser documentation for details about cookies.

NOTE The SilverStream server uses cookies to track sessions for SilverJRunner applications (such as the SMC).

URL rewriting

To support browsers that do not accept cookies, the SilverStream server rewrites URLs (with an appended **jsessionId** parameter) to correctly associate a request with a session. SilverStream application developers writing servlets or JSP pages need to understand how to use URL rewriting to support clients that do not accept cookies.

 For information on URL rewriting for servlets and JSP pages, see “How session tracking works” next.

How session tracking works

The SilverStream server uses **cookies** if the user’s browser supports them and uses **URL rewriting** if the browser does not. This determination happens at runtime for each user. The first time the SilverStream server receives a request, it sets a cookie—and also appends a **jsessionId** to the URL (because it does not yet know whether or not the client supports cookies).

When a client supports cookies, the SilverStream server will use them for session tracking (although it will rewrite the URL when it receives the **first** request). Once the client returns a cookie, the server will stop rewriting URLs for the client in this session.

NOTE Cookies use the value **JSESSIONID** (all uppercase); URL rewriting uses **jsessionId** (all lowercase).

Administrator notes If the server determines that the client does not accept cookies, it uses the **jsessionId** in the URL for session tracking whenever the user clicks a link that is contained on a page.

The first time a client establishes a session, the URL **jsessionId** is appended to the URL and is visible to the client user. On subsequent interactions between server and client, the URL rewriting keeps track of the session ID, and the **jsessionId** is only visible when a user’s mouse is held over a link on the page.

When a browser client does not accept cookies, servlets and JSP pages that use HTML links need to call one of two standard encode methods (described in “Developer notes” next) to rewrite URLs.

Developer notes The following encode methods enable the server to check for the existence of either a cookie or a **jsessionId**:

- `HttpServletResponse.encodeURL()`
- `HttpServletResponse.encodeRedirectURL()`

One of these encode methods is needed when:

- Servlets or JSP pages explicitly create or embed URLs in their responses
- Pages created via the SilverStream Designer use embedded links in custom page controls or in JavaScript

NOTE Any pages created via the SilverStream Designer automatically call the correct session management methods, provided you are **not** using embedded links in custom page controls or in JavaScript.

The **jsessionid** in the URL is a path parameter, not a query parameter. **Query parameters** are usually at the end of the URL and separated by a **?**. **Path parameters** are at the end of a component of the URL and before any query parameters. In the following example:

```
http://server/db/foo;pparam=foo?qparam=bar&rparam=bar
```

pparam is a path parameter and **qparam** and **rparam** are both query parameters.

4

Data Source Configuration

This chapter describes the basic data source configurations for the SilverStream server and how to set them up. Topics include:

- About data source configurations
- Preparing a database for access
- Configuring the SilverMaster database
- Configuring classic application databases
- Configuring connection pools

About data source configurations

The SilverStream server uses relational databases as a repository for deployed applications.

- For J2EE applications, J2EE archives are deployed to a database.
- For classic SilverStream applications, forms, pages, and business objects are stored in a database.

The SilverStream server provides connectivity to relational databases (and EIS systems for J2EE applications) so that the deployed applications can retrieve and update corporate data.

To support deployment and data access, you need to configure server access to the data tier. The server configuration depends primarily on the applications (J2EE or classic) deployed on your server. This section provides an overview data source configuration and includes these topics:

- J2EE configuration
- Classic SilverStream configuration
- Data source configuration tasks

J2EE configuration

If your environment includes J2EE applications, you'll need to configure:

- The SilverMaster database (described in “Configuring the SilverMaster database” on page 58).
- A relational database where J2EE applications are deployed. (SilverStream recommends that you deploy your J2EE applications to the SilverMaster database.)

- One or more connection pools that J2EE applications will use to retrieve and update corporate data. The connection pools can represent relational databases or EIS systems. J2EE applications (such as WARs, EARs, and EJB JARs) define their data sources as resource references in the deployment descriptor. When the archive is deployed to the server, the deployer will use deployment tools to map the resource reference to an actual data source that is available to the server. As the administrator, you'll be responsible for making sure that the data source is available to the server and that the server has the appropriate access (read/write access) to the data source. You make a data source available to a J2EE application by creating a **connection pool**. You can create a connection pool using the SMC or SilverCmd as described in “Configuring connection pools” on page 69

Classic SilverStream configuration

If your environment includes classic SilverStream applications (forms, pages, and business objects), you'll need to:

- Configure the SilverMaster database (described in “Configuring the SilverMaster database” on page 58).
- Add one or more relational databases where the applications will be deployed (or published), and any databases containing the user data.

When you add a database, the SilverStream server adds system tables to it. SilverStream uses these system tables to store your application code, Java classes, SilverStream metadata, and so on. For a listing of these tables, see Appendix C, “SilverStream System Tables and URLs”. These are ordinary database tables reserved for SilverStream use, and they can be added to your application database or stored in a separate database. If you store them separately, remember to back them up periodically.

The databases must be supported by the SilverStream server. A **supported database** is a database that can be recognized by, and can interact with, the SilverStream server.

You add a database to SilverStream using the SMC or SilverCmd as described in “Configuring classic application databases” on page 59.



See the *Release Notes* for the complete list of supported databases.

Data source configuration tasks

The following table provides the list of data source configuration tasks that you'll need to perform depending on the type of applications deployed to your system.



Application type	data source type support	Administrative tasks
J2EE	Relational databases	<ul style="list-style-type: none"> • Install a JDBC driver on the server • Create a user ID and password for the SilverStream server to use • Create a connection pool
	EIS	<ul style="list-style-type: none"> • Deploy RAR on the server • Create a user ID and password for the SilverStream server to use • Create a connection pool
Classic SilverStream	Relational databases	<ul style="list-style-type: none"> • Install JDBC driver on server • Create a user ID and password for the SilverStream server to use • Add the application database to the server


Preparing a database for access

SilverStream can access an existing database containing data and any new databases that you create. Creating new databases on most DBMSs in corporate environments is a function done by a DBA (database administrator). If you are in such an installation you may require the DBA's assistance creating new databases or changing user access rights for SilverStream to access these databases. If you are running a DBMS locally, you will be able to accomplish these DBA tasks without assistance.

NOTE You should not use **SilverStream** as the name of your database.

The following table describes the general tasks that you need to perform to set up a database that SilverStream can access.


Task	Who	How
Set up a database user account that the SilverStream server will use to connect to the database.	Your DBA if you have one.	DBMS utility for adding and modifying database account permissions (for example, Sybase Central, Microsoft Enterprise Manager, Oracle Server Manager, Informix Control Center, and so on).  For more information, see “Setting up database accounts” on page 54.
Set up an ODBC data source for any new database and any existing databases you want SilverStream to use through an ODBC bridge.	A system administrator, if set up as a system resource. Otherwise, by a developer.	ODBC control panel (Windows only; ODBC connections are currently not supported on UNIX).  For more information, see the ODBC chapter in the <i>Installation Guide</i> .
Configure the DBMS client software on the SilverStream server machine.	Your DBA, if you have one.	Use native database software (such as Oracle SQL-Net, Microsoft SQL Server client, Informix CLI, and so on).
Install the JDBC driver.	DBA or system administrator, if you have one.	Use native DBMS installer to install a JDBC driver on the SilverStream server machine (for example, jConnect).

 For information about how to set up your specific database type for use as a SilverMaster or as an application database with SilverStream, see the appropriate configuration chapter in the *Installation Guide*.

Setting up database accounts Once you have created the database, you need to set up a database account for SilverStream to use to connect to each database. All database access in an application is through the SilverStream server; clients do not access the databases directly.

You should set up a different account for your SilverMaster and for each application database so you can easily tell which applications are hitting your database when and how often and you can more easily troubleshoot performance problems.

When setting up a database, make sure the user account (such as Agsmith) has CREATE TABLE, INSERT, UPDATE, and DELETE permissions.

 For information about other types of accounts, see “Administration accounts” on page 417.

Database access

SilverStream can access databases through a native Java Database Connectivity (JDBC) driver or through a JDBC-ODBC bridge driver.

Java Database Connectivity (JDBC)

JDBC is a standard Application Program Interface (API) for allowing Java applications such as the SilverStream server to enable SQL access to relational databases. The application makes JDBC calls to the JDBC driver, which translates the calls to the API of the underlying database. JDBC drivers exist for a number of commonly used relational databases. The Java runtime system supplies an ODBC bridge driver, which allows JDBC to connect to supported databases through an ODBC driver.

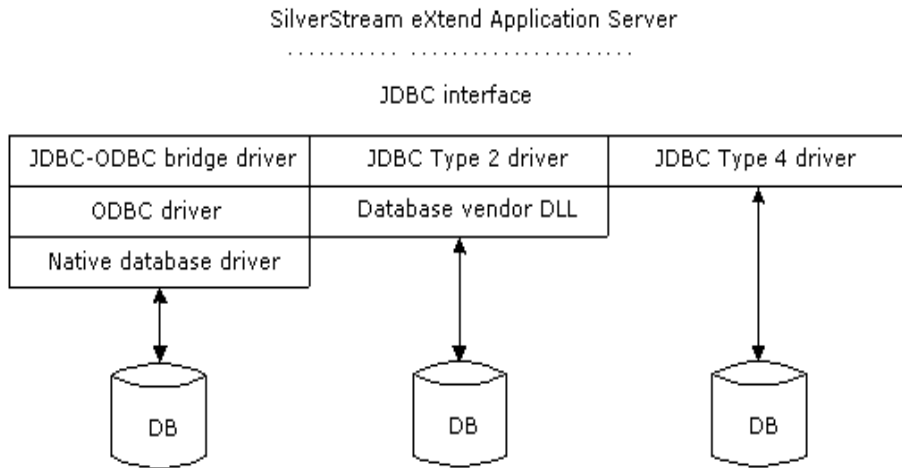
JDBC access

There are four types of JDBC drivers.

JDBC driver	Description
Type 1: JDBC-ODBC bridge	For databases that support ODBC.
Type 2: JDBC to a database vendor DLL	Supplied by the vendor or by third parties.
Type 3: JDBC to middleware software to the database	Not recommended for use with SilverStream.

JDBC driver	Description
Type 4: Pure Java to a network protocol	This is the best driver to use with SilverStream, because it works directly with the network protocol. However, only a few databases currently have Type 4 drivers available.

The following diagram shows the components of each supported JDBC driver type.



Adding JDBC driver JARs to the server classpath

To access a database via JDBC, the SilverStream server must be able to find the JAR files for the appropriate JDBC driver. That means those JARs must be added to the server's classpath. How this is done depends on the kind of database access you are setting up:

- For SilverMaster
- For any other database access

This section presents general guidelines for these setup tasks. To get the details for your DBMS, see the corresponding database configuration chapter in the SilverStream server's *Installation Guide*.

For SilverMaster

When you're setting up access to the SilverMaster database, adding JDBC driver JARs to the SilverStream server's classpath involves the following:

Platform	How JARs are added to server's classpath
Windows	Before you install the SilverStream server, you must manually set up the system environment variable AGCLASSPATH to list the required JDBC driver JAR files.
UNIX	When you run the installation program for the SilverStream server, it automatically prompts for the location of your JDBC driver JAR files. Then it edits the .agprofile file (in your SilverStream root directory) to set the AGCLASSPATH variable with that JAR information.

For any other database access

When you're setting up any other database access (other than for SilverMaster) and a different database driver is involved, you must manually add those JDBC driver JAR files to the SilverStream server's classpath. For instance, this might occur if you're accessing DB2 for SilverMaster but also need to establish connection pools for Oracle.

The typical way to do this is:

- **In Windows**, edit the system environment variable AGCLASSPATH to add the required JARs
- **In UNIX**, edit the .agprofile file (in your SilverStream root directory) to set the AGCLASSPATH variable with that JAR information

Testing your database connections

As you set up database access for the SilverStream server, it's recommended that you test each connection first using the tools provided by your database vendor. Knowing that those connections are valid can save time later if you need to troubleshoot access from the SilverStream server.

Configuring the SilverMaster database

SilverStream relies on a master database catalog called the SilverMaster for overall system management. This database is created at installation. You can use any supported database to create the SilverMaster catalog, as long as it supports autoincrementing columns.

SilverMaster functions


The SilverMaster catalog provides overall system management, including the following:

- Storing any SilverStream information that spans all databases
- Maintaining a catalog of databases managed by the server and tracking any new database added to the server
- Tracking user and group authentication information
- Managing shared SilverStream data such as images and prebuilt SilverStream subforms (such as the navigation buttons used by default on some forms)
- Managing SilverStream runtime classes (either ZIP or JAR files)
- Storing cluster information
- Storing license information
- Storing deployed J2EE archives

The SilverMaster catalog contains internal SilverStream tables used for system management. These tables are reserved for SilverStream use. For a list of these tables, see Appendix C, “SilverStream System Tables and URLs”.

NOTE For classic applications, SilverStream advises against storing unrelated (user data) tables in the SilverMaster catalog.

Default SilverMaster permissions After a default installation, users have Read access to the top level of the SilverMaster database and directories, which will enable them to log in and access any existing application databases. Users cannot add or remove any databases nor access any application objects (such as pages, forms, and views) until you grant them permission. If you have configured separate ports for different types of operations, you must use your administration port to update database configuration.

 For information about troubleshooting the SilverMaster database, see “Using the SilverMasterInit program” on page 416.

Moving the SilverMaster database

If you move your SilverMaster database from the server it was initially installed on, the SilverStream server needs to know the new connection location. For example, if you decide to upgrade your server, you may want to move your SilverMaster to the new machine. The easiest way to update SilverStream about a moved SilverMaster is to rerun the SilverStream installation program after you have moved your SilverMaster database and modified settings specific to your DBMS.

NOTE Some databases require that you update connection parameters (for example, by using ODBC, JDBC, or Oracle TNS).

In the SilverStream installation program, specify the moved SilverMaster database as you respond to the prompts. Be sure to choose the option **Install a new Server configuration file** from the screen that runs SilverMasterInit. It is not necessary to run SilverMasterInit. If you run SilverMasterInit to initialize the SilverMaster properties, you will have to rerun the license install, recreate any SilverStream users/groups, and manually add your SilverStream application databases. It is a good idea to verify the new SilverMaster database connection (using another application) before restarting the SilverStream server.

Configuring classic application databases

This section describes what you need to do to configure a relational database for use by classic SilverStream applications:

- Adding a database to the server
- Removing a database from the server
- Configuring a database



For information on setting up a database for use with SilverStream, see the *Installation Guide*.

Adding a database to the server

When you add a database to the SilverStream server, an entry is made in the SilverMaster database catalog so that SilverStream knows about the database.

Once you have created the database and configured it as described in the *Installation Guide*, you simply add it to the SilverStream server.

NOTE When adding an application database on a UNIX platform, you must add the location of the database to the AGCLASSPATH environment variable, then restart the server before you can add the database. For information about AGCLASSPATH, see “Setting the AGCLASSPATH variable” on page 138.

Storing system tables When adding a database to the server, you can optionally store the system tables in a separate database than your application (data) tables. (But note that you **cannot** store the system tables in the SilverMaster database.)

If you choose to store the system tables separately, just remember to back them up. Storing the system tables separately is a useful option if you already have other applications accessing a production database and you do not want to change the database structure, or if the SilverStream server is accessing multiple databases. If you choose to store your SilverStream system tables separately from your application’s tables, you should set up unique database accounts for each of these shadow databases.

Adding databases in a restricted production environment If your SilverStream server is running in a restricted production environment (as opposed to an unrestricted design environment), you will need to authenticate yourself before adding (or deleting) a database. In a restricted environment, no users (except the server administrator) can add databases until you grant them permission.

➤ **To add a database to a SilverStream server:**

1. Start your server.
2. Start the SMC.
3. Select the server in the left pane of the SMC. If the server is not listed, add it to the SMC, as described in “Administering a SilverStream server remotely” on page 135.
4. Select the **Configuration** icon from the toolbar.
5. Select **Databases**.
6. Click **Add Database**.

You are prompted to enter information about the database.

7. Use the table below to enter the information for the database. If you need help, see the chapter that covers your DBMS in the *Installation Guide*.

Field	What to specify
Name of the database	Enter the name of the database. For an ODBC database, the database name must be an already existing ODBC data source name.
User name and password	<p>Enter a user name and password pair that the SilverStream server can use for a database user connection to your native database. These values cannot be null.</p> <p>This user name must already be known to the native database and have the appropriate read/write permissions.</p> <p>NOTE Your administrator should have defined a unique user for each application database (all users can have the same password). For more information, see the Chapter 4, “Data Source Configuration”.</p>
Database platform	Choose from the list of supported database platforms.

Field	What to specify
Driver set	<p>Choose the driver set to use from the list (a driver set is a JDBC driver, sometimes in combination with SilverStream-specific files).</p> <p>The listed driver sets are specific to the database platform you selected. The driver set recommended for your database type is displayed by default.</p> <p>If you select a non-SilverStream driver set (a driver set whose name does not begin with SilverStream), see “Using non-SilverStream driver sets” on page 63.</p>
Store system tables separately from data tables	<p>If you plan to use a production database that other applications access, you may not want to add SilverStream system tables to it. The system tables contain the objects that comprise your application.</p> <p>This option allows you to store the SilverStream system tables in another database. If you check this option and click Next, a panel displays asking you to name the system table database. (This database must already exist.)</p>
Include only a subset of tables	<p>You may not want to use all the tables in the database you are adding. By selecting this option, you can specify a subset of tables to make available on the server.</p> <p>If you select this option and click Next, a panel displays with two list boxes. In the top box you can manually name each table you want to use. In the lower box you can specify patterns to indicate sets of tables. For example, you could specify cust% to use all tables starting with cust.</p> <p>You can later change which tables are available, using the <code>ModifyTableList SilverCmd</code>.</p>

8. Click **Finish**.

The database is added to the server.

Using non-SilverStream driver sets If you are not using a SilverStream driver set, you must supply additional information.

The following describes each of the fields on this panel.

Field	What to specify
JDBC Driver	<p>(Read-only) The fully qualified name of your JDBC driver class. For example:</p> <pre>com.sybase.jdbc.SybDriver</pre> <p>NOTE Package names, like all Java names, are case sensitive.</p>

Field	What to specify
JDBC URL	<p>The URL string defined by the driver vendor to connect to your database. The string contains replaceable parameters surrounded by percent signs (%), such as %HOST%.</p> <p>For example:</p> <pre>jdbc:sybase:Tds:%HOST%:%PORT%/%DATABASE%</pre> <p>Substitute these parameters with values appropriate to your database.</p>
JDBC URL attributes	<p>Any additional URL attributes defined by the vendor that you can use to customize the driver connection. For example:</p> <pre>cache=100</pre> <p>Leave this field empty for DB2 databases.</p>



See your JDBC driver documentation for more details.

What happens

When you add a database, SilverStream adds an entry in the SilverMaster database and also adds SilverStream system tables to your database (unless you specified to keep system tables separate from your data, in which case the system tables are added to the other database).

Storing system tables separately from the data

If you are storing the SilverStream system tables separately from the data, SilverStream will store the classic application components (like forms, pages, and views) in the specified system database.

After adding the database, SilverStream manages the system database transparently. You work only with the main database. For example, if you added Database A to the server and specified to store the system tables in Database B, when you are ready to deploy your classic application, you deploy it to Database A.

Adding a database from the command line

You can also add a database to the server from the command line or from a batch file using the `AddDatabase SilverCmd`.

Deploying a J2EE application

When you are ready to deploy your J2EE application, see the chapter on J2EE Archive Deployment in the *Facilities Guide*.

Moving an added database

If you have moved a database that you had added to the SilverStream server, remove the database from the server and then re-add it to the server.

- If you are using ODBC, you may also need to update the database's ODBC settings.
- If you are using a JDBC driver (such as jConnect), you will have to update the JDBC URL when adding the database back to the server.

Removing a database from the server

If you don't need to maintain a connection between a database and the SilverStream server, you can remove the database from the server.

➤ To remove a database from the server:

1. Start your server.
2. Start the SMC.
3. Select the server in the left pane of the SMC. If the server is not listed, add it to the SMC, as described in "Administering a SilverStream server remotely" on page 135.
4. Select the **Configuration** icon from the toolbar.
5. Select **Databases**.
6. Select the database in the **Database settings** field.
7. Click **Remove Database**.
You are asked to confirm the action.
8. Click **OK** to remove the database from the server.

What happens When you remove a database connection from the server, SilverStream removes the entry from SilverMaster but leaves the database itself fully intact (including the SilverStream system tables).

Removing a database from the command line You can also remove a database from the server from the command line or from a batch file using the RemoveDatabase SilverCmd.

Configuring a database

You can use the SMC to configure databases that have been added to a server. For example, you can use the SMC to synchronize database information, delete idle connections, and get information about the separate database containing SilverStream system tables (if you chose to store the SilverStream system tables in a different database than your application database). If you have configured separate ports for different types of operations, you must use your administration port to update database configuration.

➤ **To configure a database:**


1. Start the SMC.



NOTE If you have configured separate ports for different types of users and operations, you must specify your **administration** port to start the SMC.

2. Select the **Configuration** icon from the toolbar.
3. Select **Databases**.
4. Select the database name from the dropdown list.

The screenshot shows the 'Databases' tab in the SMC configuration interface. At the top, there are tabs for 'General', 'Advanced', 'Pools', 'Databases', 'Connections', and 'Licenses'. Below the tabs, there are two rows of settings for 'Min # of database connections' and 'Max # of database connections', both set to 2 and 10 respectively, with 'Reset all' buttons. An 'Add Database...' button is located below these settings. The 'Database settings:' section includes a 'Database:' dropdown menu set to 'SilverMaster40'. Below this are 'Username:' (set to 'jdba') and 'Password:' (masked with asterisks) fields. There are also 'Min Connections:' (set to 2) and 'Max Connections:' (set to 10) fields, each with a 'Reset' button. A 'Remove Database...' button is positioned to the right of these fields. The 'Database platform:' is 'Sybase Adaptive Server Anywhere 6', and the 'Driver set:' is 'SilverStream JDBC-ODBC bridge'. The 'JDBC Driver:' is 'com.sssw.jdbc.mss.odbc.AgOdbcDriver' and the 'JDBC URL:' is 'jdbc:sssw:odbc:SilverMaster40'. The 'JDBC URL attributes:' field is empty. At the bottom, there are 'Synchronize Database Schema' and 'Delete Idle Connections' buttons. A large blue 'Update' button is at the bottom right.

5. Enter information for the database as follows.

Field	Description
User name and password	<p>A user name and password pair, which the SilverStream server can use for a database user connection to your database.</p> <p>This user name must already be known to the database and have the appropriate Read/Write permissions.</p>
Database platform	(Read-only) The DBMS.
Driver set	(Read-only) The combination of a JDBC driver and SilverStream-specific files to use to connect to the database.
JDBC driver	(Read-only) The fully qualified name of the JDBC driver to use when connecting to the database.
JDBC URL	(Read-only) The fully qualified package name of your JDBC driver class (the database URL). The driver uses it to connect to the database. The URL is driver-specific. See your driver documentation for more information.
JDBC URL attributes	(Read-only) Any extra attributes to set for the driver. The syntax is driver-specific. See your driver documentation for more information.
Synchronize Database Schema	<p>A button that lets you synchronize the database. SilverStream may not be the only tool you use to modify your database's schema. For example, you may be using Sybase Central to modify your Sybase Adaptive Server Anywhere database structure outside SilverStream. SilverStream keeps its own image of the database schema.</p> <p>Click this button to update an existing SilverStream database structure with any structure changes that may have occurred through another tool.</p> <p> For more information about this option, see "Synchronizing the database schema" on page 68.</p>

Field	Description
Delete Idle Connections	<p>Release database connections that are not currently being used. When the server needs more connections, the connection pool will automatically regrow as needed to the maximum number of connections defined. Deleting idle connections allows you to (at least temporarily) free up some database connections for use by other applications without having to restart the server.</p> <p> For information on permanently changing the pool size, see “Setting the maximum and minimum number of database connections” on page 320.</p>
System Database Properties	<p>A button that displays only if the selected database is storing its SilverStream system tables in another database (you specify this property when adding a database to a server).</p> <p>Click this button to see information about the database storing SilverStream system tables for the selected database on the server.</p> <p> For more information about storing system tables externally, see the Main Designer chapter in the <i>Tools Guide</i> of the server’s Classic Development Help or see AddDatabase in the SilverCmd Reference in the <i>Facilities Guide</i> of the SilverStream server’s Core Help.</p>

Synchronizing the database schema You may need to synchronize the server metadata and the current database schema to ensure that tables, views, and key definitions cached on the server match the current database structure. The server metadata and the current database schema most commonly get out of sync when changes are made to the database via tools other than the SilverStream Designer.

When the server starts up, it checks for database consistency including these types of changes:

- The column count in a table does not match
- Column types do not match
- A new table or database view has been added
- A foreign key has been added
- A table, view, or foreign key has been dropped

NOTE You can prevent the server from checking database consistency with the **-nodbcheck** command-line option. You can also use the **-noexitondbcheck** command-line option to see any errors while still starting the server. If you see any errors, you should synchronize the database. For more information, see “Database not synchronized” on page 414.

When it receives a request to synchronize the database, the server:

- Generates the current database schema
- Synchronizes the generated schema with the cached schema
- Sends a response back to the client if it was successful or sends an exception if it failed

Configuring connection pools

The SilverStream server uses connection pools to allow J2EE applications to access data in relational databases (via JDBC) or one or more EIS (via a Resource Adapter deployed to the server). You make a relational database available to SilverStream by installing the appropriate JDBC driver for the DBMS and creating a JDBC connection pool for that database, and you make an EIS available to SilverStream by deploying the RAR to the server and creating a Connector connection pool. This section provides the following information:

- Adding a JDBC connection pool
- Adding a Connector connection pool
- Removing a connection pool

Adding a JDBC connection pool

This section describes how to add JDBC connection pools using the SMC’s Add JDBC Connection Pool Wizard. When you create a JDBC connection pool, you must have a JDBC driver installed on your system; SilverStream supports both JDBC 1.0 and JDBC 2.0 drivers.

This section provides the following topics.

- Panel sequence
- Starting the Add JDBC Connection Pool Wizard
- Panel reference

Panel sequence

The wizard panels and the order in which they are presented by the wizard vary depending on the type of JDBC driver you are using to access the target database. This table shows how you step through the wizard based on the type of JDBC connection pool you want the wizard to build. You can click the links to get more information about the values you must supply for each panel.

If you want to create a JDBC connection pool	You'll be responsible for
<p>For JDBC drivers for which a SilverStream Logical Data Source (LDS) key exists.</p> <p>NOTE A driver with an LDS key is a driver for which SilverStream provides a higher level of service. For example, if SilverStream has an LDS key for a driver then SilverStream knows how to handle error codes returned by the driver and can also work around bugs in the driver.</p>	<ol style="list-style-type: none">1. Starting the Add JDBC Connection Pool Wizard2. Specifying LDS or a user-specified driver3. Specifying the pool name4. Specifying the database platform5. Specifying the JDBC driver and URL6. Specifying data source configuration properties7. Specifying connection and timeout properties

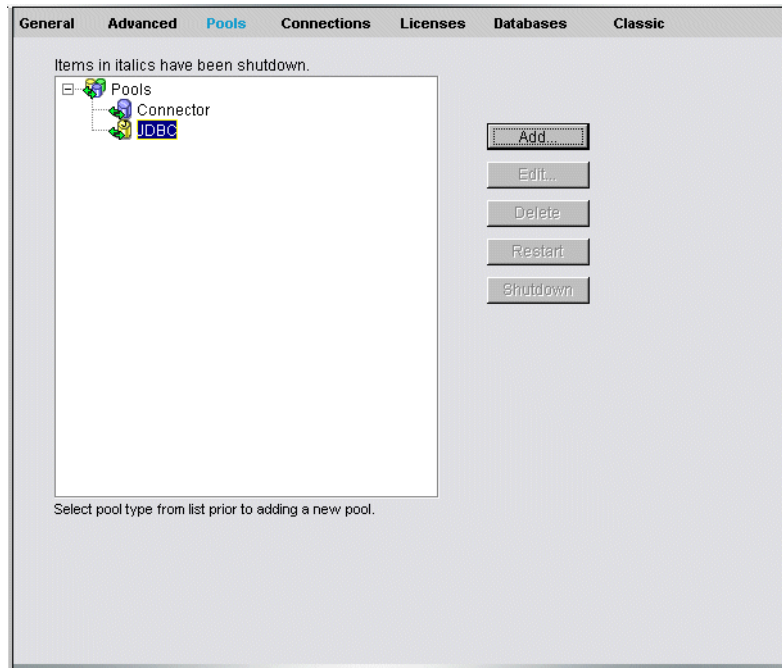
If you want to create a JDBC connection pool	You'll be responsible for
For JDBC 1.0 user-defined drivers	<ol style="list-style-type: none"> 1. Starting the Add JDBC Connection Pool Wizard 2. Specifying LDS or a user-specified driver 3. Specifying the JDBC version 4. Specifying the pool name 5. Specifying the JDBC driver and URL 6. Specifying data source configuration properties 7. Specifying connection and timeout properties
For JDBC 2.0 user-defined drivers	<ol style="list-style-type: none"> 1. Starting the Add JDBC Connection Pool Wizard 2. Specifying LDS or a user-specified driver 3. Specifying the JDBC version 4. Specifying the XA data source information 5. Specifying the pool name 6. Specifying data source configuration properties 7. Specifying connection and timeout properties


Starting the Add JDBC Connection Pool Wizard

➤ To start the Add JDBC Connection Pool Wizard:

1. Start your server.
2. Start the SMC.
3. Select the server in the left pane of the SMC. If the server is not listed, add it to the SMC, as described in “Administering a SilverStream server remotely” on page 135.

4. Select the **Configuration** icon from the toolbar.
5. Select **Pools**.
6. Choose **JDBC** and click **Add**.



 See “Panel sequence” on page 70 for more information about how to continue.

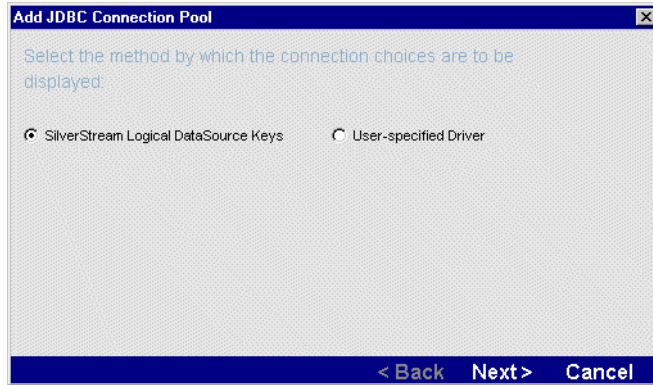
Panel reference

This section contains reference information for these tasks:

- Specifying LDS or a user-specified driver
- Specifying the pool name
- Specifying the database platform
- Specifying the JDBC driver and URL
- Specifying data source configuration properties
- Specifying connection and timeout properties
- Specifying the JDBC version
- Specifying the XA data source information

Specifying LDS or a user-specified driver

This panel is used to specify whether you are using a SilverStream LDS or a user-specified driver.

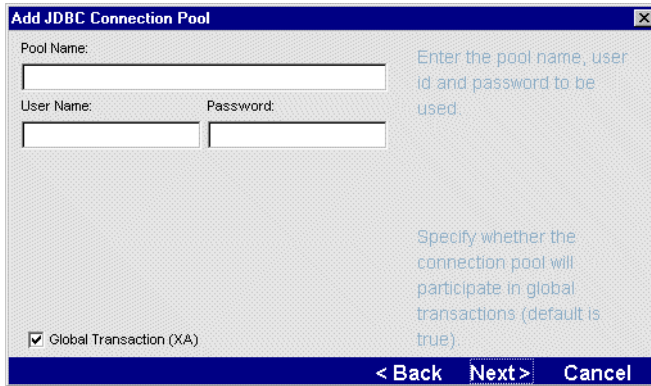


Complete the panel as follows:

Field	What to specify
SilverStream Logical DataSource Keys	Choose this option if your JDBC driver is supported as a SilverStream LDS key. (The best way to find out if your driver is supported is to choose this option and review the list in the dropdown.)
User-specified driver	Choose this option if you want to create a connection pool for a JDBC that is not represented by an LDS key.

Specifying the pool name

This panel is used to specify the name of the pool and the username/password combination that the server will use to connect to the target database.



Complete the panel as follows.

Field	What to specify
Pool Name	Enter the name for the connection pool. This name must be unique on the server. It is the name that J2EE resource references will use to connect to the database.
User Name and Password	Enter a user name/password pair that the server can use for a user connection to the native database. These values cannot be null. This user name must already be known to the native database and have the appropriate read/write permissions.
Global Transaction (XA)	<p>If this is checked (the default) then connections returned by this pool are enlisted in global transactions. J2EE applications, in general, should use transactional connection pools.</p> <p>Connections returned by nontransactional connection pools are not enlisted in global transactions even if one is active at the time of the request.</p>

Specifying the database platform

This panel lets you choose the LDS key for the JDBC driver you want to use.

Complete the panel as follows.

Field	What to specify
Database Platform	Choose from the list of supported database platforms
Driver set	Choose the driver set from the list (a driver set is a JDBC driver, sometimes in combination with SilverStream-specific files). The listed driver sets are specific to the database platform you selected. The driver set recommended for your database type is displayed by default.
LDS Key	Read-only field that displays the actual key associated with the related database platform and driver set.
Version	Read-only field that displays the JDBC version of the selected driver.

Specifying the JDBC driver and URL

This panel lets you specify information about the JDBC driver that you are supplying.

Complete the panel as follows.

Field	What to specify
JDBC Driver	(Read-only) The fully qualified name of your JDBC driver class. For example: <code>com.sybase.jdbc.SybDriver</code>
JDBC URL	The URL string defined by the driver vendor to connect to your database. The string contains replaceable parameters surrounded by percent signs (%), such as %HOST%. For example: <code>jdbc:sybase:Tds:%HOST%:%PORT%/%DATABASE%</code> Substitute these parameters with values appropriate to your database.
JDBC URL attributes	Any additional URL attributes defined by the vendor that you can use to customize the driver connection. For example: <code>cache=100</code>

Specifying data source configuration properties

This panel lets you provide any additional properties for the connection pool that your JDBC driver can support.

To supply the properties, choose **Add** and then use the following table to complete the wizard panel.

Field	What to specify
Property Name	The name of the ManagedConnectionFactory property
Property Value	The value of the ManagedConnectionFactory property.

Specifying connection and timeout properties

This panel lets you specify connection and timeout values for the connection pool.

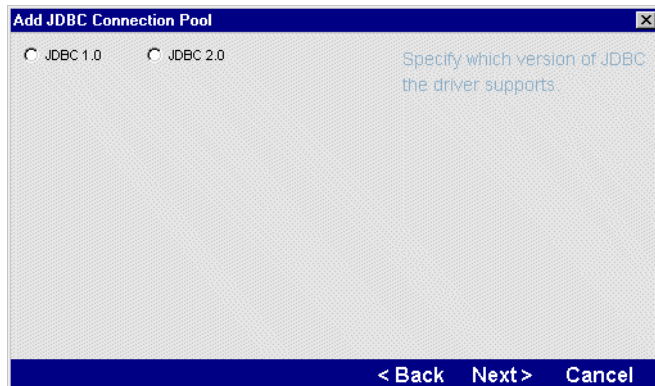
Complete the wizard panel as follows.

Field	What to specify
Minimum Connections	The minimum number of connections. The pool manager will attempt to maintain this minimum number of transactions. (This is a soft limit.)
Maximum Connections	The maximum number of connections allowed by the pool. The default is 10. Use -1 to create a pool with no maximum
Idle Connection Timeout	The idle timeout in seconds. The default is 60 seconds. When set to -1, idle timeout is disabled and no idle connections are ever closed

Field	What to specify
Connection Wait Timeout	The connection wait timeout in seconds. The default is 30 seconds. When set to -1, clients are forced to wait until a connection becomes available.
Log Level	The levels are: 0—Logging is turned off 1—Logs basic connection pool operations 2—Level 1 with more detailed operations and error messages 3—Level 2 with exception stack traces and trace output produced by JDBC driver or Connector resource adapter Messages are written to the server console

Specifying the JDBC version

This panel lets you specify the version for your JDBC driver.

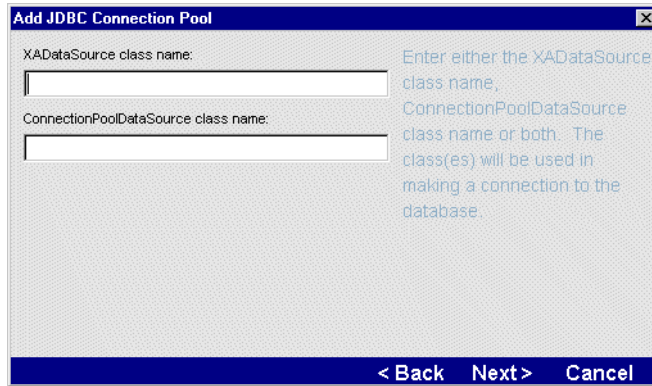


Complete the wizard panel as follows.

Field	What to specify
JDBC 1.0	Choose this option if your JDBC driver supports JDBC 1.0
JDBC 2.0	Choose this option if your JDBC driver supports JDBC 2.0

Specifying the XA data source information

This panel lets you specify the datasource class name and or connection pool class name for JDBC 2.0 drivers.



Use the following table to complete the wizard panel (you must enter a value for at least one).

Field	What to specify
XADataSource class name	Specify the fully qualified name of the XA DataSource class
ConnectionPoolDataSource class name	Specify the fully qualified Connection Pool DataSource class name

If you specify both the ConnectionPoolDataSource and XADataSource class names only one is used. The one that is used depends on the overall configuration properties that you specified in panel described in the “Specifying data source configuration properties” on page 77. The configuration properties are then applied to the instance of the data source class.

What happens

When you add a connection pool, SilverStream creates the connection to the database with the user name you’ve specified and preallocates the minimum number of connections you’ve specified.

Adding a connection pool from the command line

You can also add a connection pool from the command line or from a batch file using the SilverCmd AddCP.

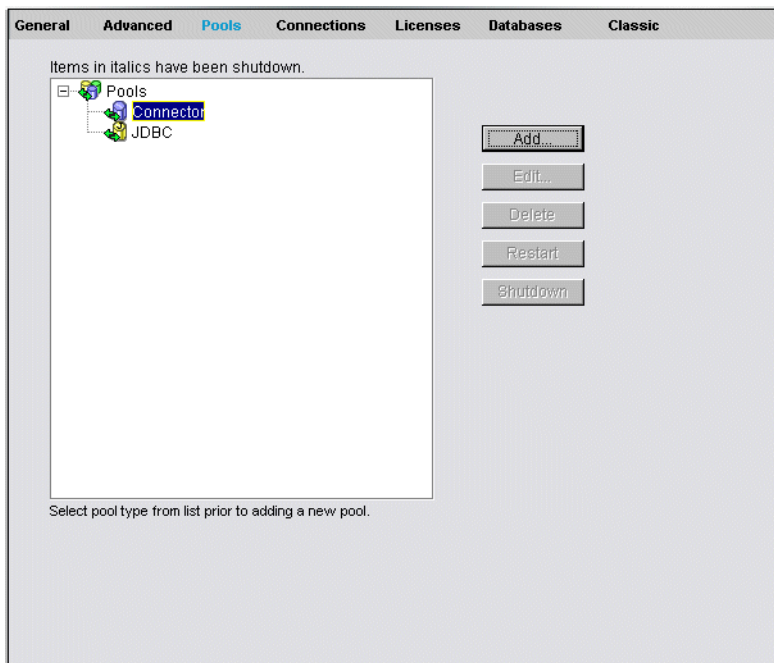
Adding a Connector connection pool

When you create a Connector connection pool, you must have a Resource Adapter deployed and enabled on the server. For more information on deploying a RAR, see Chapter 2, “J2EE Archive Deployment” in the *Facilities Guide*.

This section describes how to add connection pools using the SMC’s Add Connector Connection Pool Wizard.

➤ **To add a Connector connection pool:**

1. Start your server.
2. Start the SMC.
3. Select the server in the left pane of the SMC. If the server is not listed, add it to the SMC, as described in “Administering a SilverStream server remotely” on page 135.
4. Select the **Configuration** icon from the toolbar.
5. Select **Pools**.



6. Select **Connector** and click **Add**. You are prompted to specify the name of the pool and the username/password combination the server will use to connect to the target database.

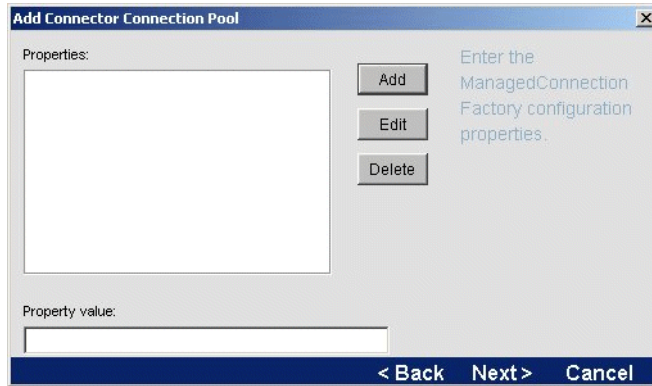


7. Provide the connection pool information as follows.

Field	What to specify
Pool Name	Enter the name for the connection pool. This name must be unique on the server. This is the name that J2EE applications that contain resource references will use to connect to the data source.
Resource Adapter Name	Enter the name under which the RAR was deployed. If you do not know the name of the deployed RAR, see the Resource Adapters panel of the SMC (available via the Deployment icon on the toolbar).
User Name and Password	Enter a user name/password pair that the SilverStream server can use for a user connection to the EIS. These values cannot be null. This user name must already be known to the EIS and have the appropriate read/write permissions.
Global Transaction (XA)	<p>If this is checked (the default) then connections returned by this pool are enlisted in global transactions. J2EE applications, in general, should use transactional connection pools.</p> <p>Connections returned by nontransactional connection pools are not enlisted in global transactions even if one is active at the time of the request.</p>

8. Click **Next**.

You are prompted to enter vendor specific properties that the pool should support.

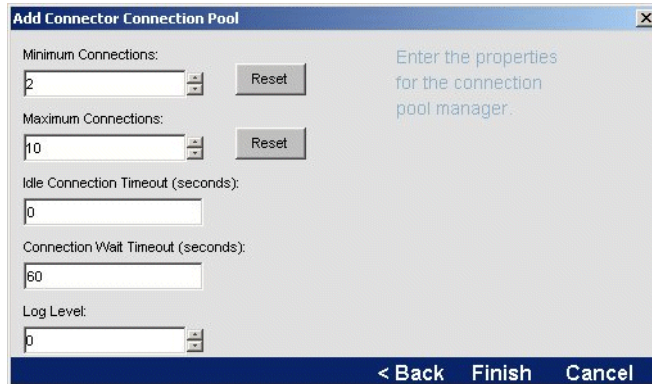


9. To supply the properties, click **Add**.

10. Complete the panel as follows.

Field	What to specify
Property Name	The name of the ManagedConnectionFactory property
Property Value	The value of the ManagedConnectionFactory property.

You are prompted to enter information about the pool connections and connection timeout values.



11. Complete the panel as follows.

Field	What to specify
Minimum Connections	The minimum number of connections. The pool manager will attempt to maintain this minimum number of transactions. (This is a soft limit.)
Maximum Connections	The maximum number of connections allowed by the pool. The default is 10. Use -1 to create a pool with no maximum.
Idle Connection Timeout	The idle timeout in seconds. The default is 60 seconds. When set to -1, idle timeout is disabled and no idle connections are ever closed.
Connection Wait Timeout	The connection wait timeout in seconds. The default is 30 seconds. When set to -1, clients are forced to wait until a connection becomes available.
Log Level	The levels are: 0—Logging is turned off 1—Logs basic connection pool operations 2—Level 1 with more detailed operations and error messages 3—Level 2 with exception stack traces and trace output produced by JDBC driver or Connector resource adapter Messages are written to the server console.

12. Click **Finish**.

What happens When you add a connection pool, SilverStream creates the connection to the EIS with the user name you've specified and preallocates the minimum number of connections that you've specified.

Adding a connection pool does not require you to restart the server unless you remove a connection pool and then add a pool (of the same type) with the same name. If the original connection pool (the one that was removed) was used by a running application, it is possible that one of the active components, such as an EJB object in the pool, has cached a reference to a `java.sql.DataSource` object. This reference might refer to the invalid connection pool. Restarting the server will clear the cached references.

Adding a connection pool from the command line You can also add a connection pool from the command line or from a batch file using the SilverCmd AddCP.

Removing a connection pool

If you don't need to maintain a connection between a database or EIS and the SilverStream server, you can remove the connection pool from the server.

➤ **To remove a connection pool from the server:**

1. Start your server.
2. Start the SMC.
3. Select the server in the left pane of the SMC. If the server is not listed, add it to the SMC, as described in “Administering a SilverStream server remotely” on page 135.
4. Select the **Configuration** icon from the toolbar.
5. Select **Pools**.
6. Select the connection pool from the list.
7. Click **Delete**.
You are asked to confirm the action.
8. Click **OK** to remove the pool from the server.

Removing a connection pool from the command line You can also remove a database from the server from the command line or from a batch file using the RemoveCP SilverCmd.

Maintaining a connection pool

You can edit a subset of connection pool properties, shut down a pool, and restart a pool.

➤ **To edit connection pool properties:**

1. Start your server.
2. Start the SMC.
3. Select the server in the left pane of the SMC. If the server is not listed, add it to the SMC, as described in “Administering a SilverStream server remotely” on page 135.
4. Select the **Configuration** icon from the toolbar.
5. Select **Pools**.

6. Select the connection pool from the list.

7. Click **Edit**.

The Edit Connection Pool Wizard displays. You can change different properties of the connection pool depending on its type. See “Panel reference” on page 72 for information about the JDBC connection pool panels; see “Adding a Connector connection pool” on page 81 for information about the connector connection pool panels.

8. Click **Next** to navigate the Edit Connection Pool Wizard to modify the properties.

9. Click **Finish** to complete the wizard. You do not need to restart the connection pool or the server for the changes to take effect.

Shutting down a connection pool

When you shut down a connection pool, it is not available to service client connection requests.

You may want to shut down a connection pool when the underlying database or EIS is temporarily brought offline—because it guarantees that the pool will not service user connection requests.

When you shut down a connection pool, all database connections are closed and all resources associated with the connections are freed. Use **Restart** (described below) to make the connection pool available again. Restarting the server will **not** restart a shut down connection pool.

➤ To shut down a pool:

1. Start your server.

2. Start the SMC.

3. Select the server in the left pane of the SMC. If the server is not listed, add it to the SMC, as described in “Administering a SilverStream server remotely” on page 135.

4. Select the **Configuration** icon from the toolbar.

5. Select **Pools**.

6. Select the connection pool from the list.

7. Click **Shutdown**.

You are asked to confirm the action.

8. Click **Yes** to shut down the pool.

Pools that are shut down appear in italic.

Recognizing an invalid connection pool If a pool name displays in bold, then it is invalid. A pool might be invalid if the connection pool failed to start at server initialization time. This may happen when the database is down or if some network problem occurs and connections cannot be created. To remove any invalid pools, see “Removing a connection pool” on page 85.

Restarting a connection pool

You can restart a connection pool that was stopped by the **Shutdown** button of the SMC.

The pool is restarted using the configuration properties (such as minimum/maximum connections, timeouts, and so on) used by the pool before it was shut down.

➤ To restart a pool:

1. Start your server.
2. Start the SMC.
3. Select the server in the left pane of the SMC. If the server is not listed, add it to the SMC, as described in “Administering a SilverStream server remotely” on page 135.
4. Select the **Configuration** icon from the toolbar.
5. Select **Pools**.
6. Select the connection pool from the list.
7. Click **Restart**.
You are asked to confirm the action.
8. Click **Yes** to restart the pool.

Part II Administering the Server

This part describes the tasks you will perform to administer the SilverStream eXtend Application Server

- Chapter 5, “Running the Server”
- Chapter 6, “Setting Up Users and Groups”
- Chapter 7, “Maintaining the Server”
- Chapter 8, “Using the Web Server Integration Modules”
- Chapter 9, “Setting Up Security”
- Chapter 10, “Using Security”
- Chapter 11, “Tuning the Server”
- Chapter 12, “Administering a Cluster”
- Chapter 13, “Using the Server Administration API”
- Chapter 14, “Troubleshooting”

5

Running the Server

This chapter describes how to run the SilverStream eXtend Application Server.

It contains sections on:

- Starting the SilverStream server
- Shutting down the SilverStream server
- Restarting the SilverStream server
- Maintaining SilverStream processes running as services
- Setting up separate ports
- Specifying general server properties
- Using server logging
- Specifying ORB settings
- Running multiple servers on one host
- Specifying character set encoding
- Running the JMS (jBroker MQ) server

Starting the SilverStream server

This section provides platform-specific information for manually starting the SilverStream server.

NOTE You can also run the SilverStream server in the background as a service in Windows or as a daemon in UNIX. For more information, see the sections in the *Installation Guide* on running the server as a service or a daemon.

This section contains the following topics:

- Starting the server
- Using startup options
- Specifying the VM to use
- Starting the server on a specific IP address or hostname

Starting the server

➤ To start the server in Windows NT:

- Do one of the following:
 - From **Start>Programs>SilverStream eXtend>AppServer version**, select **SilverStream Server**.
 - Run the SilverStream server from a DOS command line by issuing the **SilverServer** command with one or more startup options.

➤ To start the server on UNIX or Linux:

1. From the command line, change to the server's /bin directory.
2. Type **./SilverServer** and any startup options. To print a list of available options, enter the following:

```
./SilverServer -?
```



For more information, see “Using startup options” below and “Specifying the VM to use” on page 98.

Using startup options

There are two kinds of startup options you can provide on the command line with SilverServer:

Type of startup options	How you specify them
Options passed directly to the JVM or handled by the SilverServer executable to launch the JVM	Specify these options using the plus (+) sign
SilverStream-specific options passed to the SilverStream class that starts the server	Specify these options using the minus (-) sign

How it works

When passing the options that were specified with the plus (+) sign to the JVM, SilverStream changes the pluses to minuses for processing by the JVM. For example, say you specify the following command line:

```
SilverServer +verbose -nodbcheck
```

The equivalent command line is:

```
java -verbose ServerStartupClass -nodbcheck
```

Displaying +options


To see a list of possible options for the JVM, type the following commands:




Command	Description
java -?	Lists standard options.
java -X	Lists nonstandard options. Note that these options are subject to change without notice.



NOTE SilverStream automatically adds the following option with the appropriate value to the command line: **-Djava.class.path**. This option will override any corresponding option you specify on the command line.


Table of options



These are the server startup options:

Server startup option	Description
Supported Java options: +<x>	
(These options are passed to the JVM. For more information about the non-SilverStream + options, see your Java documentation.)	
+classic	<p>(Windows only) SilverStream-specific option.</p> <p>Use the classic VM.</p> <p>You must set this option along with +profile to profile server-side applications when you are running a version of the Sun Microsystems HotSpot JVM that does not support the Java Virtual Machine Profiler Interface (JVMPi).</p> <p> For more information, see “Specifying the VM to use” on page 98 and the chapter on the SilverStream Profiler in the <i>Tools Guide</i> of the server’s Classic Development Help.</p>

Server startup option	Description
+client	<p>(Windows only) SilverStream-specific option.</p> <p>Use the client HotSpot VM.</p> <p> For more information, see “Specifying the VM to use” on page 98.</p>
+cp:a <i>path</i>	<p>Appends specified <i>path</i> to the class path. This option makes additional Java classes available to SilverStream applications by appending the specified path to the class path.</p> <p>NOTE It is recommended that you use the AGCLASSPATH environment variable to extend Java classes. For more information, see “Setting the AGCLASSPATH variable” on page 138.</p>
+cp:p <i>path</i>	<p>Prepends specified <i>path</i> to the class path. Don’t use this debugging option without first contacting SilverStream Technical Support. Instead, use AGCLASSPATH to make additional Java classes available to SilverStream server applications. For more information, see “Setting the AGCLASSPATH variable” on page 138.</p>
+debug	<p>SilverStream-specific option.</p> <p>You must set this option to debug server-side objects.</p> <p> For more information, see the Debugger documentation in the eXtend Workbench help or in the SilverStream server’s Classic Development Help.</p>
+Djava.compiler=none	<p>SilverStream-specific option.</p> <p>You must use this option with +profile if you want the Profiler Timing tool to provide detailed information about how method duration is distributed across nested calls in server-side processes.</p> <p> For more information, see SilverStream Profiler in the <i>Tools Guide</i> of the server’s Classic Development Help.</p>

Server startup option	Description
+profile	<p>SilverStream-specific option.</p> <p>You must set this option to profile server-side applications.</p> <p> For more information, see the chapter on the SilverStream Profiler in the <i>Tools Guide</i> of the server's Classic Development Help.</p>
+server	<p>(Windows only) SilverStream-specific option.</p> <p>Use the server HotSpot VM.</p> <p> For more information, see "Specifying the VM to use" on page 98.</p>
+verbose[:class gc jni vmopts]	<p>Run the JVM verbosely.</p> <p>There is a SilverStream-specific option for +verbose:</p> <pre>+verbose:vmopts</pre> <p>Specifying this option tells SilverStream to output the startup options to the console without all the other output generated in verbose mode.</p>
+Xms <i>size</i>	<p>Initial Java heap size within the VM. Default value is 16 MB.</p>
+Xmx <i>size</i>	<p>Maximum Java heap size within the VM. Default value is 256 MB.</p> <p>You can override +Xms and +Xmx. For example, if you are running a development server that only services one user, you might want to run the server with a lighter memory footprint using the following command line:</p> <pre>SilverServer +Xms2m +Xmx16m</pre> <p>This sets the initial Java heap size to 2 MB and the maximum heap size to 16 MB.</p>
<p>SilverStream server options: -<x></p> <p>(These options are passed to the SilverStream server.)</p>	
-? or -help	<p>Print usage for SilverServer.exe.</p>

Server startup option	Description
--a	<p>Print the server startup properties, then exit without starting the server.</p> <p>This debugging option is useful if the server fails to start. You can see what the startup properties are.</p>
-host <i>hostname</i>	<p>The full name of the host running the server. Not required unless there are problems with hostname resolution.</p>
-jvmversion	<p>Print information about the Java VM.</p>
-minspan <i>number</i>	<p>Use with -retry <i>number</i> (see below). The duration in minutes within which the retries must be made. SilverMonitor ceases to operate after the number of minutes specified by minspan, even if not all retry attempts have been made. The default value is 10.</p> <p> For more information, see “Using SilverMonitor” on page 414.</p>
-nodbcheck	<p>Don’t check database integrity at startup.</p> <p>This option is useful when you have a database that is not synchronized. Using this option and noexitondbcheck are the only ways the server will start if the databases are out of sync.</p> <p>If you are sure your databases are synchronized, you can also use this option to shorten the server startup time.</p>
-noexitondbcheck	<p>Don’t exit if the database integrity check fails.</p> <p>Use this option to check integrity and allow access to the SMC if the database check fails.</p>

Server startup option	Description
-nominator	<p>Run without the SilverMonitor background program.</p> <p>This option is useful for debugging the server when it fails to start. If the option is not used, the server will keep trying to start.</p> <p> For more information, see “Using SilverMonitor” on page 414.</p> <p>NOTE If you start the server with -nominator, you will not be able to restart the server from the SMC (or using the API). You will have to shut down the server, then start it again manually.</p>
-noserverlisteners	<p>Do not run any business objects.</p> <p>This option is useful if there are problems in your business objects.</p>
-nostartagents	<p>Do not run ServerStart business objects.</p> <p>This option is useful if you are having problems starting the server.</p>
-p <i>file</i>	<p>Read startup properties from a specified file.</p> <p>Defaults to SilverStream\Resources\httpd.props.</p>
-retry <i>number</i>	<p>The number of times SilverMonitor should attempt to restart the server or process before ceasing to operate. The default value is 3. See -minspan <i>number</i> (above).</p> <p> For more information, see “Using SilverMonitor” on page 414.</p>
-trace	<p>Turn tracing on. Dump tracing information to the default or specified log output.</p>

Specifying the VM to use

The HotSpot VM shipped with the SilverStream server comes (on most platforms) in two versions: a client version and a server version. This section describes how the SilverStream server-side processes (the server, Cache Manager, Load Manager, and Dispatcher) and clients (such as SilverJ2EEClient, SilverJRunner, SilverStream Management Console, and Designer) use these VMs.

On Windows

On Windows, by default the SilverStream server-side processes and clients all use the client version of the HotSpot VM.

To override this behavior on Windows, you can use the following startup options with any of the SilverStream executables that start a VM:

Executable	VM it uses
+server	Server HotSpot VM
+client	Client HotSpot VM
+classic	Classic VM

On UNIX and Linux

On UNIX and Linux, the VM usage is different. These are the defaults:

Platform(s)	VM usage
Solaris and Linux	Server processes use the server HotSpot VM Clients use the client HotSpot VM
AIX	Server processes and clients use the classic VM
HP-UX	Server processes and clients use the HotSpot VM (there is only one HotSpot version on HP-UX)

To override this behavior for UNIX and Linux server-side processes, edit the server's .agprofile file. Look for the case statement and update the definition of the native search path for your platform (LD_LIBRARY_PATH, SHLIB_PATH, or LIBPATH) to point to the VM you want.

Starting the server on a specific IP address or hostname

You can set the **http-server.com.ssw.srv.host** property in the `httpd.props` file (located in the server's `\resources` directory) to direct the SilverStream server to start on a specific IP address or hostname. This feature is particularly helpful on machines with multiple network cards and multiple IP addresses (**multihoming**). This feature works identically on Windows NT and UNIX.

For example:

```
http-server.com.ssw.srv.host=192.101.1.10
```

Shutting down the SilverStream server

Use the server **Stop** button on the SMC (see procedure below) to shut down the resident or selected server when you are taking the machine out of service, or if you need to install a software patch.

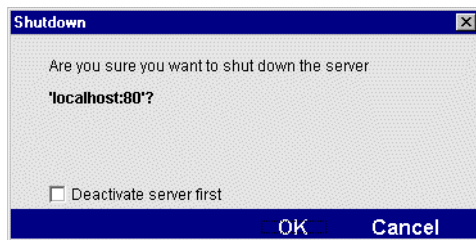
NOTE If you want to stop and restart the server in order to activate properties that you have modified, use the **Restart** button, as described in “Restarting the SilverStream server” on page 100.

TIP You can also shut down the server from the Main Designer. See the Main Designer chapter in the *Tools Guide* of the server's Classic Development Help.

➤ To shut down a server:

1. Start the SMC.
2. Select the server you want to stop from the left panel.
3. Click **Stop**.


The following confirmation displays.



4. (Optional) Select **Deactivate server first** if you want the server deactivated before it is shut down or restarted (see table below for more information).
5. Select **OK**.

What happens next depends on whether or not you selected **Deactivate server first**.

Situation	Result
Deactivate server first is not selected	The server is immediately shut down or restarted.
Deactivate server first is selected	<p>No new client sessions can be established, but existing client sessions continue to operate normally. In clusters, the deactivated server is unregistered from the Load Balancer so new sessions will not be dispatched to the server (if you are using a third-party load balancer, SilverStream has no way of notifying it that the server is deactivated).</p> <p>Once the last client session is closed (typically five minutes after the last client connection is closed), the server is declared deactivated and is shut down or restarted.</p>

 For information on programmatically handling server deactivation, see the chapter on server-triggered business objects in the *Programmer's Guide* of the server's Classic Development Help.

Restarting the SilverStream server

Using the **Restart** button is the recommended way to stop and restart a server to update server property changes you have made using the SMC.

You can restart a server only if it was started with SilverMonitor (which is the default). For more information, see “Using SilverMonitor” on page 414.

➤ To restart a SilverStream server:

1. Start the SMC.
2. Select the server from the left panel.

3. Click **Restart**.

The server restarts using the same startup parameters as when it was originally started and picks up any changes you have made to server properties.

Maintaining SilverStream processes running as services

Instead of manually starting the SilverStream server, you can run it as a service (or as a daemon in UNIX), so that it starts automatically when the server machine reboots. For information on installing the server to run as a service, see the chapter on installing the server on Windows or the chapter on installing the server on UNIX in the *Installation Guide*.

In addition to running the server as a service, you can also run the following SilverStream processes as services:

- Load Manager
- Dispatcher
- Cache Manager

To help you manage SilverStream processes running as Windows services, the server provides **SilverServiceUtil**, a Windows utility that allows you to:

- Create, list, delete, and stop SilverStream and non-SilverStream services
- Define additional Windows services for SilverStream processes
- Create new console log files each time a SilverStream service is restarted
- Define command-line arguments that will be automatically passed to a SilverStream process when it starts as a Windows service
- Reconfigure an existing SilverStream process when it is running as a service

NOTE To initially get the server set up to run as a service, you should use the installation program, as described in the section on installing the SilverStream server as a service in the Windows chapter in the *Installation Guide*. After you have the server running as a service, you can use SilverServiceUtil to maintain your service environment, including creating additional services on the same machine.

Using SilverServiceUtil

SilverServiceUtil is a command-line utility in the server's bin directory. To invoke it, change to the bin directory and enter:

```
SilverServiceUtil
```

You will see usage notes.

The utility has the following actions:

Action	Description
addDepend	Add a dependency to the service—if a service is dependent on another service, the Windows service manager won't start it until the other service has started
create	Create a new service
delete	Delete an existing service
list	List all services defined on the current machine
stop	Stop a running service
update	Update the configuration of an existing SilverStream service

These actions are described next. For complete information about using each of the SilverServiceUtil actions, enter the following:

```
SilverServiceUtil -action -help
```

Defining a dependent service

To specify that a service is dependent on another service, enter the following:

```
SilverServiceUtil -addDepend -service serviceName -prereq prereqServiceName
```

where:

Option	Description
serviceName	The name of the service that is dependent on another service.
prereqServiceName	The name of the service that <i>serviceName</i> is dependent on.

Once this dependency has been defined, the Windows service manager will not start *serviceName* until *prereqServiceName* has been started.

Creating a service

You can use `SilverServiceUtil` to create a `SilverStream` service or a non-`SilverStream` service.

Creating a SilverStream service You can use `SilverServiceUtil` to configure all installed `SilverStream` servers to run as a service.

To create a `SilverStream` service, enter the following:

```
SilverServiceUtil -create -service serviceName -display displayName -program
pathToExecutable
[-outputDir outputDirectory -maxOutputFiles numFiles -startupOptions options]
```

where:

Option	Description
serviceName	The name of the service to create. The name is arbitrary, but it must be unique.
displayName	The display name of the service. The name is arbitrary, but it must be unique.
pathToExecutable	<p>Path to the executable to invoke for the service. Specify one of the following executables (all are in the server's bin directory):</p> <ul style="list-style-type: none"> • SilverAppServerService.exe to run the server as a service • SilverCacheManagerService.exe to run the Cache Manager as a service • SilverDispatcherService.exe to run the SilverStream Dispatcher as a service • SilverLoadManagerService.exe to run the Load Manager as a service <p>If you are configuring multiple servers on a single host to run as services, make sure you point to unique executables each time you execute <code>SilverServiceUtil</code>.</p>

Option	Description
outputDirectory	<p>(Optional) Path to the directory in which to save log files.</p> <p>If not specified, the log files are saved in the SilverStream temp directory.</p>
numFiles	<p>(Optional) Maximum number of log files to create in <i>outputDirectory</i>.</p> <p>If you specify 0, or if the value is not specified, the log file will be overwritten each time the service is restarted. The log file will be named:</p> <pre>nameOfService.out</pre> <p>For example, if you are running a Version 4 server as a service named SilverAppServerService4 and have specified 0 for <i>numFiles</i>, the one and only log file would be named:</p> <pre>SilverAppServerService4.out</pre> <p>If you specify a number larger than 0, a new file will be created each time the service is restarted, up to the number you specify, after which the numbering restarts. The log files will be named:</p> <pre>nameOfService.nnn.out</pre> <p>Using the preceding example, the first log file would be named:</p> <pre>SilverAppServerService4.000.out</pre> <p>The next time the service starts, the following log file will be created:</p> <pre>SilverAppServerService4.001.out</pre> <p>A warning message is sent to the server console when 80 percent of <i>numFiles</i> has been reached. If <i>numFiles</i> itself is reached, the service will behave as if <i>numFiles</i> were defined as 0, until the output files are removed.</p>
options	<p>(Optional) Command-line options to pass to the executable when it is started. Enclose the options in double quotation marks. The specific options depend on the executable.</p>

Creating a non-SilverStream service You can also use this option to create a generic Windows service.

To create a non-SilverStream service, enter the following:

```
SilverServiceUtil -create -service serviceName -display displayName -program
pathToExecutable -generic
```

Listing and deleting services

You can list all services defined on the current machine as well as delete any existing service (SilverStream as well as non-SilverStream).

To list all services, enter the following:

```
SilverServiceUtil -list [-d]
```

If you specify -d, the display name will be listed along with the service name.

To delete a service, enter the following:

```
SilverServiceUtil -delete -service serviceName
```

Specify the service name, not the display name. (You are not asked to confirm the deletion.)

Stopping a service

To stop a service, enter the following:

```
SilverServiceUtil -stop -service serviceName [-retries numRetries -delay retryDelay]
```

where:

Option	Description
<code>serviceName</code>	The name of the service to stop
<code>numRetries</code>	(Optional) Number of times to query the service manager to determine whether the server has stopped. If not specified, the service manager is not queried.
<code>retryDelay</code>	(Optional) Seconds between retry attempts

Reconfiguring a service

You can reconfigure an existing SilverStream service. Enter the following:

```
SilverServiceUtil -update -service serviceName  
[-outputDir outputDirectory -maxOutputFiles numFiles -startupOptions options]
```

where the `-outputDir`, `-maxOutputFiles`, and `-startupOptions` arguments are the same as used in the create action.

The update action modifies the Windows registry entry for the corresponding service. The changes won't take effect until the service is restarted. You can start and stop services through the Windows Services control panel, without rebooting the machine.

Setting up separate ports

To restrict access to specific types of SilverStream server operations, you can define the following three types of ports:

Port	Description
Runtime	Allows users to run Java and HTML application objects such as EJBs, JSP pages, forms, views, pages, and so on using HTTP, HTTPS, or RMI.
Design	Allows developers to build, test, and deploy applications by having the ability to read and write the design information for pages, forms, business objects, and so on. The design port is required for anyone running the SilverStream Designer (all Designer operations use the design port, even running a page from the Designer), SilverCmd, or a third-party IDE.
Administration	Allows administrators to set or modify server administration settings such as the ability to read and write server settings, security, license information, certificates, and so on. The administration port is required for anyone making Server Administration API calls using the SMC, SilverCmd, a SilverStream administration object, or a third-party application.

Each port type excludes URLs and operations that are not associated with it. For example, the administration port only passes administration URLs. Similarly, you cannot make application design changes with the administration port. The separate ports are designed to work in conjunction with your server permission settings. For example, if your administration, design, and runtime ports are unique, any attempt to run an administration URL on any port other than the administration port will fail. Once a user successfully accesses an administration port, the server checks the user's group permissions to further determine the level of access. The administration port is required for any user running the SMC or making Server Administration API calls.

How you configure a public site would probably be different from the way you would configure an e-commerce site that uses credit card transactions. Particularly in an extranet environment, you don't want users attempting to perform administration operations or design tasks that will alter your application data. Configuring multiple server ports in conjunction with your corporate firewall lets you manage internal and external access to your applications.

Using separate ports with your firewall

There are several security advantages to defining separate SilverStream server ports for different types of users and operations:

- Opening a single runtime port through your firewall for users outside your organization limits your security risk.
- Separating administration and runtime port access helps prevent unauthorized users from attempting to administer the server. Users connecting to your applications may know the runtime port number, but only administrators need be aware of the administration port.
- Defining an administration port that can only be accessed from inside the firewall helps restrict calls made to that port.

You can set port properties using the SMC or by editing the `httpd.props` file. You can also use the Server Administration API to manage server port properties. For more information, see Chapter 13, "Using the Server Administration API".

About enabling ports

The server supports ports for runtime, design, and administrative access for each of the following three protocols: HTTP, HTTPS-RSA, and HTTPS-DSA. The default ports are 80 for HTTP, 443 for HTTPS (RSA), and for HTTPS (DSA). After installation, only the HTTP port is enabled. The DSA and RSA ports are set to the default values, but not enabled. The server is not listening on the DSA and RSA ports until you enable them.

When the SilverStream server starts, it binds a socket to each unique port value you have configured and enabled. The server does not require unique port values for the different types of access; ports having the same value will share the same socket and will allow multiple operations. For example, if you set your HTTP runtime and HTTP design ports to 8080, the server will use only one socket to accept requests for both.

TIP When you install the SilverStream server, all three HTTP runtime, design, and administrative ports are configured to whatever port number you specify as the default. You will need to update your NT program shortcut used to launch the SMC if you configure a separate administration port after you install the SilverStream server.



For alternative ways to start the SMC, see “Running the SMC” on page 21.

Clients connecting to a design or administration port may perform only operations associated with that port. Because many application and server objects involved with design or administration require runtime support, runtime operations can be performed on any port. However, runtime ports only allow runtime operations.



For how to enable HTTP ports, see “Specifying general server properties” on page 110. For more information about enabling HTTPS ports, see “Enabling RSA/DSA ports” on page 251.

When to use the administration port

The administration port is required to:

- Run the SMC
- Make server administration API calls
- Use SilverCmd—**only** for commands that will add and/or delete databases or change security properties
- Add or remove databases in the Designer

You will not see any databases you have added in the Designer until you click **Refresh** (or press F5) for each configured port.

TIP To run the SMC from the Designer (using the Manage Server icon), be sure that you have already added the server to the Designer using the Administration port.



For more information, see the section on adding a database in the Main Designer chapter of the *Tools Guide* of the server's Classic Development Help, or see AddDatabase in the SilverCmd Reference in the *Facilities Guide*.

Port types

The SilverStream server can be set to a maximum of nine unique port numbers for HTTP/HTTPS communications. The type of operations a port allows and its associated security protocols can be configured independently—that is, you can mix any of the three security protocols with any of the three port types.

All port property names (as defined in the https.props file) begin with **http-server**. For more information, see Appendix A, “The httpd.props File”.

Port property name	Connection description	Default
com.sssw.srv.port_rt	Runtime access to an unencrypted port using HTTP	80
com.sssw.srv.port_des	Design-time access to an unencrypted port using HTTP	80
com.sssw.srv.port_admin	Administrative access to an unencrypted port using HTTP	80
com.sssw.srv.https.port_rsa_rt	Runtime access to an SSL port using RSA encryption	443
com.sssw.srv.https.port_rsa_des	Design-time access to an SSL port using RSA encryption	443
com.sssw.srv.https.port_rsa_admin	Administrative access to an SSL port using RSA encryption	443
com.sssw.srv.https.port_dsa_rt	Runtime access to an SSL port using DSA encryption	443

Port property name	Connection description	Default
com.sssw.srv.https.port_dsa_des	Design-time access to an SSL port using DSA encryption	443
com.sssw.srv.https.port_dsa_admin	Administrative access to an SSL port using DSA encryption	443

NOTE Although the server port number defaults in the table match the configuration settings of earlier releases, the older port property names have been deprecated and should not be used.

Specifying general server properties

General server properties include:

- HTTP listener ports
- Name of the SilverMaster database
- User account that starts the server on UNIX



➤ **To specify general server properties:**

1. Start the SMC.
2. Select the server.
3. Select the **Configuration** icon from the toolbar.


4. Select **General**.

The screenshot shows the 'General' configuration tab for SilverStream Server. At the top, there are tabs for 'General', 'Advanced', 'Pools', 'Databases', 'Connections', and 'Licenses'. The main content area is titled 'SilverStream Server [Build Number:Release4.0.0 (020716_1)]'. Under 'HTTP ports:', there are three checked options: 'Enable Runtime port', 'Enable Admin port', and 'Enable Design port', each with a 'Port number' field set to '80'. Under 'Server logging:', 'Log output:' is set to 'Database logging'. There are three unchecked options: 'Enable HTTP logging', 'Enable Error logging', and 'Enable Trace logging'. The corresponding log table fields are 'AgLog', 'AgErrorLog', and 'AgTraceLog'. Under 'ORB settings:', 'Name services port:' is '54890', 'IOP SSL min port:' is '-1', and 'IOP SSL max port:' is '-1'. There are two checkboxes: 'Use SSL for Remote Objects' (unchecked) and 'Enable RMI Server' (checked). Under 'Misc. settings:', 'Username for server (UNIX only):' is 'root' and 'SilverMaster database name:' is 'SilverMaster40'. An 'Update' button is at the bottom right.

5. Edit any of these fields as needed:

Field(s)	Description
<p>Enable HTTP runtime port and Port number</p> <p>Enable HTTP Design port and Port number</p> <p>Enable HTTP Admin port and Port number</p>	<p>To enable HTTP listener ports, select any or all of the three HTTP port options and then specify a corresponding port number. You can enable up to three HTTP ports.</p> <p>By default, the SilverStream server listens to port 80 for all three HTTP port types. The default ports are 443 for HTTPS (RSA) and for HTTPS (DSA).</p> <p> For more information, see “Setting up separate ports” on page 106.</p> <p> To disable HTTP communications, see “Turning off HTTP communications” on page 252.</p>
<p>Username for server (UNIX only)</p>	<p>You can specify the user under whose account the server is started on UNIX. By default, it is root.</p>

Field(s)	Description
SilverMaster database name	<p>The SilverStream server relies on a master database catalog called the SilverMaster for overall system management. This database is specified during installation.</p> <p>You can change the database used as SilverMaster (for example, if you are setting up a cluster and need to change all servers in that cluster to use the same SilverMaster) by setting the SilverMaster field.</p>

6. Click **Update**.
7. To activate the changes, click **Restart**.
 For more information, see “Restarting the SilverStream server” on page 100.

Using server logging

SilverStream offers logging options that you can use for various purposes, such as server debugging, general server monitoring, and security auditing. These options allow you to log information to a file or a database or to specify your own custom class to perform the logging.

➤ To turn on logging:

1. Start the SMC.
2. Select the **Configuration** icon from the toolbar.

3. Select **General**.

SilverStream Server [Build Number:Release4.0.0 (020716_1)]

HTTP ports:

Enable Runtime port Port number: 80

Enable Admin port Port number: 80

Enable Design port Port number: 80

Server logging:

Log output: Database logging File logging User Defined

Enable HTTP logging HTTP log table: AgLog

Enable Error logging Error log table: AgErrorLog

Enable Trace logging Trace log table: AgTraceLog

ORB settings:

Name services port: 54890 Use SSL for Remote Objects

IOP SSL min port: -1 Enable RMI Server

IOP SSL max port: -1

Misc. settings:



Username for server (UNIX only): root

SilverMaster database name: SilverMaster40

Update

4. Select the logging option(s) you want to turn on or off as follows:

Field	Description	Usage
Database logging	Logs messages to SilverMaster. Messages are stored in the AgLog, AgErrorLog, and AgTraceLog system tables.	This is the default setting.
File logging	Logs messages to files that you specify.	Specify the file names for each option that you activate in the text field next to the option, which is enabled if File logging or User Defined is selected.

Field	Description	Usage
User Defined	Uses a custom Java class to do the logging.	<p>By default, SilverStream uses its own internal class to do the logging. If you want to customize the log output—for example, to specify an extended log file format—you can write your own logging class, then specify it here.</p> <p> To learn how to create and use a custom logging class, see “Sample code” on page 387.</p>
Enable HTTP logging	Writes a line in the AgLog table (or file you specify) for every client request to the server and every server response.	<p>Run in conjunction with error logging. Use standard HTTP logging when you want to see client requests to the server and also when you want to monitor server activity.</p> <p>Use in conjunction with the Statistics /Summary/ /Request time option in the SMC (see “Summary statistics” on page 157).</p> <p> For more information, see “About HTTP logging” on page 115.</p>
Enable Error logging	Records errors and miscellaneous status information in the AgErrorLog table (or the file you specify). If you enable this type of logging you get more detailed information about server errors and status.	SilverStream recommends that you turn this option on.

Field	Description	Usage
Enable Trace logging	Records server actions. Unlike HTTP logging and error logging, trace logging concentrates on tracking server events as well as error messages. When enabled, the AgTraceLog table (or the file you specify) contains additional tracing information that SilverStream Technical Support uses to track down server issues.	Turn on this option only if SilverStream Technical Support requests it.

5. Click **Update**.

SilverStream starts logging the specified information.

About HTTP logging

By default, when you enable HTTP logging, the server logs HTTP messages in the standard W3C common log file format (see www.s3.org) to the database. You can redirect the log to a file as described above.

There is also a compound log file format, which is like the common log file format except that it also logs the Referer (which allows click tracing) and User-Agent (which logs the browser type) fields from each HTTP request. SilverStream provides built-in support for the compound log file format.

➤ To log using this format:

1. In the SMC, under Server logging, select **User Defined**.
2. In the **Java class** field, specify the following:

```
com.sssw.srv.http.CompoundLogger
```
3. Specify the file to log the HTTP requests to (and optionally, files for error and trace logging, also supported by the CompoundLogger class).

4. Restart the server.

TIP Instead of using the SMC, you can also specify compound logging by setting these values in `httpd.props` and restarting the server:

```
http-server.com.ssw.srv.logger=com.ssw.srv.http.CompoundLogger
http-server.com.ssw.srv.logger.logname=fileName
```

There are many log file formats. SilverStream provides built-in support for the common and compound log formats, but you can write your own logging class using the SilverStream `AgiLogger` interface. To learn how to create and use a custom logging class, see “Sample code” on page 387.

Viewing the log

If you are using the built-in logging class, you can view the log in the SMC (see “Displaying logs” on page 151). If you are logging to the database, you can also use the Form Designer or View Designer to build a form or view to view the log. You can also use the SilverCmd `PrintLog` to view the log in the SilverCmd console window or a file. `PrintLog` allows you to view the data regardless of whether you are using the built-in logging class or the database.

Maintaining log tables and files

Log information can expand quickly. Clean out your log tables and log files to keep them manageable. You can use SilverCmd `ClearLog` to delete records (a SilverStream developer can write a timed business object to do this). Use your native database utilities to maintain these tables, or use any editor to shorten or delete extraneous information from log files.

Specifying ORB settings

You can use the SMC to specify whether to use RMI—and if so, its name services port—whether to use SSL for the remote objects, and the ports to use for IIOP SSL.

➤ **To specify ORB settings:**

1. Start the SMC.
2. Select the **Configuration** icon from the toolbar.

3. Select **General**.

SilverStream Server [Build Number:Release4.0.0 (020716_1)]

HTTP ports:

Enable Runtime port Port number:

Enable Admin port Port number:

Enable Design port Port number:

Server logging:

Log output: Database logging File logging User Defined

Enable HTTP logging HTTP log table: AgLog

Enable Error logging Error log table: AgErrorLog

Enable Trace logging Trace log table: AgTraceLog

ORB settings:

Name services port: Use SSL for Remote Objects

IOP SSL min port: Enable RMI Server

IOP SSL max port:

Misc. settings:

Username for server (UNIX only):

SilverMaster database name:

Update

4. Specify the RMI options.

Field	Description
Name services port	The port that SilverStream starts the RMI name services on (for example, all clients that need to find an EJB use this service). The default is 54890.
Enable RMI Server	<p>Specifies whether to enable use of RMI for unencrypted client communications. You can enable RMI separately or together with HTTP.</p> <p>If selected, the SilverStream server exports a remote server object using RMI/IOP and accepts RMI sessions, so that non-HTTP clients that want to call invoked business objects (especially EJBs) on the SilverStream server don't require an HTTP session on the server.</p> <p>If this option is not selected, the RMI server is not created and it does not accept RMI sessions.</p> <p>NOTE To encrypt a remote transaction, enable the Use SSL for Remote Objects option.</p>

Field	Description
Use SSL for Remote Objects	<p>Specifies whether to use SSL encryption to secure the SilverStream RMI Server (if enabled), remote sessions, and the remote user transaction.</p> <p>If selected, remote objects (such as EJBs) can be encrypted and exported by non-HTTP clients using RMI/IIOP.</p>
IIOP SSL min port	<p>Specifies the lower bounds (in a range) for IIOP SSL communications. If you do not specify a range, the ORB picks the first available port. Use -1 when you do not need to specify a range.</p> <p>You must create an IIOP SSL port range:</p> <ul style="list-style-type: none"> • To allow interoperability with network firewalls. Controlling the range used by IIOP SSL communications allows the firewall administrator to open these ports and configure traffic appropriately. • When your environment supports session-level failover for EJBs using IIOP SSL communications. The range must be large enough to allow 1 port for each unique combination of EJB security attributes used by the beans deployed on your system. The maximum number of combinations of security attributes (therefore the largest range) is 64 (when using the standard set of cipher suites. A range of 16 is a reasonable number of ports for most common installations.
IIOP max port	<p>Specifies the upper bounds (in a range) for IIOP SSL communications. If you specify -1 there is no upper bound to the range.</p>

5. Click **Update**.
6. To activate the changes, click **Restart**. For more information, see “Restarting the SilverStream server” on page 100.

Running multiple servers on one host

You can run more than one SilverStream server on a host at one IP address. (The SilverStream server also supports **multihoming**, where you have multiple IP addresses through multiple network cards on one host. See “Starting the server on a specific IP address or hostname” on page 99).

If you choose to run more than one server on a host with the same IP address, you need to be aware of the following issues:

- Specifying unique ports
- Host-based properties




Specifying unique ports



Multiple servers running on one host must be configured to use unique ports. You can specify runtime, design, and administration ports in the SMC.

Ports	Default	Information on setting
HTTP	80	“About enabling ports” on page 108
RSA (used with HTTPS/SSL communications with an HTML client)	443	“Enabling RSA/DSA ports” on page 251
DSA (used with HTTPS/SSL communications with a Java client)	443	“Enabling RSA/DSA ports” on page 251
RMI name services	54890	“Specifying ORB settings” on page 116
SSL IIOP port	-1	Specifying ORB settings



Host-based properties

Some server properties are host-based; so if you are running with more than one SilverStream server on the same host using the same IP address, these properties are used by all servers on the host. That means if you change one of these properties for any server on the host, the property is changed for all servers running on that host.

Server properties	Shared server settings
Buffer	<ul style="list-style-type: none"> • Number of prefetch buffers • Buffer size for buffering replies <p> For more information, see “Setting performance parameters” on page 303.</p>
Cache	<ul style="list-style-type: none"> • Whether the content cache is enabled • The maximum size of the disk cache • The maximum size of any file that will be cached in the disk cache • The maximum size of the in-memory cache • The maximum size of any file that will be cached in the in-memory cache <p> For more information, see “Managing the server content cache” on page 311.</p>
Client connection	<ul style="list-style-type: none"> • Maximum number of client connections • Free client connections for busy • Free client connections for light • Idle connections for light • Minimum number of idle client connections (this is not configurable in the SMC; you can set this value in the API through <code>AgiAdmServer.PROP_IDLECLIENTCONNSBUSY</code>) <p> For more information, see “Managing client connections” on page 306.</p>

Server properties	Shared server settings
Database	<ul style="list-style-type: none"> • Maximum number of database connections • Minimum number of database connections  For more information, see “Setting the maximum and minimum number of database connections” on page 320.
UNIX user name	<ul style="list-style-type: none"> • The user name the server is to be run under (UNIX servers only)  For more information, see “Specifying general server properties” on page 110.

Some cluster properties are also host-based. If you are running a cluster with more than one SilverStream server on the same host using the same IP address, then the cluster properties listed in the following table are used by all servers in the cluster on the host. If you change one of the cluster properties for any server on the host, the property is changed for all servers in the cluster running on that host.

Cluster properties	Shared server settings
Cache Manager	<ul style="list-style-type: none"> • Start sleep interval • Reconnect sleep interval • Start try count • Reconnect try count  For more information, see “Managing failover” on page 354.
Load Manager	<ul style="list-style-type: none"> • Connect try interval • Connect sleep interval • Connect try count • Connect sleep count  For more information, see “Managing failover” on page 354.

Specifying character set encoding

The SilverStream server uses the following server property when URL-encoding and -decoding form content:

```
http-server.com.sssw.srv.international.UrlEncoding
```

NOTE SilverStream stores the encoding and other server startup properties in the **httpd.props** configuration file, which is located in the SilverStream Resources directory. For more information, see Appendix A, “The httpd.props File”.

By default, the SilverStream server uses utf-8 (Universal Character Set Transfer Format) for URL encoding and decoding. Because utf-8 can encode ASCII characters without requiring modification, utf-8 works well for English and most Western languages. Because languages using multibyte encodings are not a subset of utf-8, character encoding and decoding will not work properly with them.

When to change the encoding scheme

You typically need to change the encoding scheme only when the majority of client browsers in your environment use character encodings that are not ISO 8859-1 (Latin 1). For example, a Japanese Web site that serves content to its employees using the ShiftJIS encoding may want to change its SilverStream server’s encoding property to SJIS.

➤ To change from the default utf-8 to another encoding:

1. Add the following line to the httpd.props file (located in the resources directory beneath the SilverStream root directory):

```
http-server.com.sssw.srv.international.UrlEncoding=NewEncoding
```
2. Enter the language mapping needed at your site in place of the *NewEncoding* variable. Check the Sun Web site if you are unsure about the Java string mapping for your language.
3. Restart the SilverStream server.
URL content will be encoded using the new encoding scheme **after** you restart the server.

Running the JMS (jBroker MQ) server

The SilverStream eXtend Application Server includes jBroker MQ for its JMS (Java Message Service) implementation. That means jBroker MQ provides the JMS server that runs with the SilverStream server to support messaging in your J2EE applications.

This section describes some things you need to know about using this JMS server with the SilverStream server:

- Starting the JMS server
- Using JMS servers in clusters
- Displaying JMS debug messages



To learn more about jBroker MQ, see the jBroker MQ Help.

Starting the JMS server

You can start the JMS server in either of the following ways:

Method	How it works
Automatic	<p>When the SilverStream server starts, it can check for the JMS server and automatically start it if necessary. To use this approach, you need to make sure the SilverStream server's <code>httpd.props</code> file specifies this property setting:</p> <pre>http-server.com.sssw.srv.jmsServerLaunch=true</pre> <p>When you install the SilverStream server, the installation program asks if you want to configure jBroker MQ, then sets this property according to your response. If you later change your mind, you can edit the <code>httpd.props</code> file yourself to specify <code>true</code> or <code>false</code>.</p> <p>By default, the installation program sets the <code>jspServerLaunch</code> property to <code>true</code>. But note that this property defaults to <code>false</code> if you remove it from the <code>httpd.props</code> file.</p> <p>In the automatic approach, the SilverStream server launches the JMS server as a child process. As a result, the JMS server will terminate when the SilverStream server terminates.</p>
Manual	<p>You can start the JMS server yourself, as described in the jBroker MQ Help.</p> <p>If you start the JMS server manually before starting the SilverStream server, the SilverStream server won't try to start the JMS server (regardless of the <code>jspServerLaunch</code> property setting).</p>

Using JMS servers in clusters

You can increase the reliability of JMS servers in your environment through the use of clusters. Here are some common configurations:

Configuration	What you do
Clustered SilverStream servers with individual JMS servers	Set up a cluster of SilverStream servers following the instructions in Chapter 12, “Administering a Cluster”. By default, each SilverStream server in the cluster has its own local JMS server.
Clustered SilverStream servers accessing clustered JMS servers	<ol style="list-style-type: none"> 1. Set up a cluster of SilverStream servers following the instructions in Chapter 12, “Administering a Cluster”. 2. Set up a cluster of JMS servers following the instructions in the jBroker MQ Help. You’ll need to manually edit the msgsvc.properties file installed for each SilverStream server (look in the jBroker MQ lib directory).

Displaying JMS debug messages

You can troubleshoot JMS-related problems at runtime by displaying debug messages to the SilverStream server console. To turn on basic JMS debugging, edit the SilverStream server’s `httpd.props` file to specify this property setting:

```
http-server.com.sssw.srv.jms.debug=1
```

For more detailed JMS debugging, specify a number greater than 1 for this property. To disable JMS debugging, specify 0 (the default).

6

Setting Up Users and Groups

This chapter describes how to define **Silver Security** users and groups—users and groups known only to SilverStream.

It contains these sections:

- About Silver Security users and groups
- Managing Silver Security users and groups
- Using the Locksmith privilege

NOTE SilverStream also provides access to external security providers, including Windows NT, LDAP, NIS+, and certificate issuers. For information about setting up access to users and groups from these providers, see “Accessing security provider systems” on page 208.

About Silver Security users and groups

You can define Silver Security users and groups in many ways. For example, you might want to define groups based on your site’s organization—such as Accounting, Sales, and so on—and assign users to those groups. The groups can contain Silver Security users as well as users defined in external security realms. Users can belong to multiple groups.

After you define Silver Security users and groups, you can define access to any directories or objects in the system based on the Silver Server users and groups. For example, you might want to set certain permissions for members of the Accounting group and other permissions for members of the Developers group.



For more information about using users and groups to set data permissions, see “Authorization and access control” on page 271.

Two predefined groups After installation, the SilverStream server provides two predefined groups: Administrators and Developers. Both groups initially contain only the server administrator. Use these groups as a starting point for creating your own users and groups. If you want to use names that differ from the predefined group names, you can rename and then delete them. For more information, see “Managing Silver Security users and groups” on page 130.

Group	Description
Administrators	After installation, the server administrator is the only member of this group. This person is initially the only one with the Locksmith privilege (that includes the ability to add new users and groups). See “Using the Locksmith privilege” on page 134. Add to this group any users that you want to be able to perform administration tasks to this group. You can assign users in this group all or a subset of all administration permissions. To administer the server, users need to be assigned Modify Server Configuration access. See “Administrative server permissions” on page 272.
Developers	After installation, the only privilege users in this group have (compared to users not part of the Administrators group) is the ability to browse directory listings.

Case sensitivity Silver Security user names and passwords are case sensitive as follows:

- **User names** are case-insensitive if the SilverMaster database is case-insensitive (for example, **administrator** and **Administrator** are considered the same name). User names are case-sensitive if SilverMaster is case-sensitive.
- **Passwords** are always case-sensitive. For example, **admin** and **Admin** are always considered different passwords.



For more information, see “Default group permissions” on page 288.

About your administrator account

Your administrator account can be assigned to any user recognized by SilverStream security (SilverStream, NT, LDAP, NIS+, or Certificate user).

When you installed the SilverStream server, you specified the user name and password for the SilverStream administrator account. This account was used when the new SilverMaster database catalog was created for SilverStream system management.

You use the server administrator account to log in to the SMC to administer the SilverStream server. You also need to specify the server administrator account to run the Designer and some of the SilverMasterInit command-line options.

The server administrator user account is part of the predefined Administrators group and has the Locksmith privilege. The Locksmith privilege provides Set Permissions privileges to any object on the server. Only accounts with the Locksmith privilege are able to assign Locksmith privilege to another account.



For more information, see “Using the Locksmith privilege” on page 134.

NOTE The **server** administrator account, which restricts who can log in and administer the SilverStream server, is distinct from the **database** administrator account. The SilverStream server uses the database administration account when connecting to the SilverMaster database. The only time you need to specify the SilverMaster database account is when you are running SilverMasterInit at the command line.

➤ To create a new administrator account:

1. Log in to the SMC using the existing Administrator account.
2. Create a new administrator account or select an existing user from one of the security realms to be the administrator.
3. Click **Properties** and assign the new account **Locksmith** privilege.
4. Add the new administrator account to the **Administrators** group.
5. Close the SMC.
6. Restart the SMC and log in as the new administrator.
7. Verify (using the **Properties** dialog) that the new account has **Locksmith** privilege.
8. (Optional) Delete the older Administrator account.

Managing Silver Security users and groups

You can use the SMC to add Silver Security users, edit user properties, and add Silver Security groups.

NOTE You can also perform these tasks using SilverCmd. For more information, see SetUserGroupInfo in the SilverCmd Reference of the *Facilities Guide* of the SilverStream server's Core Help.

Adding Silver Security users

➤ **To add a user:**

1. Start the SMC.
2. Select the **Security** icon from the toolbar.
3. Select **Users & Groups**.
4. Expand **Silver Security** and select **Users**.
5. Choose the **New User** icon at the bottom of the right pane:



You are asked whether you want to define a SilverStream user or a certificate user.

6. Select **SilverStream user** and click **Next**.



For information on defining certificate users, see “Manually installing client certificates” on page 260.

The New User panel displays.

A screenshot of the 'New User' dialog box. It has a title bar with 'New User' and a close button. The dialog contains five text input fields: 'Name:', 'Full Name:', 'Description:', 'Password:', and 'Confirm password:'. At the bottom, there is a blue bar with three buttons: '< Back', 'Finish', and 'Cancel'.

7. Type in the appropriate information in each field.
The Name field specifies the short name for the user. This is the name the user types in the Login box.
8. After completing the panel, click **Finish**.

Editing user properties

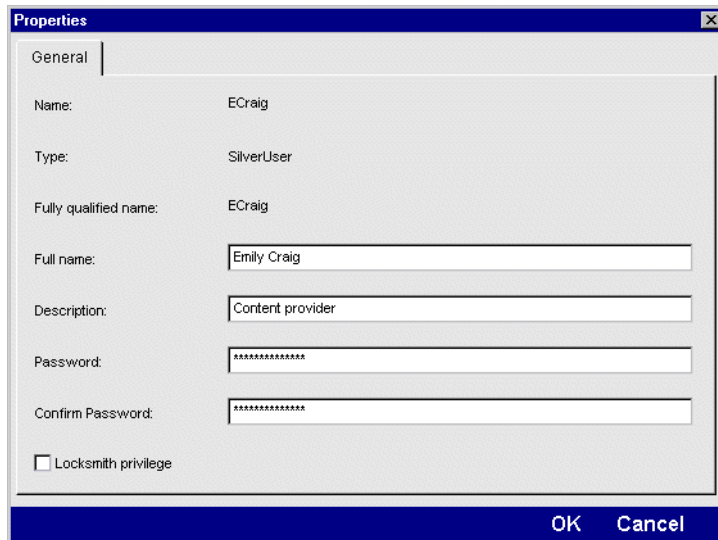
You can use the SMC to change user properties. (For users defined in external security providers, the only editable property is the Locksmith privilege; for more information, see “Using the Locksmith privilege” on page 134.)

Not allowing users to modify their properties By default, users can change their own user properties. You can turn off this privilege. For more information about this privilege, see “Enabling authentication” on page 266.

➤ **To edit user properties:**

1. Start the SMC.
2. Select the **Security** icon from the toolbar.
3. Select **Users & Groups**.
4. Expand the **Silver Security** list of users.
5. Highlight a user name and choose **Properties**.

The following panel displays.



The screenshot shows a dialog box titled "Properties" with a "General" tab. The dialog contains the following fields and options:

- Name: ECraig
- Type: SilverUser
- Fully qualified name: ECraig
- Full name: Emily Craig
- Description: Content provider
- Password: [Redacted]
- Confirm Password: [Redacted]
- Locksmith privilege

At the bottom of the dialog are "OK" and "Cancel" buttons.

6. Modify any of the four editable fields.

The Fully Qualified Name field corresponds to the Name field used to create the user. This field is not editable.

If you have Locksmith privilege, you can also change whether the user you are modifying has Locksmith privilege.



For more information, see “Using the Locksmith privilege” on page 134.

7. Click **OK**.

Adding Silver Security groups

Creating groups helps streamline security administration by allowing you to categorize users within a larger context, such as a business organizational unit or a work role. A user can belong to one or more user groups within a SilverStream database, and can be granted access to objects by group or individual status.

➤ To create a group:

1. Start the SMC.
2. Select the **Security** icon from the toolbar.
3. Select **Users & Groups**.
4. Expand **Silver Security** and select **Groups**.
5. Choose the **New Group** icon:



The following panel displays.

The image shows a dialog box titled "New Group". It contains two text input fields: "Name:" and "Description:". At the bottom of the dialog box are two buttons: "OK" and "Cancel".

6. Enter a name and a description for the group.
7. Click **OK**.

➤ **To add users to a group:**

1. Start the SMC.
2. Select the **Security** icon from the toolbar.
3. Select **Users & Groups**.
4. Expand **Silver Security** and expand **Groups**.
5. Select the SilverStream group to which you want to add users.
6. Choose the **Add Users to Group** icon:



The following panel displays.



NOTE Your panel may look different depending on which external security providers you have configured and the operating system used by the SilverStream server. For more information, see “Accessing security provider systems” on page 208.

7. To add a user to the group, select the user in the left panel, then choose **Add**.
You can add users defined by external security providers, such as NT domains, to Silver Security groups.
8. To remove a selected user, choose **Clear**.
9. To remove all users in the group, choose **Clear All**.
10. When finished, click **Close**.

Using the Locksmith privilege

The SilverStream-defined user Administrator has the Locksmith privilege by default. The Locksmith privilege allows users to do the following:

- Get and set data access permissions, even if these permissions are denied elsewhere in the system (for example, if the user does not belong to a group for which set permission is allowed).



For information about defining security for the server and objects on the server, see “Authorization and access control” on page 271.

- Read server property settings from the SilverMaster database, even if this permission is denied elsewhere in the system.

Since the Locksmith privilege also allows you to set permissions, you can also give yourself server administrative permissions.

NOTE Locksmiths don’t have all permissions just by virtue of being Locksmiths; but as Locksmiths they can give themselves any permissions they want.

- Grant and revoke the Locksmith privilege for other users. See “Editing user properties” on page 131.

NOTE Since the Locksmith privilege provides powerful access to server functions and properties, you should limit it to yourself and other trusted users.

Keep at least one Locksmith Be careful not to delete all users with the Locksmith privilege: a user must have Locksmith privilege to grant it to someone else. So if no one has Locksmith privilege, it cannot be granted.

If you find yourself in that situation, you can run SilverMasterInit with the **-l** command-line option to define a Locksmith account.



For more information, see “Using the SilverMasterInit program” on page 416.

7

Maintaining the Server

This chapter describes how to perform some typical server maintenance tasks. It has sections on the following:

- Administering a SilverStream server remotely
- Managing licenses
- Setting the AGCLASSPATH variable
- Maintaining deployed J2EE objects
- Managing J2EE transactions
- Monitoring server activity
- Integrating with existing Web servers
- Setting up mail on the server
- Setting Fulcrum full-text properties

Administering a SilverStream server remotely

You can use the SilverStream Management Console (SMC) to remotely administer servers. You can administer as many servers as you want from one SMC console. If you are running a server in a cluster, you can choose additional server clusters to administer. For more information about server clusters, see “Server clustering” on page 327.


➤ **To administer a server:**

1. Verify that the server you want to administer is up and running.
2. Start the SMC.
3. Choose the **Choose** (server) icon.

The Add Server dialog displays.



4. Specify *server:port*, where:

Parameter	Description
server	The name of the server (such as localhost or <code>http://hostname</code>).
port	The administration port. You need to specify the port only if it is not the default port (port 80).  For more information about ports used by the server, see “Setting up separate ports” on page 106.

5. Click **OK**.
6. Use the SMC to administer the server.

Managing licenses

This section describes how to install or delete SilverStream user licenses.

About licenses

SilverStream uses a license key to administer access to the server. License management is integrated with session management. Each user connection requires a license. The license is held for the duration of the connection.

License keys:

- Have limits on either users or CPUs
- Are tied to a server machine host ID

You receive an initial encrypted license key when you buy your product. Each product you purchase comes with a certain type of license. You use the Installation program to add this license key to your server. This information is stored in the database.

You may only need the initial user-based license key during your application development cycle. However, when it comes time to deploy your application and go live, you will probably want to call SilverStream and purchase CPU-based licenses for your application server.

If you have a user-based license and your application is put into production, when the number of simultaneous users increase you may see a message like **Maximum server sessions exceeded**.

You should add new licenses through the SilverStream eXtend Application Server Installation program.

When you have multiple licenses installed, the SilverStream server will use the least restrictive license it finds. For example, if you have installed both a single-user license and a CPU-based license, SilverStream uses the CPU-based license.

➤ **To add licenses (NT):**

1. Shut down the SilverStream server.
2. Run the SilverStream eXtend Application Server Installation program.
3. From the Setup Function screen, select **Add a new server license**.
The License Information screen appears.
4. Fill in the license serial number.
TIP If you receive this long encrypted number electronically, paste it into the License number field to avoid typing errors.
5. Specify the requested information for your SilverMaster, where the license information is stored.
6. Click **Finish**.
The SilverStream server uses the license number and other information you enter to generate an encrypted license key.

➤ **To add licenses (UNIX):**

1. Shut down the SilverStream server.
2. Run the **upldlic.sh** utility, which is at the root directory on the eXtend Application Server CD.
3. When prompted, specify where the SilverStream server is installed and what the license string is.
SilverStream generates an encrypted license key.

➤ **To access or delete a license:**

1. Start the SMC.
2. Select the **Configuration** icon from the toolbar.
3. Select **Licenses**.
The license form displays information about your license.
4. To delete a license, select it and then select **Remove License**.

Setting the AGCLASSPATH variable

The SilverStream server supports an environment variable called **AGCLASSPATH**, which allows you to extend the Java classes used by your application. You can use **AGCLASSPATH**, for example, when you want to include third-party elements such as database drivers with your application. Use this variable instead of the **+cp** Java class path option described in “Starting the SilverStream server” on page 91. The SilverStream server overrides the **CLASSPATH** variable.

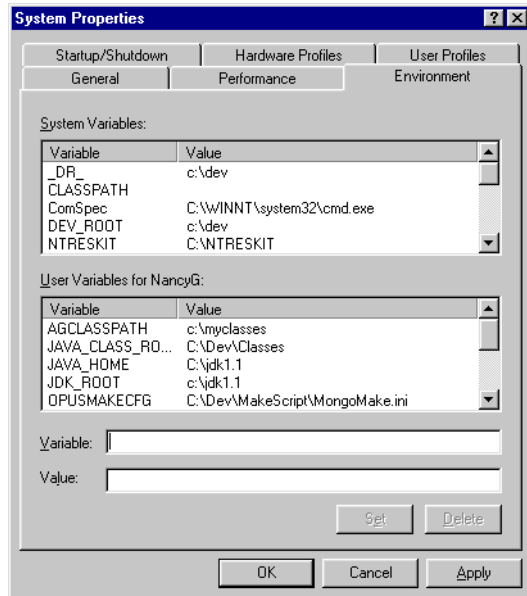
NOTE When deploying business objects using created Java classes, SilverStream developers should create JAR files using the SilverStream Designer rather than by setting **AGCLASSPATH**. For more information, see the *Programmer’s Guide* in the server’s Classic Development Help.

In UNIX, set the environment variable **AGCLASSPATH** following the appropriate procedure for the shell you are using. For NT, use the following procedure.

➤ To set **AGCLASSPATH** in NT:

1. From the NT Start menu, choose **Settings>Control Panel**.
2. Choose the **System** icon, then select the **Environment** tab.

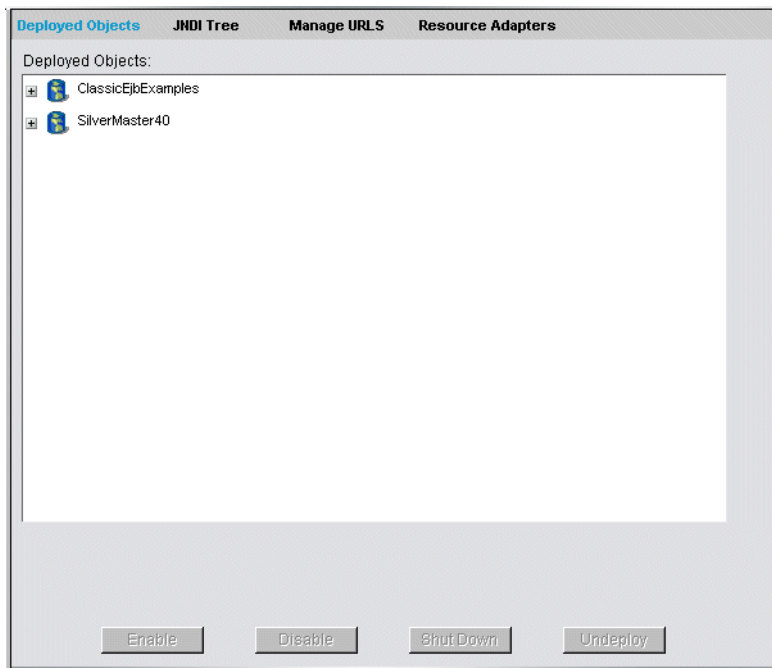
The following dialog displays.



3. Type **AGCLASSPATH** in the Variable field, then type a path in the **Value** field.
The screen above shows AGCLASSPATH with a value of **c:\myclasses**.
4. Click **Set**.

Maintaining deployed J2EE objects

The Deployment options section of the SMC provides information about the J2EE objects that have been deployed on the server.



The following table describes the functionality for each of the panels.

Use this panel	To perform this function
Deployed Objects	Manage J2EE deployed applications like EARs, WARs, EJB JARs, application client JARs, and RARs
JNDI tree	View the RMI JDNI tree for any SilverStream server currently managed by the SMC or view the InitialContext of any server available on to the SMC via the network.
Default URLs	<p>Use this panel to specify a default page for a database or server.</p> <p>Database URL—The page that displays when the user requests the URL for the database, such as <code>http://localhost/MyApp/</code>. If there is no default database page defined, the user would see a directory listing for the database (if this is allowed by the administrator). For SilverStream classic applications, this is also the page that displays when you select the Run Database icon in the Main Designer.</p> <p>Server URL—The page that displays when the user requests the URL for the server, such as <code>http://localhost/</code>. If there is no default server page defined, the user would see a directory listing for the server (if this is allowed by the administrator).</p>
Resource Adapters	View information about deployed resource adapters and their settings.

Using the Deployed Objects panel

The **Deployed Objects** panel lists:

- **EJB JAR** files containing Enterprise JavaBeans (EJBs)
- **WAR** files containing JSPs and servlets
- **EAR** files that package other archive files into a full application
- J2EE application-client deployments



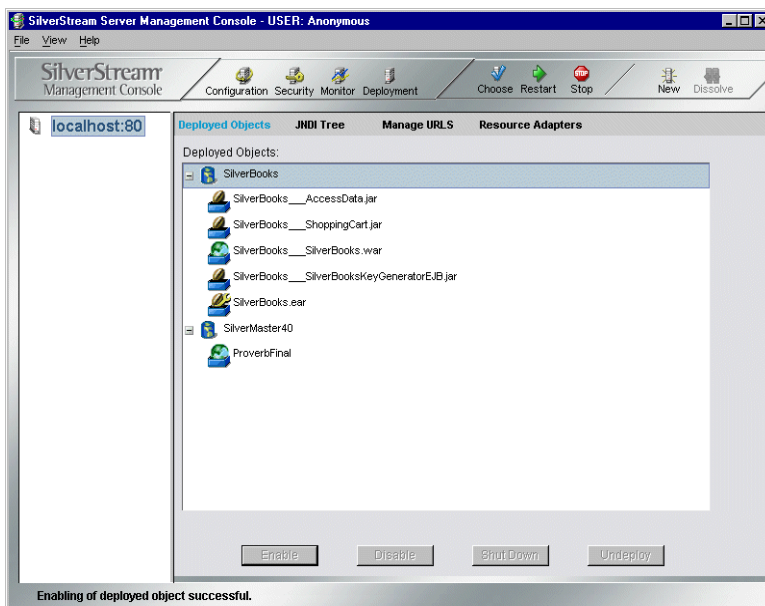
For more information, see the J2EE deployment chapter in the *Facilities Guide*.

➤ **To manage deployed J2EE objects:**

1. Start the SMC.
2. Select the **Deployment** icon from the toolbar.
3. Select **Deployed Objects**.
4. Expand the database containing the deployed objects you want to manage.

You can **Undeploy** WARs, EARs, CARs, and RARs

You can **Enable, Disable, Shutdown, and Undeploy** EJB JARs.



5. Select a deployed object and perform one of the following actions.

Action	Description
Enable	For EJB JARs only. Enables a disabled EJB JAR. If a JAR is enabled, the beans in the JAR are available.
Disable	For EJB JARs only. Disables an enabled EJB JAR. If a JAR is disabled, none of the beans in the JAR are available. Disabling a JAR stops any running EJBs in the JAR. A disabled JAR remains disabled until it is explicitly enabled.
Shut Down	For EJB JARs only. Shuts down the JAR and all its beans for the current server session. When a JAR is shut down, none of the beans in the JAR are available. When the server is restarted, enabled JARs that had been shut down in the previous server session are again available.
Undeploy	Shuts down and removes deployed objects from the server.

➤ **To view deployed RARs:**

1. Start the SMC.
2. Select the **Deployment** icon from the toolbar.
3. Select **Resource Adapters**.
The deployed resource adapters are displayed in the dropdown.
4. Select a resource adapter from the list to display the adapter's settings.

➤ **To view the JNDI tree:**

1. Start the SMC.
2. Select the **Deployment** icon from the toolbar.
3. Select **JNDI tree**.

4. Choose the radio button for the function you want to perform.

Radio button	Description
RMI	Displays the RMI JNDI tree on the selected server.
Specify URL	Displays the InitialContext for any server available to the SMC via the network. To view the InitialContext for an LDAP server named beetle, you would enter the following: <code>ldap://fleabeetle/dc=silverstream.com</code>

5. If you chose Specify URL, click **Submit**.

➤ **To specify the default database URL:**

1. Start the SMC.
2. Select the **Deployment** icon from the toolbar.
3. Select **Manage URLs**.
4. Select the **Database URL** radio button.
5. Choose the database whose default page you want to set from the Database dropdown.
6. Enter an URL in the URL text box and click **set Default URL**.
The URL should be a database-relative URL.

➤ **To specify the default server URL:**

1. Start the SMC.
2. Select the **Deployment** icon from the toolbar.
3. Select **Manage URLs**.
4. Select the **Server URL** radio button (it is selected by default).
5. Choose a database from the Database dropdown.
6. Enter an URL in the URL text box and click **set Default URL**.
The URL should be a server relative URL and should include the database name (the same as chosen from the database dropdown).

Managing J2EE transactions

The SilverStream server supports J2EE transactions via the jBroker Transaction Manager (jBroker TM). The jBroker TM is the transaction service for the jBroker ORB. It provides an implementation of the JTA TransactionManager and UserTransaction interfaces. These interfaces represent the contract between the transaction manager and the application server, and between the transaction manager and user applications.

You can use the SMC to specify settings for the jBroker TM transaction log, the resources that jBroker TM uses to recover transactions, and so on.

How jBroker TM recovers transactions Transactions need to be recovered when the server suffers a catastrophic error and needs to be restarted. Here is the process that the jBroker TM performs at server restart:

1. The jBroker TM reads the transaction log file.
2. It determines that a transaction needs to be recovered when items in the log have been prepared for a transaction but the transaction did not complete (in other words, the transaction did not commit or rollback).
3. The TM creates a number of worker threads (the number is specified via the SMC as described in the procedure below). These worker threads are used to recover the incomplete transactions.
4. If a transaction includes access to a remote resource (like a CORBA resource), the worker threads will attempt to access those same resources. If it is unable to locate a remote resource, the worker thread will sleep for a specified amount of time (the Resource recovery retry time limit specified in the SMC) before it attempts to access the resource again. In the meantime, the other recovery worker threads will be working on other incomplete transactions in the log.

➤ **To specify jBroker Transaction Manager settings:**

1. Start the SMC.
2. Select the **Configuration** icon from the toolbar.
3. Select **Advanced**.

4. Select the **Transactions** tab.

The screenshot shows the Administration console with the following configuration for the Transactions tab:

- Preallocate log files when creating
- Log file max. size (kb):
- Transaction timeout (seconds):
- Resource recovery retry time limit (minutes):
- Recovery worker threads:
- Server logging: _____
- Log output: _____
- JTS log file directory:

An **Update** button is located at the bottom right of the configuration area.

5. Specify the settings as follows:

Field	Description
Preallocate log files when creating	Specifies whether to preallocate the transaction log file.
Log file max size (kb)	The maximum size of the transaction log file in KB. When the maximum file size is reached, the Transaction Manager attempts to create a new log file.

Transaction timeout (seconds)	<p>The amount of time allowed for all transactions managed by the Transaction Manager to complete. The timer begins when the transaction starts.</p> <p>If the transaction does not complete before the transaction timeout period ends, jBroker TM will roll back the transaction.</p> <p>Developers can override this value for specific transactions using the UserTransaction interface.</p>
Resource recovery retry time limit (minutes)	<p>If a worker thread is unable to obtain a remote resource during a transaction recovery, this setting specifies the amount of time a worker thread will sleep before it attempts to access the remote resource again.</p>
Recovery worker threads	<p>The number of threads jBroker TM should create to process a log file during a recovery. The higher the number of recovery worker threads the faster the recovery. However, since there are unlikely to be a large number of transactions to recover, the recovery will probably not take a lot of time. These threads are destroyed after the recovery completes.</p>
JTS log file directory	<p>The disk location for the transaction log file.</p>

6. Click **Update**.
7. To activate the new settings, click the **Restart** button.

Monitoring server activity

The SMC provides several options that allow you to easily monitor server activity. These options are described in the following sections:

- Displaying charts of server activity
- Displaying logs
- Displaying views of server statistics

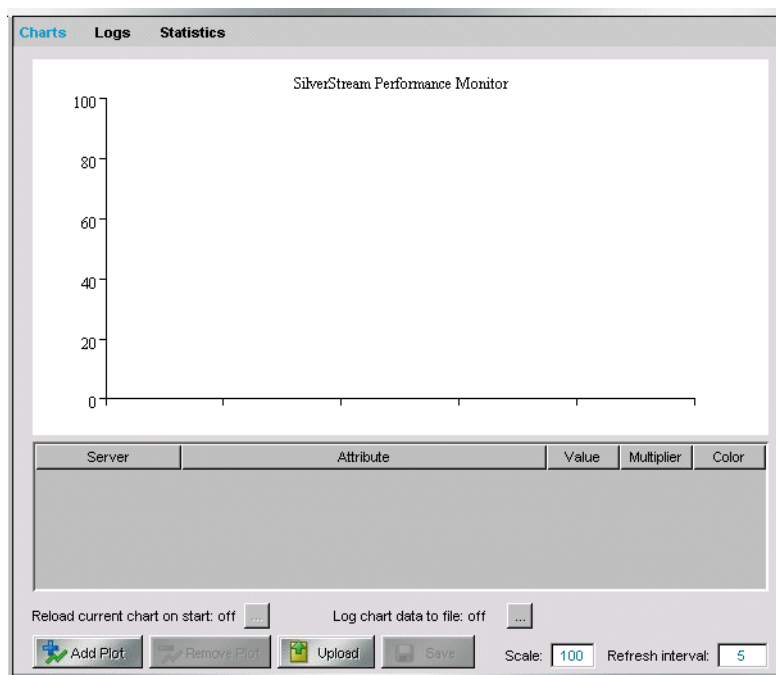
Displaying charts of server activity

You can display real-time charts of various server statistics.

➤ To display a chart of server activity:

1. Select the server or cluster you are administering.
2. Select the **Monitor** icon from the toolbar.
3. Select **Charts**.

An empty chart displays.



4. Click **Add Plot**.

The Add Plot dialog displays, allowing you to choose which statistics you want to chart. The statistics are divided into categories.

5. Select a statistic you want to chart and click **Add**. Statistics marked with $\uparrow\downarrow$ are count values; statistics marked with Δ record change in values.

The statistic is added to the table below the chart. The current value of the statistic is shown, as is the color of the plot line that will be generated.

6. Select additional statistics if you want, clicking **Add** after each one.
7. When you have selected all the statistics you are interested in, click **Close**.

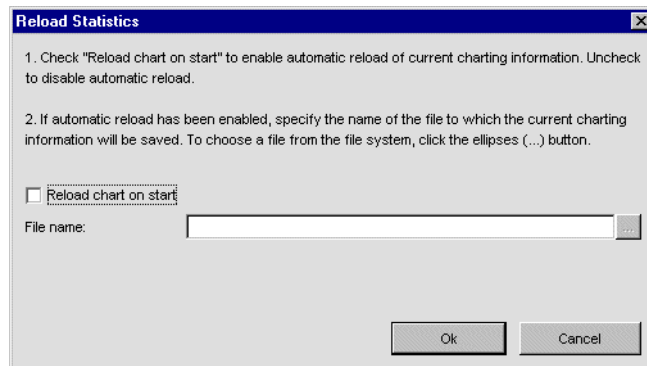
What happens The SilverStream server plots all statistics you have specified and displays the exact values in the table below the chart. By default, the values are refreshed every five seconds. (To change this, enter a new number in the Refresh Interval box.)

Saving statistics settings for automatic reload

By default, the statistics you are plotting are not saved and reloaded between sessions of the SMC.

➤ To save statistics settings:

1. Click the **Reload current chart on start** button at the bottom of the Charts panel. The Reload Statistics dialog displays.



2. Check the **Reload current chart on start** check box.
3. Specify a file name or choose the ellipses to choose a file from the file system.
4. Click **OK**.

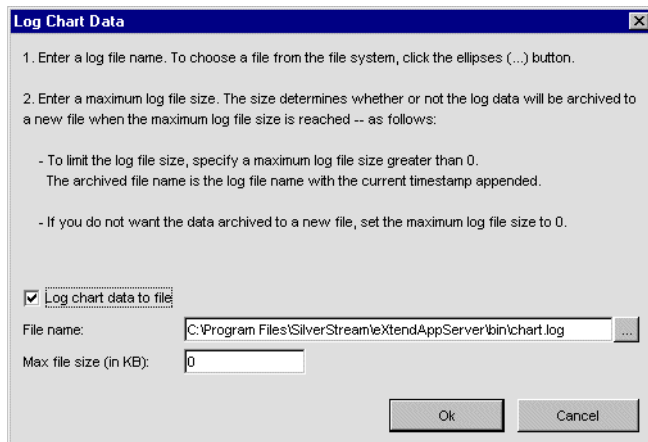
The charting data is saved to an XML file that is read each time you restart the SMC.

Saving chart data

By default, the statistics that you chart are not saved.

➤ To save chart data:

1. Click the **Log chart data to file** button at the bottom of the Charts panel. The Log Chart Data dialog displays.



2. Check the **Log chart data to file** check box.
3. Specify a file name or choose the ellipses to choose a file from the file system.
4. Specify the log file size or specify 0.

If you specify a log file size: when that file size is reached, the data is dumped to a file of the same name—but with the timestamp appended (to make the files unique).

If you specify 0: there is no limit on how large this file may grow (except the limits imposed by the file system).
5. Click **OK**.

The data is saved to a tab-delimited file.

Changing the scale and refresh rate

You can change the scale of the Y-axis by specifying a new value in the **Scale** field (the default is 100). You can also change how often the SilverStream server updates the values by changing the value in the **Refresh interval** field (the default is every five seconds).

Removing a plot

➤ To remove a plot:

- Select a plot and click **Remove plot**.

Editing a plot

It may be that you are plotting different statistics with very different scales but want them to appear clearly in the chart. To do this, you can change the multiplier for particular plots to equalize the values and make the chart easier to read.

➤ To edit a plot:

1. Select **the multiplier column** in the row of the statistics that you want to edit.
2. Select a **multiplier value** from the dropdown

Saving a statistics set

Once you have charted a set of statistics that you are interested in, you can save the specification of the statistics set in a file so that you can easily view the set of statistics later.

NOTE The file stores the list of statistics that are plotted but does not store the statistics' values.

➤ To save a statistics set:

1. Display the statistics you want to save as a set.
2. Click **Save**.
The Save dialog displays.
3. Specify the file you want to save the statistics set in. If you don't provide an extension, SilverStream gives the file the extension XML.
4. Click **Save**.
You can view the statistics set later by clicking **Load**, as described next.

Viewing a statistics set

➤ To view a statistics set you have saved:

1. Click **Load**.
The Open dialog displays.
2. Select the file that defines the statistics set you want to view and click **Open**.

Displaying logs

If you have enabled server logging and are using the built-in SilverStream logging class to log to the database, you can display the log(s) in real time in the SMC. (If you are logging to a file or using a custom class to do the logging, you can't display the log in the SMC.) As an alternative, you can use the PrintLog SilverCmd.



For more information about server logging, see “Using server logging” on page 112.

➤ To display a log:

1. Select the server or cluster you are administering.
2. Select **Monitor** options.
3. Select the **Logs** panel.
4. Select the tab corresponding to the logging you have enabled and which you want to view.
The log displays.

What you can do You can:

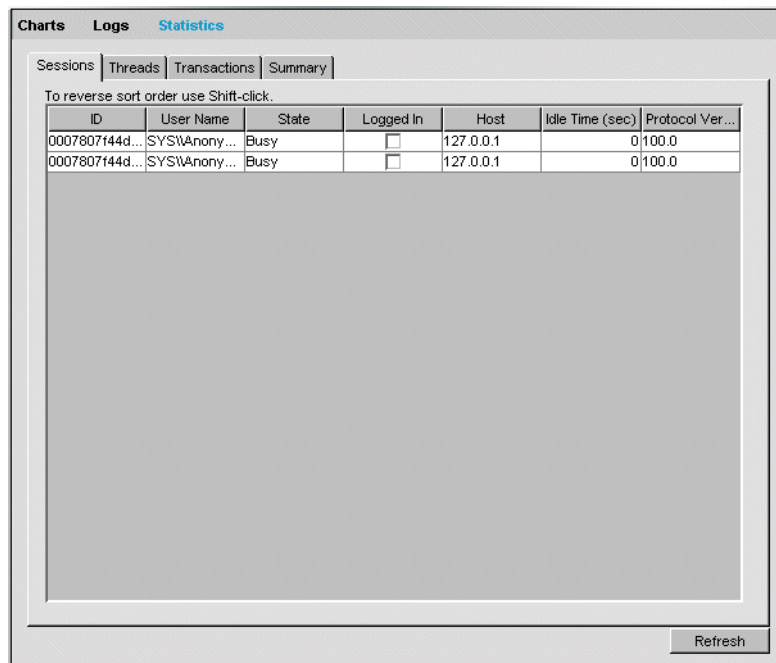
- Update the log by clicking **Refresh** (the log does not update automatically)
- Sort the log by clicking a column's header
- Resize a column by placing your mouse pointer on the column's right border and dragging the mouse
- To view the message, double-click on the row or click **Log Detail**

Displaying views of server statistics

You can display in the SMC tabular views of server statistics related to individual sessions and threads, as well as a summary of server activity.

➤ **To access server statistics in a view:**

1. Select the server or cluster you are administering.
2. Select the **Monitor** icon from the toolbar.
3. Select **Statistics**.



4. Select the tab for the category you want.
Statistics are updated dynamically.

What you can do You can:

- Force an update by clicking **Refresh**
- Resize a column by placing your mouse pointer on the column's right border and dragging the mouse

About the statistics The following sections describe the statistics that are displayed.

Session statistics

This tab displays statistics for each current client session.

Session statistic	Description
ID	Displays the session ID returned by a call to the AgiSession internal system table.
User name	Displays the user name of the person or entity logged on to this session. If unknown, displays Anonymous .
State	The state of the connection.
Logged in	Displays true (checked) if the user has logged in (for example, from the browser's login or the SilverStream Designer menu item). Otherwise, displays false (unchecked).
Host	Displays the host of the client source, if known.
Idle time	Displays the time elapsed (in seconds) since the last client connection on this session ended.
Protocol version	Displays the SilverStream protocol used for the session.

Viewing statistics in a browser You can also display these statistics in a browser. Point your browser to <http://server/SilverStream/Sessions>.

Thread statistics

This tab displays statistics for each server thread.

Charts Logs **Statistics**

Sessions Threads Transactions Summary

To reverse sort order use Shift-click.

Name	State	Start Date	Busy Time (sec)	Session
antiGC			0	
Thread-2			0	
rawStoreDaemon			0	
HTTP-Socket-Liste...	Awaiting Connection	2002.05.10 at 12:2...	41359	
Memory-Usage-Mo...	Waiting for work	2002.05.10 at 12:2...	0	
JBroker IOP Listener			0	
JBrokerJTS_TimeO...			0	
JBrokerJTS_LogMo...			0	
StoreSweeper	Idle	2002.05.10 at 12:2...	0	
Command-Shell			0	
Timer Manager			0	
Business Object Ti...			0	
JBroker Scavenger			0	
ClientPool watcher	Idle	2002.05.10 at 12:2...	0	
client0	Processing GET fo...	2002.05.10 at 12:2...	2783	0007807f44d0149...
client1	Awaiting request	2002.05.10 at 12:2...	60	0007807f44d0149...
client2	Free	2002.05.10 at 12:2...	0	
client3	Free	2002.05.10 at 12:2...	0	
client4	Free	2002.05.10 at 12:2...	0	
client5	Free	2002.05.10 at 12:2...	0	
client6	Free	2002.05.10 at 12:2...	0	
client7	Free	2002.05.10 at 12:2...	0	
client8	Free	2002.05.10 at 12:2...	0	
client9	Free	2002.05.10 at 12:2...	0	
client10	Free	2002.05.10 at 12:2...	0	

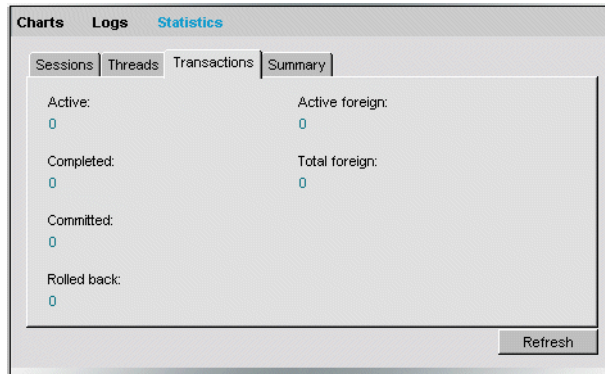
Refresh

Following is a description of each of the fields.

Thread statistic	Description
Name	<p>Nonclient threads are used for various internal tasks, such as cleaning up server data structures, indexing full-text tables, and so on. Client threads handle incoming requests. There should be at least as many client threads as there are MAXCLIENTCONNS listed in the SMC Connections panel (see “Client connection parameters” on page 307).</p> <p>ResultBuffer threads are prefetch buffer threads as defined in the SMC (see “Setting performance parameters” on page 303). They are dynamically added as needed, but should not exceed the number defined in the SMC.</p>
State	A brief description of the current state of the thread.
Start date	The start date for the thread. In many cases this date is the same as when the server was started. However, for dynamically allocated threads (such as prefetch buffers) this value will be different.
Busy time	The elapsed time (in seconds) since the thread has been actively working, as opposed to waiting. This value reflects how busy the server is in general.
Session	An internal session ID for this thread. See the Sessions tab to determine user/host information.

Transactions statistics

This tab provides statistics for transactions managed by jBroker TM (Transaction Manager).



Following is a description of each of the fields.

Transactions statistic	Description
Active	The number of active transactions this server is responsible for managing
Completed	The number of completed transactions
Committed	The number of committed transactions
Rolled back	The number of transactions rolled back
Timed out	The number of transactions timed out
Total	The sum of the active, rolled back and completed transactions (excludes foreign transactions).
Active foreign	The number of active foreign transactions. A foreign transaction is a transaction that was started in another process (such as a different server or client application) that was propagated to this server via call to an EJB. The foreign transaction is controlled by the other process.
Total foreign	The sum of the active, rolled back and completed foreign transactions.

Summary statistics

This tab provides access to different types of category summaries. The following describes the items for each tab selection.

- **Server statistics**

Server statistic	Description
Server load	Provides a scaled value, based on overall server activity.
Total number of hits since server was last started	The number of HTTP message requests the server has received since it started.
Date/time server was started	The date and time the server was started.
Number of bytes emitted by server	The number of bytes returned by the server since it started.

- **Request time statistics** The statistics for this tab are enabled only when HTTP logging is enabled. For a description of HTTP logging, see “Using server logging” on page 112.

Request time statistic	Description
Minimum request processing time	The minimum elapsed time to process any HTTP request, from when the server receives the header until it transfers the reply.
Minimum request URL	The URL of the request that took the least amount of processing time (see previous item).
Maximum request processing time	The maximum elapsed time to process any HTTP request, from when the server receives the header until it transfers the reply.
Maximum request URL	The URL of the request that took the most processing time (see previous item).
Mean request processing time	The mean (average) of the processing times for all requests received by the server.

- **Memory statistics** The statistics for this option apply to the Java Virtual Machine (JVM).

Memory statistic	Description
Free memory	The result for the Java call <code>Runtime.freeMemory()</code>
Total memory	The result of the Java call <code>RunTime.totalMemory()</code>
GC count	<p>The total number of times the Java Garbage Collector has run since the server has started.</p> <p>This number indicates how hard the server is being hit, how many objects are allocated, and how much memory pressure the server is under.</p>

- **Thread statistics.** This option provides statistics about Client threads, which handle incoming client connections.

Thread statistic	Description
Free thread count	The current number of threads not associated with a client connection and available for immediate use.
Idle thread count	The number of threads associated with a client connection, but not currently handling a user request.
Total thread count	<p>The total number of client threads allocated.</p> <p>This number should be equal to the Maximum number of client connections in the SMC Connections panel (see “Client connection parameters” on page 307).</p>

Viewing statistics in a browser You can also display most of these summary statistics in a browser. Point your browser at `http://server:port/SilverStream/Statistics`. The page updates itself automatically every five seconds.

Integrating with existing Web servers

SilverStream provides **Web server integration (WSI) modules** that allow you to redirect requests for selected pages served by your Web server to a SilverStream server. You can use these WSI modules to integrate a SilverStream server with your Web server.



For more information, see Chapter 8, “Using the Web Server Integration Modules”.

Setting up mail on the server

Handling mail received from clients is an important aspect of Web application development. SilverStream developers might write business objects that are triggered when mail is received on the server. To implement this type of object, the SilverStream server must be configured to poll for e-mail in one or more mailboxes from one or more POP3 or IMAP servers.

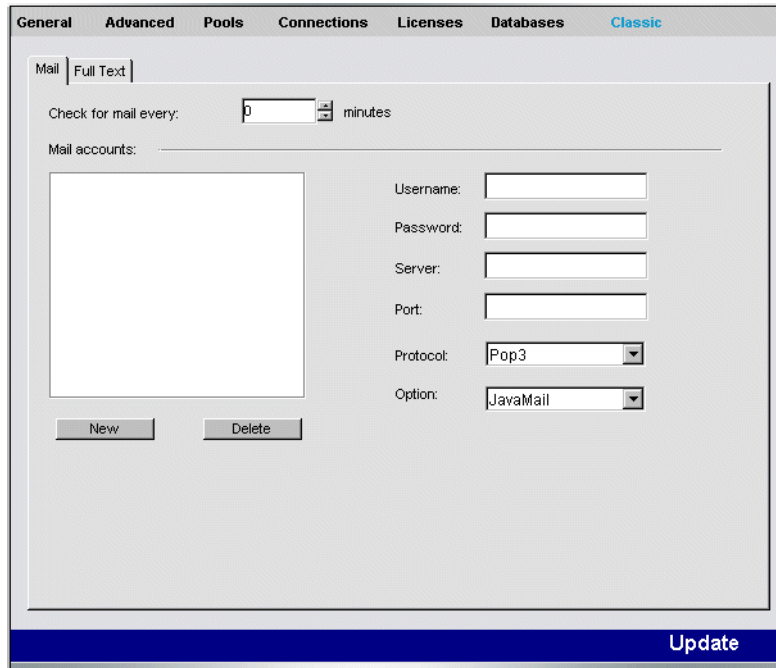
The Mail option on the SMC allows you to set up mailboxes for this purpose. For example, if a developer needs a business object to automate the maintenance of a mailing list in the database, you would first create a mail account for mailing list maintenance. Then the developer could write a business object that looks at e-mail received at that mailbox for subscribe and unsubscribe requests.

NOTE You can configure mail-triggered business objects messages in JavaMail format. The JavaMail format enables you to manipulate received messages and send new messages using the standard J2EE mail API.

➤ To poll for mail:

1. Start the SMC.
2. Select the **Configuration** icon from the toolbar.

3. Select **Classic**. (If the Classic panel is not visible, choose **View>Display Classic Settings**.)



4. Click **New** and enter data in the fields. You must provide data for all of the fields displayed on this tab.

Field	What to specify
Check for mail every <i>n</i> minutes	The time interval for the mail server to check for incoming mail messages. Enter a value or use the spinner control. When set to zero, the server does not check for mail.
Username	The e-mail ID that the server uses to receive mail. This name is like any other mail account name on your system. Make sure you set up a mail account specifically for the server and not for an individual user. The name is a POP3 or IMAP mail account, and it should be unique to the SilverStream server.
Password	The password for the mail account.

Field	What to specify
Server	The name of the mail server—for example: mail.myCompany.com
Port	The port number the mail server uses to receive mail. The default port number is 110 (the default for a POP3 server). The default port number for IMAP is 143.
Protocol	The mail protocol used to receive mail. The default is POP3. You may also choose IMAP.
Option	The format in which received messages are available: AgoMail or JavaMail. The default mail format is SilverStream AgoMail. You can only use AgoMail with Pop3. JavaMail can use IMAP or POP3.

5. Click **Update**.

The new settings take effect immediately.

Once enabled, the SilverStream server acts as a POP3 or IMAP client.

Setting Fulcrum full-text properties

You can use the SMC to set some administrative properties used by the Fulcrum SearchServer full-text search engine.



For more information about Fulcrum SearchServer and how to enable full-text search in SilverStream applications, see the full-text search chapter in the *Tools Guide* of the SilverStream server's Classic Development Help.

➤ **To set Fulcrum full-text properties:**

1. Start the SMC.
2. Select the **Configuration** icon on the toolbar.
3. Select **Classic**. (If the Classic panel is not visible, choose **View>Display Classic Settings**.)


4. Select the **Full Text** tab.

The screenshot shows a configuration window with the following elements:

- Tabs: General, Advanced, Pools, Connections, Licenses, Databases, Classic (selected)
- Sub-tabs: Mail, Full Text (selected)
- Indexer interval (minutes):
- Maximum result row count:
- Enable partial indexing at startup (Disabled = full re-index)
- Disable indexing before full-text search
- Update button (bottom right)

5. Specify values as follows.

Property	Description
Indexer interval	<p>Specifies how often Fulcrum SearchServer checks and performs incremental indexing for rows whose full-text-search columns have been modified.</p> <p>The default is 20 minutes.</p> <p>NOTE Indexing is forced before a query against a modified table is processed.</p>
Maximum result row count	<p>Maximum number of hits Fulcrum will return for a full text search query.</p> <p>The default is 1000. Set this value to 0 if you don't want any limit to the number of hits returned.</p>

Property	Description
Enable partial indexing at startup	If this property is selected, SilverStream will not reindex tables marked for full-text search at server startup time.  For more information, see “Indexing full-text-search tables” below.
Disable indexing before full-text search	If this property is selected, SilverStream will not perform any indexing before a full-text-search even if columns have been modified. This property is useful when you have large tables that don’t often change that you want to search. By default, this property is not set.

6. Click **Update**.

Indexing full-text-search tables

If the **Enable partial indexing at startup** property is not selected, at startup time the SilverStream server deletes all previous full-text indexing from all tables that have full index search enabled, then reindexes all the tables from scratch. After the initial indexing, the tables are marked for incremental indexing only when updates are made to them within SilverStream.

Full reindexing of tables consumes resources unnecessarily if the tables are never updated outside SilverStream.

You can select the **Enable partial indexing at startup** property (which is selected by default) to turn off full reindexing when the server starts. This means that at server startup, previous indexes will be maintained, tables will be marked for incremental indexing, and incremental indexing for SilverStream-modified tables will continue.

If **Enable partial indexing at startup** (or **Disable indexing before full-text search**) is not selected, full-text search tables will be indexed from scratch each time the server starts.

NOTE If you set **Enable partial indexing at startup** but still have some tables that are updated outside SilverStream, you can specify full indexing for these tables: open the SilverStream Business Object Designer and create a business object that will fire when the server is started. Add the following line of code for each table that you want to be reindexed:

```
com.ssw.srv.agents.AgFullText.index("db", "table");
```

8

Using the Web Server Integration Modules

The SilverStream server provides **Web server integration (WSI)** modules for Microsoft's IIS and Netscape's iPlanet Enterprise servers that run on AIX, HP-UX, Solaris, and Windows. The SilverStream WSI modules allow you to integrate a SilverStream server into your existing IIS or iPlanet Web server framework.

This chapter describes how to integrate the SilverStream server with your existing Web servers using the SilverStream WSI modules. It includes the following topics:

- About the WSI modules
- Planning your WSI installation
- Installing and configuring the WSI module
- About the WSI configuration file
- Enabling the WSI module for IIS for Windows
- Enabling the WSI module for iPlanet for Windows and UNIX platforms
- Redirecting requests to multiple SilverStream servers
- About connection pooling
- Authentication and security integration considerations
- enCommerce getAccess integration

Web server versions See the *Release Notes* for the supported Web server versions.

About the WSI modules

URL requests sent to the Web server are forwarded by the WSI module to the SilverStream server. The WSI then returns the URL reply from the SilverStream server to the requesting browser.

The WSIs extend the URL namespace of your existing Web server directory structure by allowing you to include dynamic content in your existing Web applications. URL masking means users cannot tell when they switch between static content on the Web server and dynamic content on the application server.

How the WSI modules work

With a WSI module, the SilverStream server services a portion of your Web server's URL namespace. The SilverStream WSI module services requests for specified URLs to the Web server by opening HTTP connections to a SilverStream server and forwarding the requests to it. The WSI then returns the response. You specify which URLs the WSI module will forward using a configuration file that the WSI reads when the Web server starts.

To improve response time, the WSI module will reuse socket connections between itself and the SilverStream server. The WSI maintains a connection pool to the SilverStream server that reuses these connections as needed.

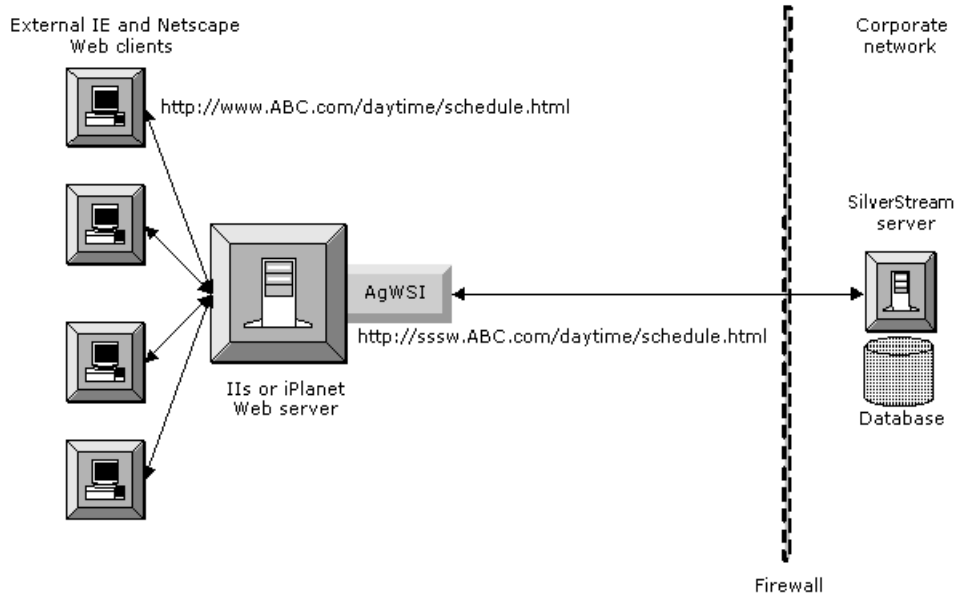
With the WSI module, there is no direct communication between the browser and the SilverStream server: all calls pass through the WSI.

WSI module examples

The following examples show how a WSI might be used in a corporate network.

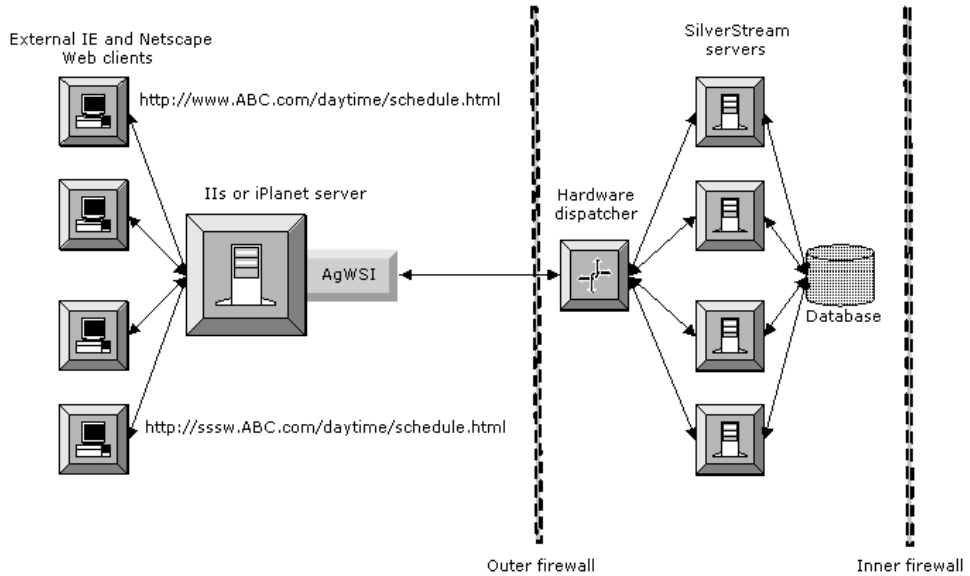
- In the first four examples, the WSI module forwards a client request for **http://www.ABC.com/daytime/schedule.html** to **http://sssw.ABC.com/daytime/schedule.html** (even though the URL address of the client would not change).
Together the examples show ways to reliably handle an increasing volume of requests.
- The last example shows a configuration where client requests are forwarded to different SilverStream servers, depending on the request.

Web server with one SilverStream server In this simple scenario, the WSI forwards requests from a Web server to a single SilverStream server.

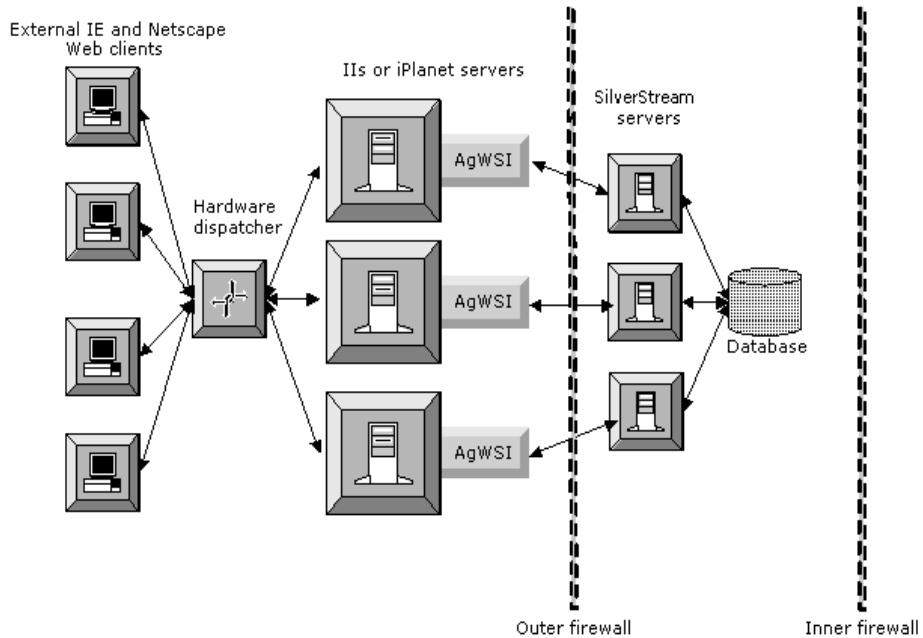


NOTE There are many ways to set up your environment. For example, in each of the following figures, you could position your database behind the inner firewall. Positioning the corporate database outside the DMZ (the area between the two firewalls) helps protect your database.

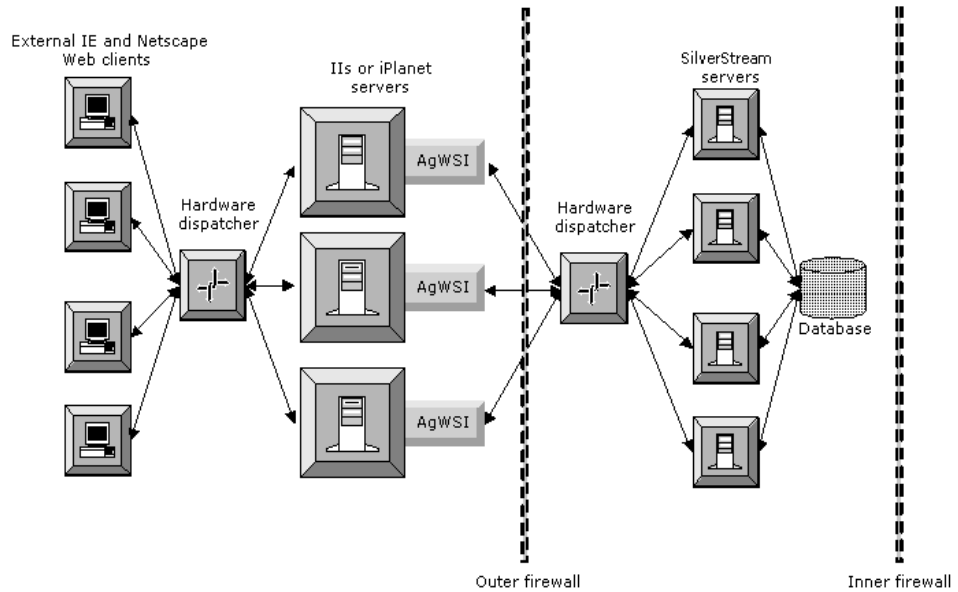
Web server with clustered SilverStream servers Here the WSI forwards requests from a single Web server to a cluster of SilverStream servers. This approach increases database access and reduces the risk of the application server becoming a single point of failure.



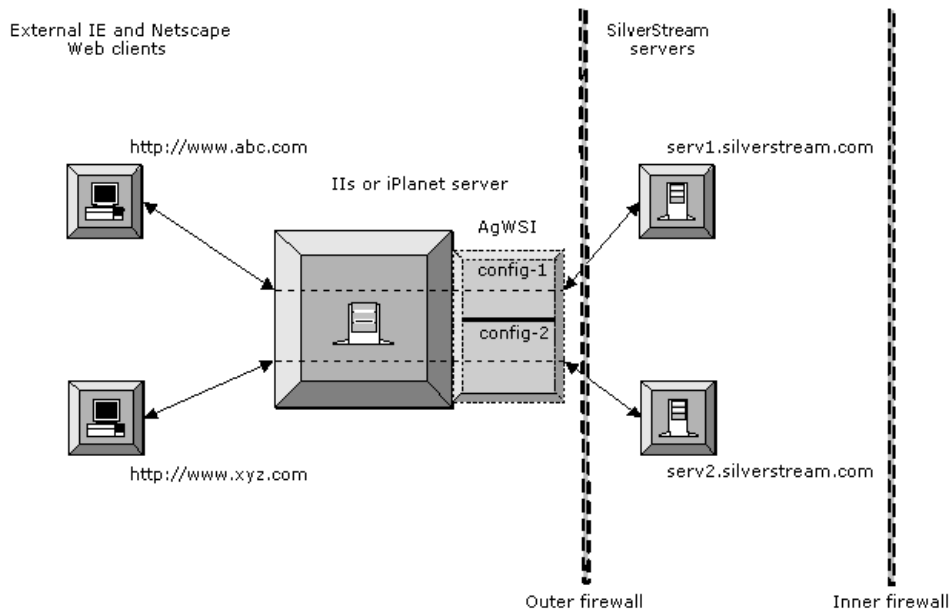
Multiple Web servers and SilverStream servers Here the WSI forwards requests from multiple Web servers to multiple SilverStream servers. This approach is more reliable for handling a large volume of requests (several Web servers have been added to the front end).



Hardware dispatcher for load balancing Here again the WSI forwards requests from multiple Web servers to multiple SilverStream servers. Now the largest volume of requests can be more reliably processed—because of multiple servers at the front and back ends and the use of dispatcher load balancing at both ends.



Web server with multiple SilverStream servers Here the WSI is set up with multiple configurations, allowing for requests to be routed to different SilverStream servers depending on the incoming request. Here requests for abc.com are redirected to serv1.silverstream.com. Requests for xyz.com are redirected to serv2.silverstream.com. For more information, see “Redirecting requests to multiple SilverStream servers” on page 188.



Using a WSI in a cluster

Because the SilverStream Dispatcher load balances server clusters by redirecting URLs instead of masking them, the SilverStream WSI modules do not work with the SilverStream Dispatcher. If the SilverStream server is running as part of a cluster, the WSI requires a third-party hardware dispatcher to securely redirect requests to appropriate SilverStream servers through one port in the firewall.

Planning your WSI installation

The WSI module uses a DLL for Windows operating systems and a shared library file for UNIX platforms. Using the WSI module is a two-part process: installing and configuring the DLL or shared library, then enabling the WSI. Before using the WSIs, you need to follow the IIS and/or iPlanet instructions in the following sections:

- Installing and configuring the WSI module
- Enabling the WSI module for IIS for Windows
- Enabling the WSI module for iPlanet for Windows and UNIX platforms

The WSI module instructions assume you have already installed the SilverStream server.

Installing and configuring the WSI module

The WSI module needs to be on the same machine as the Web server. The WSI configuration file (AgWSI.conf) must be in the same directory as the WSI module, since the WSI reads the configuration file when the Web server starts.

To run WSI modules and maintain separate log files for both IIS and iPlanet, you should create separate directories to store the WSI DLL and the shared library file with an associated AgWSI.conf file. If you are **not** running both IIS and iPlanet WSI modules on the same machine, you do not need to create separate directories.

You can run the SilverStream server remotely—you do not need to run the SilverStream server from the Web server machine.

The WSI modules for IIS and iPlanet both use the AgWSI.conf file to set configuration settings. The following procedure describes only the required settings. See “About the WSI configuration file” on page 174 for the complete list of WSI configuration settings.

➤ To install and configure the WSI module:

1. Create a directory for the SilverStream WSI module(s). For example:
 - C:\agnsapi for iPlanet on Windows
 - /opt/agnsapi for iPlanet on UNIX platforms
 - C:\inetpub\wwwroot\agisapi for IIS on Windows (create a physical directory beneath the Web server root or a virtual directory as described in “WSI.root.dir” on page 181)

- Copy the following files from the SilverStream installation directory:

Web server platform	Files to copy from the SilverStream installation directory
AIX	In \WSI\AIX: <ul style="list-style-type: none"> • libagnsapi.a (for AIX iPlanet server) • AgWSIUser • AgWSI.conf
HP-UX	In \WSI\HP: <ul style="list-style-type: none"> • libagnsapi.sl (for HP-UX iPlanet server) • AgWSIUser • AgWSI.conf
Solaris	In \WSI\SolarisSPARC: <ul style="list-style-type: none"> • libagnsapi.so (for Solaris iPlanet server) • AgWSIUser • AgWSI.conf
Windows	In \WSI\WinNT: <ul style="list-style-type: none"> • agisapif.dll (for IIS server) • agnsapi.dll (for Windows iPlanet server) • AgWSIUser.exe • AgWSI.conf

- Open the **AgWSI.conf** file and edit the required values as described in the following steps. See “About the WSI configuration file” on page 174 for the complete list of statements, including optional configuration settings.
- Specify a SilverStream server host.
The **SilverServer.host** statement specifies the name of the destination SilverStream server that services URL requests from the Web server.

5. Specify a SilverStream server port number for one or both of the following:
 - The **SilverServer.http.port** statement specifies the **nonsecure** port of the destination SilverStream server.
 - The **SilverServer.https.port** statement specifies the **secure** port of the destination SilverStream server.

If your IIS or iPlanet Web server is running both HTTP and HTTPS, specify port settings for both protocols and make sure that the SilverStream server is also running both protocols. The WSI will always forward requests to the SilverStream server using the matching protocol.

6. Specify the URLs to be forwarded to the SilverStream server.

The **SilverServer.urls** statement specifies which URLs the SilverStream server will provide. You can specify multiple SilverServer.urls statements. See “SilverServer.urls” on page 177 for URL formats.
7. (IIS only) Specify the WSI root directory you created in Step 1.

The **WSI.root.dir** statement specifies the directory the WSI DLL runs from.
8. If needed for the WSI module for IIS, specify the setting described in “WSI.auth.NTLM.remove” on page 179.
9. Save the **AgWSI.conf** file to your WSI root directory.

NOTE You can set up different sections in the configuration file, performing Steps 4 through 8 for each section—in order to support redirecting requests to multiple SilverStream servers. For more information, see “Redirecting requests to multiple SilverStream servers” on page 188.
10. Configure your Web server to use the WSI by following one of these procedures:
 - “Enabling the WSI module for IIS for Windows” on page 182
 - “Enabling the WSI module for iPlanet for Windows and UNIX platforms” on page 184

About the WSI configuration file

You configure the WSI with the AgWSI.conf file. The following file shows the default configuration for both the WSI for IIS and the WSI for iPlanet. The entries in bold are values you enter during the procedure in “Installing and configuring the WSI module” on page 172.

```
# -----  
# SilverStream WSI configuration file  
# -----  
SilverServer.host=mysssw.myco.com  
SilverServer.http.port=80  
SilverServer.https.port=443  
WSI.root.dir=/agisapi  
SilverServer.urls=/mydb
```

```
#
# Optional: Additional URL
SilverServer.urls=/SilverStream
#
# Optional Settings:
WSI.debug=0
WSI.error.url=/myerror.html
WSI.auth.NTLM.remove=false
WSI.auth.echo=false
Connection.http.max=100
Connection.https.max=100
Connection.idle.time=25
```

Configuration file settings

This section describes each configuration setting, specifies which settings are required, and provides defaults and examples.

Connection.http.max

Optional.

Connection.http.max is the maximum number of concurrent **nonsecure** HTTP connections between the WSI and the SilverStream server.

Users will be notified if the connection pooling limit is exceeded if you have created and specified a **WSI.error.url** file. The WSI reuses socket connections between itself and the SilverStream server.

Default:

```
Connection.http.max=100
```

Connection.https.max

Optional.

Connection.https.max is the maximum number of concurrent **secure** HTTPS connections between the WSI and the SilverStream server.

Users will be notified if the connection pooling limit is exceeded if you have created and specified a **WSI.error.url** file. The WSI reuses socket connections between itself and the SilverStream server.

Default:

```
Connection.https.max=100
```

Connection.idle.time

Optional.

Connection.idle.time specifies how often (in minutes) the WSI scans the connection pools for idle connections.

Default:

```
Connection.idle.time=25
```

SECTION

Optional.

SECTION names a configuration section. Each section contains the statements that specify the handling of a set of Web requests by a SilverStream server. Use multiple sections in the configuration file if you want the Web server to redirect requests to different SilverStream servers, depending on the client request.

Each section must define all required settings. If an optional setting is not defined in a section, it will be given the default value, not the value defined in any other section.

Format:

```
SECTION=label
```

The label is arbitrary.

Examples:

```
SECTION=abc_com
```

```
SECTION=xyz_com
```



For more information about the use of configuration sections, see “Redirecting requests to multiple SilverStream servers” on page 188.

SilverServer.host

Required.

SilverServer.host is the name of the destination SilverStream server that services URL requests from the Web server.

Example:

```
SilverServer.host=myssw.myco.com
```

SilverServer.http.port

Required if the SilverStream server HTTP port is not using the default port number (80).

`SilverServer.http.port` specifies the **nonsecure** port of the destination SilverStream server. Use the value zero (0) to specify that the WSI should not forward any requests coming in on the nonsecure port.

Default:

```
SilverServer.http.port=80
```

SilverServer.https.port

Required if the SilverStream server HTTPS port is not using the default port number (443).

`SilverServer.https.port` specifies the **secure** port of the destination SilverStream server. Use the value zero (0) to specify that the WSI should not forward any requests coming in on the secure port.

Default:

```
SilverServer.https.port=443
```

SilverServer.urls

Required.

`SilverServer.urls` specifies which URLs will be forwarded to the SilverStream server. You must specify a new setting for each URL root you want forwarded to the SilverStream server.

There are two formats: Simple URL forwarding and URL forwarding with translation (masking).

Simple URL forwarding

Format:

```
SilverServer.urls=<Root_URL_to_Forward>
```

Examples:

```
SilverServer.urls=/SilverStream
SilverServer.urls=/myDb
```

In the preceding example, all URLs that begin with either /SilverStream or /myDb will be forwarded to the SilverStream server. This includes:

```
http://myWebServer/SilverStream/Sessions
http://myWebServer/SilverStream/Pages
http://myWebServer/myDb/SilverStream/Pages/MyPage.html
```

To forward **all** URLs to the SilverStream server, specify the following:

```
SilverServer.urls=/
```

URL forwarding with translation (masking)

Format:

```
SilverServer.urls=<URL_root_at_Web_server>=<translated_URL_root>
```

Example:

```
SilverServer.urls=/Pages=/myDb/SilverStream/Pages
```

The first URL will be forwarded to the SilverStream server after the second URL is substituted for it. In this example, all URLs from the Web server that begin with /Pages will be forwarded to the SilverStream server with /myDb/SilverStream/Pages substituted for /Pages. The URL sent to the Web server as:

```
http://myWebServer/Pages/MyPage.html
```

is forwarded to the SilverStream server as:

```
http://mySilverServer/myDb/SilverStream/Pages/MyPage.html
```

WSI.auth.echo

Optional.

When a request sent to the Web server contains an HTTP authorization header, the WSI will send an HTTP header (called x-agwsi-AuthORIZATION) to the SilverStream server that echoes the value of the header when WSI.auth.echo is set to true.

This setting allows the SilverStream application to retrieve the user login information when the user login has been masked with the **WSI.auth.user** command. For example, when a third-party product (such as enCommerce getAccess) is performing authentication and authorization services, the WSI.auth.echo setting allows the SilverStream application to retrieve the name of the user who logged into the application and initiated the request.

The HTTP header will appear in the following (name/value) format:

```
x-agwsi-Authorization: Basic Base64EncodedUserName/Password
```

Default:

```
WSI.auth.echo=false
```

NOTE The SilverStream server uses the AgiHttpRequest API to retrieve the authorization header.

WSI.auth.NTLM.remove

Optional (used with IIS only).

Set WSI.auth.NTLM.remove to true if NT authentication is enabled for your IIS directories. Setting the value to true removes the NTLM authentication headers so users' requests can be successfully forwarded to the SilverStream server. For more information, see "Using IIS NTLM authentication with the WSI module" on page 191.

Default:

```
WSI.auth.NTLM.remove=false
```

WSI.auth.user

Optional.

When WSI.auth.user is specified, the WSI module intercepts the authentication headers that will be forwarded to the SilverStream server and replaces them with the credentials of a single known SilverStream user, then it adds HTTP authentication headers to every request that it forwards to the SilverStream server. Any existing authentication headers on incoming requests to WSI will be replaced by the authentication setting.

To protect the security of the authentication settings, the user name and password are not stored in clear text in the AgWSI.conf file. You must run the AgWSIUser utility to generate the **WSI.auth.user** statement that is needed in AgWSI.conf to represent a given user and password. The AgWSIUser utility encrypts the user name and password in a way that the WSI can read. For more information on running the utility, see "The AgWSIUser utility" on page 191.

The authentication setting can be used:

- With your enCommerce getAccess integration
- To remove NTLM authentication headers added by IIS
- To distinguish requests that have been forwarded by the WSI to the SilverStream server from requests that have been sent directly from a browser

WSI.debug

Optional.

WSI.debug specifies the WSI logging level. The WSI logs to the AgWSI.log file, which is stored in your WSI module directory. Choose from these levels:

Level	Information logged
0	None
1	<ul style="list-style-type: none">• Each request method, URL, and whether it was processed successfully• Any errors in the connection between the WSI module and the SilverStream server• Connection pool cleanup messages
2	Level 1 information, plus: <ul style="list-style-type: none">• (iPlanet only) Full HTTP response and reply headers and content lengths
3	Level 2 information, plus: <ul style="list-style-type: none">• URL mapping results

To log the URLs arriving at the destination server, use the SMC to configure your SilverStream server debug value to 1 or 2. For more information, see “Low-level debugging” on page 408.

Default:

```
WSI.debug=0
```

WSI.error.url

Optional.

WSI.error.url specifies a customized error page that users will see if a WSI connection error occurs. Users will see a generic browser notification when the WSI module cannot connect to the SilverStream server unless you create and specify a WSI error URL.

It is a good idea to create an HTML file that tells users about the problem and advises them to retry the URL connection later.

Specify the WSI error file name and its location on your Web server. You need to store your error page file in your Web server’s directory structure, since the WSI redirects the browser to this page on your Web server.

Default:

```
WSI.error.url=\myerror.html
```

WSI.host

Optional.

WSI.host specifies the HTTP host header to filter when matching URLs to be forwarded to a SilverStream server. If this statement is not specified, the request host header is ignored and only URL matching is used as a filter.

This setting is used for multi-homed Web server configurations, where the Web server is hosting multiple separate host names, and the WSI needs to forward URLs to different SilverStream servers based on the request host name. The WSI.host setting can specify either the hostname or the hostname and port number. If the port number is not specified, the WSI will accept any port number on the request's host header provided the host names match.

Examples:

```
WSI.host=www.abc.com  
WSI.host=www.abc.com:8080
```



For a sample use of this statement, see “Redirecting requests to multiple SilverStream servers” on page 188.

WSI.root.dir

Required for IIS only.

WSI.root.dir is the WSI virtual directory name where the WSI module runs from. Since the WSI for IIS is both a filter and an extension, you need to specify the URL for the WSI in the IIS Web root (/wwwroot) directory structure.

You need to install the WSI module for IIS in a directory that is visible from the IIS Web root directory. If you install the WSI in a physical directory beneath the Web root, the WSI is automatically visible from within IIS. However, if you install the WSI module in a directory that is outside the IIS root directory, you must create a virtual directory using the MMC so the WSI appears in the IIS directory.

You must set WSI.root.dir relative to the Web server root directory.

Examples:

- If installing the WSI into C:\inetpub\wwwroot\agisapi, a physical directory beneath the Web root, specify WSI.root.dir=/agisapi.

- If installing the WSI into a virtual directory such as C:\WSI, use the MMC to create a virtual directory such as /agisapi that maps to C:\WSI and then specify WSI.root.dir=/agisapi.


Default:

```
WSI.root.dir=/agisapi
```

Enabling the WSI module for IIS for Windows

This section describes how to enable the WSI for Microsoft IIS. This procedure assumes that you have done the setup work described in “Installing and configuring the WSI module” on page 172.

➤ To enable the WSI module for IIS:

1. Make sure your edited **AgWSI.conf** file is in your WSI root directory (such as C:\AgISAPI) as described in Step 2 under “Installing and configuring the WSI module” on page 172.
2. Start the **World Wide Web Publishing Service** and open the **Internet Service Manager**.
 For more information on Microsoft Web server configuration, see your Microsoft documentation.
3. From the **Microsoft Management Console (MMC)**, right-click **Default Web Site**.
4. Select **New** and then **Virtual Directory**. The New Virtual Directory Wizard displays.
If your WSI module does **not** use a virtual directory, do the following:
 - In the MMC, right-click your WSI directory and select **Properties**.
 - On the **Directory** tab, disable **Read** and **Write** access permissions and make sure **Execute** permissions are enabled.
 - Go to Step 9.
5. From the **New Virtual Directory Wizard**, enter a virtual directory name (such as AgISAPI) and click **Next**.

This step ties the virtual directory to the physical directory and is optional if you install the WSI directly under the IIS physical root directory. The virtual path is a subdirectory of the IIS Web root directory. The WSI can be installed in any physical directory, provided it is on the same machine as the Web server.

IIS will use this alias to access the directory the WSI DLL runs from. This name should match the name used in the **AgWSI.conf** file to define the **WSI.root.dir** setting as described in “About the WSI configuration file” on page 174.

6. Enter the physical path of the WSI directory (such as C:\AgISAPI) and click **Next**.
The WSI directory is where you installed the SilverStream WSI module.
7. Disable all permission access check boxes except **Allow Execute Access (includes Script Access)**, which should be selected.
8. Click **Finish**.
9. Verify that the directory you just created appears in the list of Web server directories beneath the **Default Web Site** in the left pane of the MMC.
10. Right-click **Default Web Site** and select **Properties**.
11. Select the **ISAPI Filters** tab and click **Add**.
12. In the **Filter Properties** dialog, enter the **WSI Filter Name**.
13. In the **Executable** field, specify the absolute path to the **agisapif.dll**. For example, C:\AgISAPI\agisapif.dll.
14. Click **OK** to close the **Filter Properties** dialog.
15. Click **Apply** in the **Default Web Site Properties** dialog.
The WSI module for IIS should appear in the **Filter Name** list. A green arrow to the left of the **WSI Filter Name** indicates whether (or not) the filter is enabled.
16. Click **OK** to close the **Default Web Site's Properties** dialog.
17. Close the **Internet Service Manager**.
18. Stop and then restart the **World Wide Web Publishing Service** using the Windows control panel.
19. To verify that the WSI module is working, start the browser and connect to an URL that you specified in the **AgWSI.conf** file.

When the WSI module starts successfully, it starts populating the AgWSI.log file.

Here is a sample log file:

```
# -----
# SilverStream WSI Configuration
# -----
Started at           : Tue Oct 2 14:10:19 2000
Root Directory      : /sssw/
Host                : richg.silverstream.com
Ports               : 8080 445
Maximum HTTP Connection : 8
Maximum HTTPS Connection : 8
Connection Idle Time : 25 minutes.
Redirect URL on Error : /wsi/myerror.html
Authentication NTLM   : true
-----
```

If the WSI does not start successfully, an error message displays.

Enabling the WSI module for iPlanet for Windows and UNIX platforms

This section describes how to enable the WSI module for Netscape iPlanet.

NOTE This procedure assumes that you have done the setup work described in “Installing and configuring the WSI module” on page 172.

In order to enable the WSI module on a Netscape server, you need to edit its configuration files.

- For iPlanet 6.x, you add initialization settings to **magnus.conf** and service-related entries to **obj.conf**.
- For iPlanet releases lower than 6.x, you add all the entries to **obj.conf**.

When a WSI module is running with a Netscape Web server, the WSI configuration file (AgWSI.conf) must reside in the same physical directory as the WSI module that you specify in Step 2 of the following procedure.

➤ To enable the WSI module for iPlanet:

1. For iPlanet 6.x, open your Netscape Web server **magnus.conf** file.
For iPlanet releases lower than 6.x, open your Netscape Web server **obj.conf** file.
2. Go to the **Init** section and add the first of two lines of text to initialize the module name and location.

This **first line** sets the initialization function that ensures that the WSI module is loaded and its functionality is made available to the server. You need to enter the WSI module name and path for your system. What you enter as the name of each WSI module depends on which platform the WSI module will run on:

- For iPlanet on AIX, enter **libagnsapi.a**
- For iPlanet on HP-UX, enter **libagnsapi.sl**
- For iPlanet on Solaris, enter **libagnsapi.so**
- For iPlanet on Windows, enter **agnsapi.dll**

When you specify the **WSI module name**, enter (all on one line) the path to the SilverStream WSI modules for your system:

```
Init fn="load-modules" shlib="WSI_module_name"  
funcs="AgNSAPIInit, AgNSAPINameTrans, AgNSAPIService"
```

AIX example:

```
shlib="/opt/AgNSAPI/libagnsapi.a"
```

HP-UX example:

```
shlib="/opt/AgNSAPI/libagnsapi.sl"
```


Solaris example:

```
shlib="/opt/AgNSAPI/libagnsapi.so"
```

Windows example:

```
shlib="C:\AgNSAPI\agnsapi.dll"
```

3. Now (still in the **Init** section) add the **second line** of text. This line initializes the WSI module when the iPlanet server starts.

For UNIX platforms, enter:

```
Init fn="AgNSAPIInit" LateInit="yes" agroot="/opt/SilverStreamDir/bin"
```

For Windows, enter:

```
Init fn="AgNSAPIInit" LateInit="yes"
```

4. For all releases of iPlanet, open **obj.conf**, go to the **NameTrans** section, and do **one** of the following:

- If you are **not** using enCommerce, at the beginning of the **NameTrans** section add the following:

```
NameTrans fn="AgNSAPINameTrans"
```

This name translation command submits URLs to the SilverStream server for servicing and determines whether or not the requested URL gets translated.

- If you are using **enCommerce**, add the following to the beginning of the **NameTrans** section instead of the text shown above:

```
NameTrans fn="AgNSAPINameTrans" agroot=AnyPath
```

You only need to set the **agroot** portion of the **NameTrans** parameter if you are using the enCommerce getAccess security software.

The getAccess authentication service requires that a URL be mapped to a physical path in order to secure the URL request. The WSI module will use the path you specify to provide the physical path that getAccess requires. Because the WSI does not check for the existence or the validity of the path, you can specify any path. The WSI will set a physical path for URLs that it will service relative to the root path that you specify.

For example, if you enter

```
agroot=/opt/silverstream
```

in the preceding **NameTrans** function, an incoming URL such as

```
http://server/MyDB/SilverStream/Pages/xyz.html
```

would be mapped by the WSI to

```
/opt/SilverStream/MyDB/SilverStream/Pages/xyz.htm
```



For more information, see “enCommerce getAccess integration” on page 192.

5. At the beginning of the **Service** section, add the following line:


```
Service fn="AgNSAPIService"
```

This service command forwards each request to the SilverStream server and returns (echoes) a response to each incoming request.

6. Stop and restart the Netscape Web server to verify that the WSI module was loaded correctly.

WSI for iPlanet sample configuration files

The following is an example of completed `magnus.conf` and `obj.conf` files for iPlanet 6.x that specify a WSI module running on Windows. The WSI entries you must add are shown in **bold**. (For releases before 6.x, all the items in **bold** would be in `obj.conf`.)

 For more information about Netscape Web Server configuration, see your Netscape documentation.

The `magnus.conf` file

```
#ServerRoot D:/iPlanet/Servers/https-richg
ServerID https-richg
ServerName richg
ExtraPath D:/iPlanet/Servers/bin/https/bin;${NSES_JRE_RUNTIME_LIBPATH}
ErrorLog D:/iPlanet/Servers/https-richg/logs/errors
MtaHost name-of-mail-server
DNS off
Security off
ClientLanguage en
AdminLanguage en
DefaultLanguage en
RqThrottle 128

Init fn=flex-init access="$accesslog" format.access="%Ses->client.ip% - %Req->vars.auth-
user% [%SYSDATE%] \">%Req->reqpb.clf-request%\ " %Req->srvhdrs.clf-status% %Req-
>srvhdrs.content-length%"
Init fn=load-types mime-types=mime.types
Init fn="load-modules" shlib="D:/iPlanet/Servers/bin/https/bin/NSServletPlugin.dll"
funcs="NSServletEarlyInit,NSServletLateInit,NSServletNameTrans,NSServletService"
shlib_flags="(global|now)"
Init fn="load-modules" shlib="d:/WSI/WinNT/agnsapi.dll"
funcs="AgNSAPIInit,AgNSAPINameTrans,AgNSAPIService"
Init fn="NSServletEarlyInit" EarlyInit=yes
Init fn="NSServletLateInit" LateInit=yes
Init fn="AgNSAPIInit" LateInit="yes"
```

The `obj.conf` file:

```
# Use only forward slashes in pathnames--backslashes can cause
# problems. See the documentation for more information.

<Object name=default>
NameTrans fn="AgNSAPINameTrans"
NameTrans fn="NSServletNameTrans" name="servlet"
NameTrans fn="pfx2dir" from="/servlet" dir="D:/iPlanet/Servers/docs/servlet"
name="ServletByExt"
NameTrans fn=pfx2dir from=/mc-icons dir="D:/iPlanet/Servers/ns-icons" name="es-internal"
NameTrans fn="pfx2dir" from="/manual" dir="D:/iPlanet/Servers/manual/https" name="es-
internal"
NameTrans fn=document-root root="$docroot"
PathCheck fn=nt-uri-clean
PathCheck fn="check-acl" acl="default"
PathCheck fn=find-pathinfo
PathCheck fn=find-index index-names="index.html,home.html"
ObjectType fn=type-by-extension
ObjectType fn=force-type type=text/plain
Service fn="AgNSAPIService"
Service type="magnus-internal/jsp" fn="NSServletService"
Service method=(GET|HEAD) type=magnus-internal/imagemap fn=imagemap
Service method=(GET|HEAD) type=magnus-internal/directory fn=index-common
Service method=(GET|HEAD|POST) type=*-magnus-internal/* fn=send-file
AddLog fn=flex-log name="access"
</Object>

<Object name=cgi>
ObjectType fn=force-type type=magnus-internal/cgi
Service fn=send-cgi
</Object>

<Object name="servlet">
ObjectType fn=force-type type=text/html
Service fn="NSServletService"
</Object>

<Object name="jsp092">
ObjectType fn="type-by-extension"
ObjectType fn="change-type" type="magnus-internal/jsp092" if-type="magnus-internal/jsp"
Service fn="NSServletService" type="magnus-internal/jsp092"
</Object>

<Object name="ServletByExt">
ObjectType fn=force-type type=magnus-internal/servlet
Service type="magnus-internal/servlet" fn="NSServletService"
</Object>
```

```
<Object name="es-internal">
PathCheck fn="check-acl" acl="es-internal"
</Object>
```

Redirecting requests to multiple SilverStream servers

You can define multiple sections in an AgWSI.conf file in order to direct client requests to different SilverStream servers. Each configuration section is labeled with a **SECTION** statement and contains the statements that specify which requests are redirected to which SilverStream server. Each section must contain all statements listed as required in “Configuration file settings” on page 175.

In addition, your Web server might be **multihoming**—hosting more than one host name. Using the **WSI.host** statement, you can configure the WSI to forward requests based on the request’s host name (and optionally port). If a section does not have a WSI.host statement, the request host header is ignored and only URL matching is used as a filter. For more information, see “WSI.host” on page 181.

For each incoming request to the Web server, the WSI searches all the configuration sections until it finds a match based on the host name and port in the request header (if specified by WSI.host) and the URL of the request. The request is then forwarded to the SilverStream server specified in that section.

A sample configuration file redirecting to different SilverStream servers

The following shows a configuration file with three sections, specifying two different Web server host names and redirecting requests to two different SilverStream servers. The sample also shows how to configure the WSI so that specified URLs can be accessed only through a secure (HTTPS) connection.

```
SECTION=WWW_ABC_COM

# Redirect all URLs for www.abc.com to serv1.silverstream.com
# HTTP requests will be forwarded to port 80 on the SilverStream server
# HTTPS requests will be forwarded to port 443 on the SilverStream server

WSI.host=www.abc.com
WSI.root.dir=/AgISAPI

SilverServer.host=serv1.silverstream.com
SilverServer.http.port=80
SilverServer.https.port=443
SilverServer.urls=/
```

```
Connection.http.max=100
Connection.https.max=100
Connection.idle.time=25
```

SECTION=WWW_XYZ_COM_SECURE

```
# Redirect URLs starting with /db1/approot/secure
# for www.xyz.com to serv2.silverstream.com
# only from HTTPS (secure port) (SilverServer.http.port is set to 0)
```

```
WSI.host=www.xyz.com
WSI.root.dir=/AgISAPI
```

```
SilverServer.host=serv2.silverstream.com
SilverServer.http.port=0
SilverServer.https.port=443
SilverServer.urls=/db1/approot/secure
```

```
Connection.http.max=100
Connection.https.max=100
Connection.idle.time=25
```

SECTION=WWW_XYZ_COM_HTTP

```
# Redirect all other URLs for www.xyz.com
# to serv2.silverstream.com on any port
```

```
WSI.host=www.xyz.com
WSI.root.dir=/AgISAPI
```

```
SilverServer.host=serv2.silverstream.com
SilverServer.http.port=80
SilverServer.https.port=443
SilverServer.urls=/
```

```
Connection.http.max=100
Connection.https.max=100
Connection.idle.time=25
```

About connection pooling

The WSI module uses connection pooling to improve response time. Instead of creating and maintaining a connection to the SilverStream server for each client connected to the Web server, the WSI reuses its connections to the SilverStream server for multiple client connections. The WSI will open new connections to the SilverStream server as needed for concurrent request processing. The number of concurrent connections is limited by the **Connection.http.max** and the **Connection.https.max** settings in the AgWSI.conf file.

WSI maintains separate connection pools for HTTP and HTTPS protocols. To ensure that these connection pools do not use up too many system resources, the WSI periodically scans the pools and closes any connection that has not been used between scan cycles.

Connections have the following states:

- **Connected** (connection is busy and active)
- **Inactive** (connected but time limit has not expired)
- **Idle** (not active and idle limit has expired)
- **Not connected**

The WSI periodically runs a background thread to check the connection pools for inactive and idle connections to see which connections are connected but not busy. The WSI thread marks all inactive threads as idle and closes any connections that are already marked as idle. When a connection is marked as idle, that connection is closed based on the connection idle time limit you set (or the default of 25 minutes). The interval starts when the WSI module is loaded, not when the connection was created.

A connection needs to remain inactive for two scan cycles before it is disconnected. For example, if the specified connection idle time limit is set to 15 minutes, an inactive connection will be disconnected sometime between 15 and 30 minutes after becoming inactive (because the WSI may not start pooling until the middle of an existing interval). If an idle connection is used during the time interval between the scan cycles, the connection goes back into the connection pool marked as inactive.

Authentication and security integration considerations

Because of variations in server platforms, architecture, and third-party security providers, you may need to be aware of several security considerations when using the WSI:

- Using IIS NTLM authentication with the WSI module
- The AgWSIUser utility

- HTTPS client certificate authentication issues
- enCommerce getAccess integration

Using IIS NTLM authentication with the WSI module

If you secure your Web site using the Microsoft Windows NT LAN Manager (NTLM) authentication, the WSI default header settings will not work. After authenticating incoming requests, IIS adds an NTLM HTTP authentication header to each request. Because the NTLM HTTP authentication header is not supported by the SilverStream server, incoming requests will be rejected unless you configure the WSI for IIS module in one of the following ways:

- Replace all authentication headers with the ones set by the AgWSIUser utility. For more information, see the “The AgWSIUser utility” on page 191.
- Remove the NTLM HTTP authentication headers from all requests sent to the SilverStream server. Specifying **WSI.auth.NTLM.remove** in the AgWSI.conf file to true allows users’ requests to be successfully forwarded to the SilverStream server once NTLM headers are removed. For more information, see “WSI.auth.NTLM.remove” on page 179.

The AgWSIUser utility

AgWSIUser is a command-line utility that generates the **WSI.auth.user** statement that is needed in AgWSI.conf to define a WSI user and password. The AgWSIUser utility encrypts the user name and password in a way that the WSI can read.



For more information, see “WSI.auth.user” on page 179.

➤ To use the AgWSIUser utility:

1. From the command line, change to your WSI root directory.
2. Enter the following command:

```
AgWSIUser username password
```

Use your SilverStream user name and password. If your password is blank, just enter the user name. You can enter any valid SilverStream user. For example, to use NT credentials run the following:

```
AgWSIUser myNTDomain\myNTUser myNTPassword
```

The AgWSIUser utility prints the corresponding WSI.auth.user statement in the command window.

3. Paste the generated statement into the appropriate section of the AgWSI.conf file. (If you are not using sections, paste it anywhere in the file.)

At startup, the WSI module will decrypt the user name and password and generate an HTTP authentication header that it will add to every request it forwards to the SilverStream server.

HTTPS client certificate authentication issues

The WSI module cannot forward the client certificate from the browser to the SilverStream server. Because the WSI opens a new HTTPS connection to the SilverStream server, the WSI would need to access the private key of the client certificate to be able to use the client certificate for this connection. The private key is not available, because it is securely stored by the browser on the client machine.

Because the WSI cannot send client certificates to the SilverStream server, the WSI will not work with a SilverStream server that is configured to require client certificates for authentication.

enCommerce getAccess integration

The SilverStream WSI module and the enCommerce getAccess security software can be used together on the IIS Web server. When integrated, getAccess will provide authentication and authorization services at the Web server, and the WSI module will provide access to the SilverStream server.

NOTE If your WSI module for iPlanet is integrated with getAccess security, you need to secure your Web site's URLs by entering a path statement in the NameTrans function of the obj.conf file. For more information, see "Enabling the WSI module for iPlanet for Windows and UNIX platforms" on page 184.

Because authentication and authorization take place at the Web server with the getAccess service, the SilverStream server does not need to know about and check the authorization of every user. Instead, it only needs to authenticate and authorize a single user (the user that the WSI module is configured to use).

As described in "The AgWSIUser utility" on page 191, the WSI module intercepts the authentication headers that will be forwarded to the SilverStream server, and replaces the enCommerce credentials with credentials of a single known SilverStream user.

About SilverStream and getAccess security

This section describes the SilverStream and getAccess security models and discusses how to integrate them.

The two models

The SilverStream server The SilverStream eXtend Application Server provides built-in security features including authentication, access control, data privacy, data integrity, and single signon to multiple SilverStream servers. In addition to providing its own internal users and groups, the SilverStream server is integrated with several types of security providers that include NT, LDAP, NIS+, and client certificates. You can control access to resources using simple access control lists (ACLs) or using advanced security expressions. You can specify the following types of access:

This type of access	Specifies who can
Read	View an object including its source code
Write	Modify the source code, or delete the object
Execute	Run application objects (such as pages, forms, views, and so on)
Set Permissions	Alter access control information on an object
Select	Access row-level data in individual tables

You can define SilverStream security using the SilverStream Management Console (SMC) to set security expressions or using the security management API for programmatic access. You can specify authentication for individual resources or server-wide. Security expressions that you set for resources at the directory level can be propagated to the children of the directory.

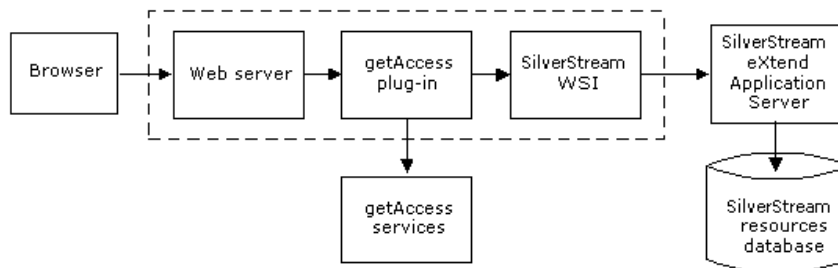
getAccess The getAccess services include authentication and access control. Browser requests to an IIS Web server are processed by a getAccess plug-in. The plug-in detects unauthenticated browser sessions and routes these session requests to a login page. The client can log in as an internal getAccess user or against one of the getAccess plug-in authentication modules. The getAccess plug-in checks the runtime access of the authenticated user to the URL-specified resource and associated components.

As with SilverStream security, with `getAccess` you can define users and groups (or roles) that control access to SilverStream resources. SilverStream and `getAccess` users can be authenticated against external security providers. For `getAccess` users, you can define password policies governing minimum password length and password expiration. You must individually specify and protect resources on the host Web server.

The SilverStream WSI module for IIS receives browser requests from a client through the Web server and forwards them to the application server.

NOTE You can stack WSI modules to process different groups of requests so that once one WSI module is done with a request the next WSI would receive it.

The figure below shows the architecture and process used to integrate `getAccess` with the SilverStream server:



How it all works

getAccess plug-in The `getAccess` plug-in initially receives and checks the user request, since it is loaded at a higher priority than the SilverStream WSI. The `getAccess` plug-in checks with the `getAccess` services to see if the resource URL has been secured. If it has, the `getAccess` plug-in checks for an authentication header indicating whether the client's `getAccess` user identity has been established. If it has not, the request is forwarded to a `getAccess` login page. Once the user has logged in, the `getAccess` plug-in passes user credentials to the `getAccess` services to determine if the user is authorized to access the requested resource. If the user is not authorized, the `getAccess` plug-in returns an access-denied error.

If the `getAccess` plug-in determines that the user is allowed access to a SilverStream server resource, the request is next processed by the SilverStream WSI module.

SilverStream WSI The SilverStream WSI module forwards the request to the application server host specified in the AgWSI.conf file. Because the SilverStream server cannot verify a getAccess user, the WSI checks the request for an authentication header and then substitutes the credentials set as defaults in the AgWSI.conf file. For more information, see “The AgWSIUser utility” on page 191.

Once the getAccess plug-in determines that the user is allowed access, the SilverStream server returns the requested URL to the plug-in and then to the browser user.

NOTE In the preceding scenario, all security access is controlled by getAccess. The single known set of SilverStream credentials (defined by the AgWSIUser utility) is only used to connect to the application server. You still need to set SilverStream security to secure application resources against users connecting directly to the application server (rather than through getAccess).

Configuring the integration

This section assumes you have installed enCommerce getAccess and the SilverStream Application Server into your existing IIS Web server framework.

➤ To configure enCommerce getAccess for the WSI for IIS:

1. Select the **Default Web Site** from the list of Web server directories in the left pane of the MMC (Microsoft Management Console).
2. Right-click **Default Web Site** and select **Properties**.
3. Select the **ISAPI** tab and verify that the getAccess plug-in takes precedence over the SilverStream WSI. The getAccess plug-in must have a higher priority than (and must be loaded before) the WSI module.
4. Add the WSI.auth.user setting in the AgWSI.conf file as described in “The AgWSIUser utility” on page 191.

The AgWSIUser utility generates the appropriate WSI.auth.user statement with the user name and password encrypted for pasting into the AgWSI.conf file. At startup, the WSI module will decrypt the user name and password and generate an HTTP authentication header that it will add to every request it forwards to the SilverStream server.

5. Start the SilverStream server and run the SMC.

6. Assign the authenticated user (specified in Step 4) **Execute** access to all SilverStream resources accessed through `getAccess`. You need to set SilverStream security expressions on all resources to be protected by `getAccess`.

This step is required even though `getAccess` (using a `getAccess` user login) performs the access control through the Web server. The SilverStream security expressions prevent unauthorized access to resources when a client connects directly to the application server. Because `getAccess` only provides runtime access, SilverStream expressions should be used to control administrative and design-time access.

A typical scenario would be to assign Read (design-time) access and Write access to the Developers group, Set Permissions access to the Administrators group, and Execute access to both the Developers group and the authenticated WSI user defined in `AgWSI.conf` file.

If access to the application server is controlled by a firewall, you may want to assign unrestricted Execute access to all users. Once `getAccess` and the SilverStream WSI module are configured, you should use the `getAccess` administration tool to protect SilverStream resources.



For more information on configuring `getAccess` users and roles and plug-in authentication providers, see your `enCommerce getAccess` documentation.

➤ To protect SilverStream resources:

1. From the `enCommerce getAccess` Resource Administration screen, click **Create**.
2. Provide values for **Resource ID**, **Resource Name**, and **Description**.
3. Specify the Web server that is running `getAccess`.
4. Specify the SilverStream database name and directory for the **Relative URL** field. For example:

```
/myDatabase/Pages/
```

5. Specify the path to the SilverStream WSI DLL, appended with the URL path to the resource in the **Items to Protect** field. For example:

```
c:\SilverStream\bin\myDatabase\Papers\paperEdit.html
```

You may optionally specify any additional referenced resources such as image files.

6. Fill in the remaining fields.
See your `getAccess` documentation for appropriate values.
7. Save the newly created `getAccess` resource and assign roles to restrict access to specific users. In order for the new resource to be activated, the server configuration must be updated.
8. Click the **Servers** button to open the Server Administration tool.
9. Select the appropriate server and click **Update Config**.

10. Click the **Update** button.
11. Click the **Reload Config** button and then click **Reload** to reload the configuration. The resource is now ready to be tested.

When getAccess users request a resource through the Web server, they are authenticated and authorized by the getAccess plug-in and associated services. If users are authorized by getAccess, the SilverStream WSI module substitutes the WSI authentication headers (defined by the AgWSIUser utility) and then requests the resource from the application server.

9

Setting Up Security

This chapter describes how to configure a SilverStream server for security.

It contains sections on:

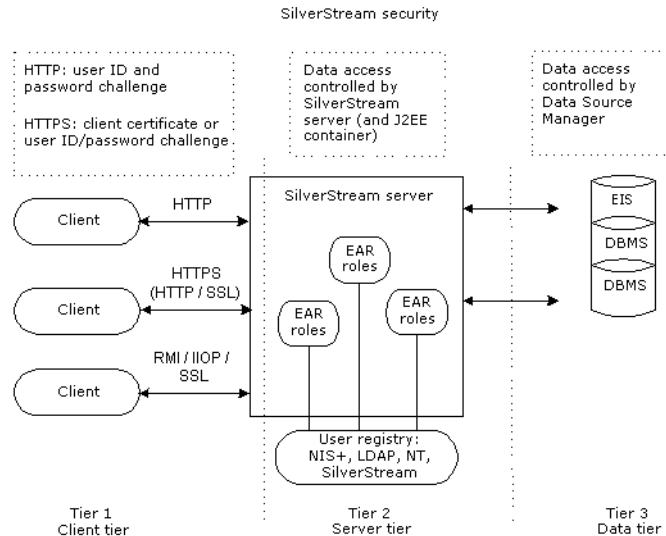
- Security configuration
- About authentication
- Establishing a secure connection to the server
- Accessing security provider systems
- Using security provider login formats
- Using certificates
- Enabling authentication
- Using Cryptographic Hardware Integration
- Managing trusted clients



For information about setting up SilverStream users and groups, see Chapter 6, “Setting Up Users and Groups”.

Security configuration

In the SilverStream three-tiered architecture, security is set up at the server tier, between the client tier and the data tier.



The SilverStream server acts as a single data source user with multiple connections into a database (where applications are deployed) or a connection pool (from which data is retrieved by the application). Acting in this manner, the server adds extra user and object security for the data source. In effect, the SilverStream server is treated as just a user with access privileges. Native data source activity and security measures are not compromised.

SilverStream supports both regular HTTP and HTTP using the Secure Sockets Layer (SSL) 3.0 protocol (HTTPS). HTTPS provides data encryption between the SilverStream client and server to ensure privacy and data integrity.

EJBs running on the SilverStream server use the IIOP over SSL protocol to ensure privacy and data integrity. The IIOP over SSL support is provided by the jBroker ORB, which has the ability to create secure objects with the desired quality of protection using cipher suites specified per bean in the bean's deployment plan. SilverStream supports RSA authentication only for EJBs.

 For more information on the jBroker ORB's IIOP over SSL support, see the jBroker ORB documentation in the online help system.

About SSL The SSL (Secure Sockets Layer) handshake protocol was developed by Netscape to provide security and privacy over the Internet. HTTPS provides data encryption between the SilverStream client and server to ensure privacy and data integrity. The SSL protocol also authenticates the server before data is exchanged by the higher-level application. SSL is application-independent, allowing protocols like HTTP and FTP to be layered on top of it transparently. The SSL protocol is able to negotiate encryption keys as well as authenticate the server before data is exchanged by the higher-level application. The SSL protocol maintains the security and integrity of the transmission channel by using encryption, authentication, and message authentication codes.

You may choose to require SSL to access your application. Because SSL affects performance, you may decide to use SSL only for specific data-sensitive portions of your site. Or you could consider using Cryptographic Hardware Integration (CHI), which enables significant application server SSL encryption/decryption performance enhancements. See “Using Cryptographic Hardware Integration” on page 268 for information on installing and using CHI on your server.

About HTTPS With HTTPS, you get a communications channel that provides privacy, user authentication, and message integrity. SSL is implemented in SilverStream as follows:

- The SilverStream server must have an authentication certificate (also called public-key certificate, digital ID, or digital certificate) to make an SSL connection. The certificate is a digital “ID card” that cannot be forged. This certificate describes the server and includes a chain of trust.
- Both client and server encrypt what they send using information from both their own certificate and the certificate at the other end (if the client has a certificate). This means the sender can be sure that only the intended recipient can decrypt the data, and the recipient can be sure that the data came from the place it claims to have come from and that no tampering has occurred.

Mapping J2EE roles to existing users You can map roles defined in a J2EE archive’s deployment plan to users and groups that the SilverStream server recognizes as SilverStream users or users and groups from an external security system. See “Accessing security provider systems” on page 208 for information on how this works.

Types of encryption used for authentication

Authentication begins and ends with the client session. Both the RSA (Rivest-Shamir-Adleman) and DSA (Diffie-Hellman) encryption algorithms are based on public and private keys. The SSL protocol requires the server to have an X.509 certificate containing its identity, its public key, and the identity and signature of the Certificate Authority (CA) that issued the certificate. The client authenticates the server based on the certificate it receives. The client then encrypts the public key and sends it back to the server to be used for encrypting further data transmissions. The encryption algorithms usually used are RC4, DES, and 3DES.

The client session operates in one of three modes:

Mode	Description
RSA (Rivest-Shamir-Adleman)	Encryption provides secure communications between SilverStream Java clients, non-SilverStream Java clients, HTML clients, and the SilverStream server.
DSA (Diffie-Hellman)	Encryption provides a secure channel for the SilverStream server to communicate with SilverStream Java and non-SilverStream Java clients.
Base64 encoding	Encryption method used with HTTP protocol (when SSL is not implemented) to send user name and password information from the client to the server. This encryption method can be decrypted easily. To ensure a secure exchange of user name and password, use an SSL connection. The SilverMaster stores SilverStream user names with encrypted passwords, or you can use any of the external security providers supported by SilverStream. For external security systems, SilverStream verifies the password information with the external security system provider.

Security functions

SilverStream security performs four major functions:

Function	Description
Authentication	This is done through a challenge, such as requesting a user ID/password pair when using HTTP, or through an authentication certificate when using HTTPS.
Access control	Once the system verifies a user's identity, it checks whether the user is allowed to perform the requested operation on the requested object.
Data integrity	The system ensures that the data received over the network is the same data that was sent.
Data privacy	The system prevents unauthorized users from seeing data during transmission.

The SilverStream security system handles all data integrity and data privacy functions with virtually no administrator involvement.

The remainder of this chapter describes how to implement authentication in the SilverStream environment.

About authentication

Authentication is the process of determining user identity. Some SilverStream applications identify users through a challenge, such as requesting a user ID/password pair, or through an authentication certificate. When an Anonymous user tries to access an object on your site, you can require a login or return an error. If you require a login, you can do so at the server level or the object level. Use the SMC to configure specific objects to require a login for access. Alternatively, use the **Require user authentication** setting at the server level to force users to log in when they first connect to the server.



For more information, see “Enabling authentication” on page 266.

Establishing a secure connection to the server

If you intend to use SSL communications, you need to obtain a server certificate and install it on the server. You also need to enable the RSA and/or DSA ports and disable HTTP if you want to require only SSL.

TIP RSA provides secure communications between the SilverStream server and Java clients, HTML clients, and EJBs. To simplify all SilverStream server SSL communications, you may want to use RSA.

When doing administrative tasks such as adding users and databases to the server, you may want a secure (SSL) connection between the server and the client you are using (such as the SMC, the SilverStream Designer, or perhaps a browser) so that all communication is encrypted.

The following three sections describe:

- Establishing a secure connection between a Java client and the SilverStream server
- Establishing a secure connection between an HTML client and the SilverStream server
- Establishing a secure (SSL) connection between an EJB client and the SilverStream server

Establishing a secure connection between a Java client and the SilverStream server

Secure communications between the SilverStream server and Java clients (such as the SMC, SilverJRunner, SilverJ2EEClient, the SilverStream Designer, or an external Java client) can use the RSA or DSA protocol.

Because you can configure three unique ports for each protocol, the port you specify in the following procedure depends on whether it is intended to be a runtime port for users, a design port for developers, or an administration port.



For more information, see “Setting up separate ports” on page 106.

➤ **To establish a secure (SSL) connection between a Java client and the SilverStream server:**

1. Install an RSA or DSA certificate on the SilverStream server.



For information, see “Using certificates” on page 226.

2. Enable the RSA or DSA port (or ports) in the SMC.



For information, see “Enabling RSA/DSA ports” on page 251.



3. Connect your client to the server using HTTPS at the DSA or RSA port.
The port you specify depends on what type of operations you want to run.
 - With the SMC, click the **Choose** (server) icon and specify the administration port.
 - With the SilverStream Designer, click the **Choose Server** icon and specify the design port (which will also let you test runtime operations).
4. In the resulting dialog, specify your server followed by the number of the runtime, design, or administration port you want to use.
 - For an RSA port, specify **https://server:RSA_port**
For example:
`https://tara:port`
 - For a DSA port, specify **https://server:DSA_port** on the command line for the hostname.
For example:
`https://tara:443`

NOTE If you want to use the RSA port default, you need only specify **https://hostname** on the command line. If you want an RSA connection on a port number other than the 443, you must specify the port value on the command line.

Establishing a secure connection between an HTML client and the SilverStream server

Secure communications between the SilverStream server and an HTML client (browser) uses the RSA protocol.

➤ To establish a secure (SSL) connection between an HTML client and the SilverStream server:

1. Install an RSA certificate on the SilverStream server.
 For information, see “Using certificates” on page 226.
2. Enable the RSA port in the SMC.
 For information, see “Enabling RSA/DSA ports” on page 251.

3. Open your browser to the server using HTTPS at the RSA port.

The RSA port you specify depends on what type of operations you want to perform. To run the application, specify the runtime port. If you use a custom HTML administration tool, specify the administration port.

Specify your server followed by the (optional) number of the RSA runtime or administration port:

```
https://server:port
```

For example:

```
https://tara:443
```

- NOTE** If you want to use the RSA port default, you need only specify **https://hostname** on the command line. If you want an RSA connection on a port number other than the 443, you must specify the port value on the command line.


Establishing a secure (SSL) connection between an EJB client and the SilverStream server

Secure communication between the SilverStream server and EJB clients is established using the IOP over SSL capabilities of the jBroker ORB. At startup, the SilverStream server exports the RSA certificate to the jBroker ORB. When the deployment plan of an EJB specifies a cipher suite and the SilverStream server has an RSA certificate installed, the ORB ensures that the communication is secure.



SilverJRunner, SilverJ2EEClient, and external Java clients require access to the agrootca.JAR file in order to participate in secure communications. This file is installed in the lib subdirectory of the SilverStream installation directory. This file is installed automatically for SilverJRunner and SilverJ2EEClient clients. External Java clients must use the AGROOTCA system level Java property to specify the location of this JAR file. For more information on setting this property, see the chapter on writing external Java clients in the *Programmer's Guide* of the server's Classic Development Help.

Communications failures may happen when:

- The server does not have an RSA certificate installed
- SilverJRunner or SilverJ2EEClient do not have the matching CA certificate of the server's certificate
- SilverJRunner or SilverJ2EEClient do not have the agrootca.jar file installed

 For information on specifying cipher suites for deployed EJBs, see the chapter on J2EE archive deployment in the *Facilities Guide*.

➤ **To establish a secure (SSL) connection between an EJB client and the SilverStream server:**

1. Install an RSA certificate on the SilverStream server.
 For information, see “Using certificates” on page 226.
2. Enable the RSA runtime port in the SMC.
 For information, see “Enabling RSA/DSA ports” on page 251.
3. For HTML or Java clients, connect your browser to the server using HTTPS at the RSA runtime port:

```
https://server:RSA_port_rt
```

For example:


```
https://tara:443
```

or

```
https://tara
```

NOTE If your RSA runtime port uses port 443 (the default), you need only to specify **https://hostname** on the command line. If you want an RSA connection on a port number other than 443, you must specify the port value.

For EJB applications that contain stateful session beans for which session-level failover is specified, you must also create a range of ports for IIOP SSL communications.

 For more information on creating the IIOP SSL port range, see “Specifying ORB settings” on page 116

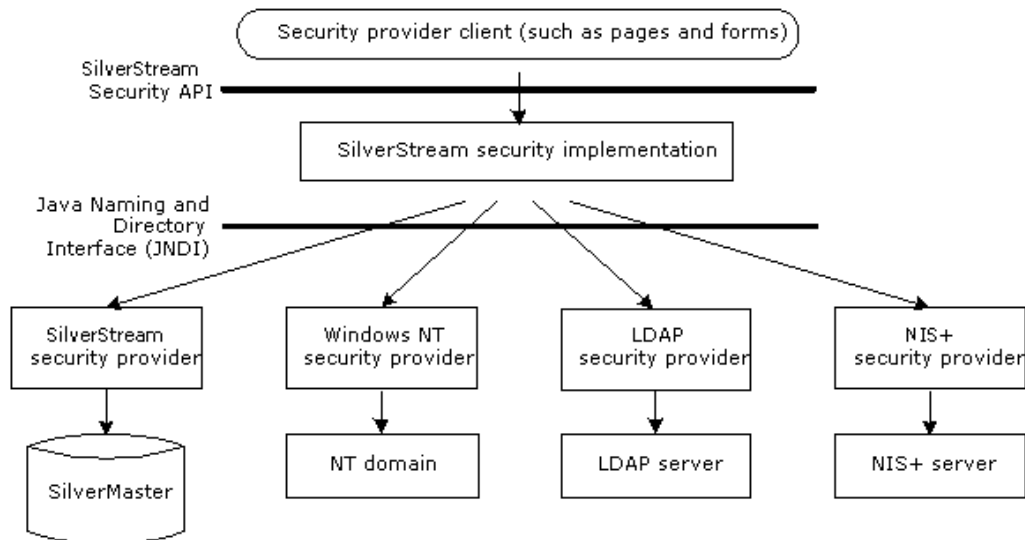
Accessing security provider systems

The system verifies users and their permission levels within a SilverStream application according to lists of groups and users that you provide. User and group information can be defined in SilverStream or can be obtained from an external security system. For SilverStream-defined entities, all information is stored in the SilverMaster database catalog. For external security, all information is obtained from the external system.

SilverStream recognizes users and groups from the following systems:

Security provider	Description
SilverStream Security	Native security that maintains a list of valid users and groups in the SilverMaster database.
Windows NT directory services	Capability to connect the SilverStream server to the NT Domain name registry.
Lightweight Directory Access Protocol (LDAP)	A directory service that connects the SilverStream server to defined LDAP directories.
NIS+	Network Information Services Plus, a name service that is available on SunOS 5.x and later operating systems.
X.509 certificates	SilverStream supports client certificates generated from authority servers such as VeriSign, Netscape Certificate Server, and Microsoft Certificate Server. For more information, see “Using certificates” on page 226.


SilverStream implements the Java Naming and Directory Interface (JNDI), which connects the SilverStream server to native UNIX security and to Windows NT and LDAP directories.



Adding security provider access

You can use the SMC to access security provider systems. After you set up access to provider directories, you can use users and groups in expressions to define access control.

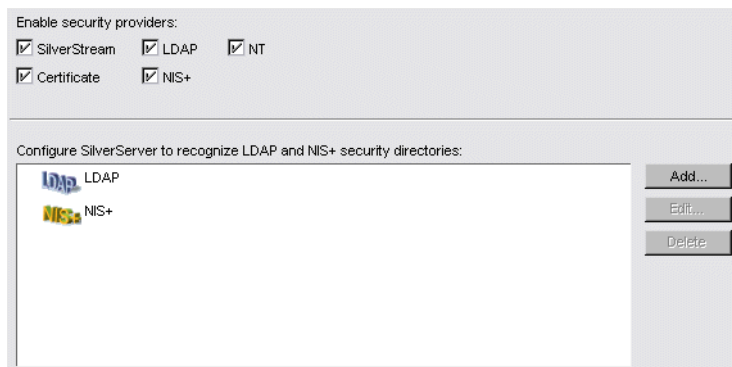
If you want to change from using Silver Security to a new security provider, make sure your administrator account has access permissions to the new security provider account. If you disable Silver Security before you grant the administrator account access to the new security provider, you will need to run `SilverMasterInit -l` to regain access to the SilverStream server.

 For information about setting up SilverStream users and groups, see Chapter 6, “Setting Up Users and Groups”. For information about access control, see “Authorization and access control” on page 271.



➤ To add security provider access:

1. Start the SMC.
2. Select the **Security** icon on the toolbar.

3. Select Security Providers.



Any LDAP and NIS+ servers that are known to the SilverStream server are listed.

4. Select the type of provider you want to register (all providers are selected by default).
 - NT is valid only when running Windows NT. If you choose NT, it is not necessary to add an NT domain; NT provides system calls that SilverStream uses to discover the primary and trusted domains. But you do need to set up the server so users can log in with their NT names. See “Using NT security” on page 211.
 - NIS+ is valid only when you are running Solaris.
5. To add an LDAP or NIS+ server connection, choose the appropriate item and click **Add**.
 -  For more information on adding the LDAP security provider, see “Using LDAP security” on page 213.
 -  For more information on adding the NIS+ security provider, see “Using NIS+ security” on page 220.

Resetting the security resource timeout

The SMC also allows you to set the security resource timeout period. This tells the server how frequently to reload the list of available users and groups from defined security providers.

➤ To reset security resource timeout:

1. Start the SMC.
2. Select the **Security** icon from the toolbar.

3. Select **General**.

The screenshot shows the configuration interface for SilverStream, specifically the 'General' tab. The interface includes several settings:

- Require user authentication
- Disable HTML directory listing
- Allow users to modify own account
- Security resource timeout (minutes):
- Default Security Realm:
- Default Security Authority:

4. Edit the **Security resource timeout** value as needed.

This setting determines how frequently the SilverStream server will upload current lists of users and groups from the NT, LDAP, and/or NIS+ servers, which would include any updates to these lists. The default value is 15 minutes.

You may want to increase this number if the information in the external system does not change frequently or if the connection to it is slow.

Using NT security

SilverStream allows you to use various NT directory services to manage NT users and NT groups. For example, users logging in with their NT user name and password only have to do this once per session (unless you set additional security at the server or cluster level).

Local and global groups

Using NT users and NT groups can help simplify security administration. For example, defining local groups lets you combine users and global groups from multiple domains into a single group.

A **local group** is available only within the domain in which it is created. A **global group** is available within its own domain as well as any trusted domain.

Local groups defined on the server machine can include local users as well as global groups and users from the primary domain or any trusted domain. An NT local group cannot, however, contain other local groups.



For more information on NT user groups, see your Windows NT documentation.

Speeding NT authentication Supporting local groups can result in slower NT authentication if you have many large trusted domains. If this is a problem for you, you can speed authentication by disabling local group support—in either of two ways:

- Programmatically setting the property `PROP_SUPPORT_NT_LOCAL_GROUPS` (in `AgiAdmServer` and `AgiAdmCluster`) to `Boolean.FALSE`.
- Adding the following line to the `httpd.props` file in the SilverStream Resources directory:

```
http-server.com.sssw.srv.SupportNTLocalGroups=false
```

NT privileges needed when running the server as a service

The SilverStream server requires two operating system privileges to support NT users and groups: **Act as Part of the Operating System** and **Log on as a Service**. These two privileges get set when the SilverStream server is configured as a service running under the default NT System account. However, if you change the service to run under a user account or if you decide to stop running the server as a service, you must make sure that these two NT system privileges are set.

If you change the service to run under a user account, the NT Control Panel automatically grants the **Log on as a Service** privilege to that account; however, you will need to manually configure the **Act as Part of the Operating System** privilege. To allow users from trusted domains to log in, you must configure the **Log on Locally** privilege regardless of whether or not the SilverStream server is running as a service.

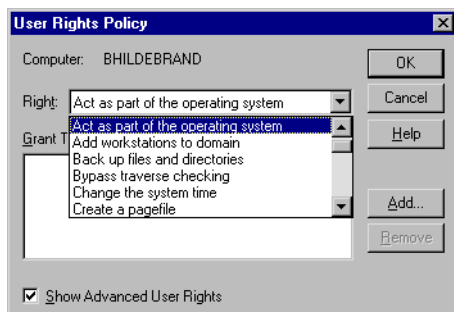
➤ To set up NT security:

1. Go to the Windows NT Start menu and choose **Programs>Administrative Tools**.
2. Start the **User Manager**.

If you are running NT Server, go to Step 3. If you are running an NT workstation, go to Step 5.

3. Select **User>Domain**.
4. Set the Domain field to the name of the machine in the dialog so that the rights being set will apply to this machine. Click **OK**.
5. Select **Policies>User Rights**. The User Rights Policy dialog displays.

6. Make sure the **Show Advanced User Rights** check box is checked.
- If you are running the SilverStream server under the default system account go to Step 9.
7. Select **Act as part of the operating system** from the dropdown list. All users and groups that have this privilege are listed in the **Grant To** field.



8. If the name of the account that launches the SilverStream server does not appear in this list, click the **Add** button and add the user to the list.
9. If your environment does not support trusted domains, go to Step 12.
10. Select **Log on Locally** from the dropdown list.
11. Add the desired users and groups from the trusted domains.
These users will now be able to log in to the SilverStream server.
12. Reboot your computer for these settings to take effect.

Using LDAP security

Lightweight Directory Access Protocol (LDAP) is a directory service that allows Internet clients to query and manage an arbitrary database of hierarchical attribute/value pairs over a TCP/IP connection. LDAP provides a specification that allows applications to communicate with it. The SilverStream eXtend Application Server allows you to specify LDAP users and groups, to display LDAP attributes, and to use LDAP users and groups in access control expressions.

Access to LDAP information

A SilverStream server interacts with an LDAP server in two distinct ways:

- When it needs to verify a user's credentials during login
In this case, the SilverStream server passes a **specific user's** login information to the LDAP server.

- When it needs to display **generic** information such as lists of users and groups
How (or whether) your SilverStream server accesses this generic information depends on how your LDAP server is configured:
 - If your LDAP server **does not allow anonymous access**, you can configure the SilverStream server to pass the system login credentials to the LDAP server. You need to provide these system credentials in Step 5 of the setup procedure that follows.
 - If your LDAP server **allows anonymous access**, no system login credentials need to be passed. You do not need to specify a system account for LDAP in Step 5.

Connecting to LDAP servers using SSL

To prevent information about LDAP groups and users (including client credentials) from being transferred as clear text, use an SSL connection between the SilverStream server and the LDAP server.

To use SSL communications with the SilverStream server, you must already have configured your LDAP server to support SSL and have installed certificates on the LDAP server.



For more information, see your LDAP server documentation.

NOTE When using SSL communications with LDAP, you can set the SilverStream server certificate to be sent to the LDAP server if it's requested. When the LDAP server is set to request or require certificates, it will attempt to verify any certificates sent to it.

Connecting to LDAP servers that only support LDAP Version 2

You can set the SilverStream server connection to use only the LDAP Version 2 protocol. By default, the SilverStream server first tries to connect to the LDAP server using LDAP Version 3. If the connection attempt fails, the LDAP server is supposed to report an error, in which case the SilverStream server will try to connect using the Version 2 protocol.

This approach will not work with an LDAP server (such as Microsoft Site Server) that doesn't always report the error using LDAP Version 3. If LDAP Version 3 is not supported, set the **Force LDAP Version 2** option in the SMC.

➤ To set up LDAP security:

1. Start the SMC.
2. Select the **Security** icon from the toolbar.
3. Select **Security Providers**.


4. Select **LDAP** in the provider list, then click **Add**.



A wizard displays.

5. Select **LDAP** and click **Next**.

The following panel displays.

6. Specify the server and (optionally) the login attribute and user name/password as follows.


Item	Description
Server	<p>The name of the LDAP server. The server name must be recognized on your network. If the LDAP server uses a nondefault port, you must specify it as part of the server name.</p> <p>For example, localhost:636.</p>
Use SSL	<p>Use this option if the specified LDAP server and port are configured to use SSL communications.</p> <p> For more information, see “Connecting to LDAP servers using SSL” on page 214.</p>


Item	Description
Send Certificate to Server	<p>This option allows the SilverStream server certificate to be sent to the LDAP server. When the LDAP server is set to request or require certificates, it will attempt to verify any certificates sent to it.</p> <p>If the LDAP server is set to request or require certificates (and this option is enabled), the SilverStream server certificate is sent to the LDAP server so that it can verify it against a list of trusted CA certificates. If the LDAP server is set to request or require a certificate (and the SilverStream server does not have one), the SilverStream server will ignore the Send Certificate to Server command for certificate requests. However, the connection will fail if the LDAP server requires a certificate.</p> <p>You can only select the Send Certificate to Server check box if you have also selected Use SSL.</p>
Force LDAP Version 2	<p>Set this option to work with an LDAP server (such as Microsoft Site Server) that does not support LDAP Version 3.</p> <p> For more information, see “Connecting to LDAP servers that only support LDAP Version 2” on page 214.</p>
User Login Attribute	<p>(Optional) If you specify a value for this property, it defines the LDAP attribute that can be used to uniquely identify a user. Make sure you pick an attribute that is unique for all users.</p> <p>TIP Specifying a value here can simplify login for LDAP users. For more information, see “Simplifying login for LDAP users” on page 224.</p>
User Name and Password	<p>If appropriate, enter a user name and password to allow the SilverStream server access to LDAP information. The SilverStream server will use these system login credentials anytime the SilverStream server needs to access generic LDAP server information.</p> <p>If your LDAP server supports anonymous access, these account values are not required.</p> <p> For more information, see “Access to LDAP information” on page 213.</p>

7. Click **Next**.

The following panel displays.

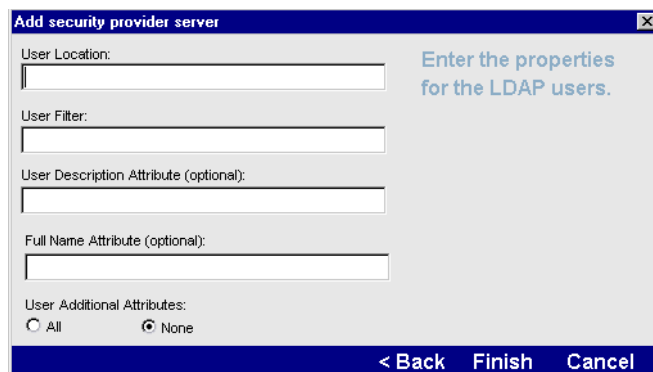
Use this panel to specify groups on the LDAP server.

Item	Description
Group Location	<p>(Required) A distinguished name that identifies the level in the hierarchy where you want to start searching for group entries. For example, to start at an organizational unit called employees that exists under the organization called silverstream, enter the following:</p> <pre>ou=employees,o=silverstream</pre> <p>Every group under and including employees in the hierarchy will be included.</p> <p> For more information about distinguished names, see “Simplifying login for LDAP users” on page 224.</p>
Group Filter	<p>(Required) The LDAP search filter is used to determine what constitutes a group for this LDAP server. A common usage is to specify a value of the object class attribute that identifies a group. The filter definition can be any valid LDAP search filter. For example:</p> <pre>(objectclass=groupofuniquenames)</pre>
Attribute of Group	<p>Required for LDAP servers (Netscape Directory Server) that use an attribute of a group object to define group membership.</p>


Item	Description
Attribute of User	Required for LDAP servers (like Microsoft Site Server) that use an attribute of a user object to define group membership.
Group/Users Attribute	(Required) An attribute used to display all members (users) of a group in the SMC. The name you enter is the LDAP group or user attribute that defines group membership. For example: uniquemember
Group Description Attribute	(Optional) An attribute used to identify a group description in the SMC. The name you enter is the LDAP attribute to which you want to map the description. For example: notes
Group Additional Attributes	Select All if you want all of the specified LDAP group attributes to be listed in the SMC. Select None if you want no additional attributes to appear. The specified attributes will be displayed in a tab when you select a group in the Users & Groups panel and open the Property Inspector.  For more information, see “Accessing users and groups” on page 220.

8. Click **Next** when you have finished specifying groups.

The next panel asks you to specify users on the LDAP server.



9. Specify users as follows:

Item	Description
User Location	<p>(Required) A distinguished name that identifies the point in the hierarchy where you want to start searching for users. For example, to start at a point (or node) entitled developers that exists under software, enter the following:</p> <pre>ou=developers,o=software</pre> <p>Every user under and including developers in the hierarchy will be included.</p>
User Filter	<p>(Required) The LDAP search filter is used to determine what constitutes a user for this LDAP server. A common usage is to specify a value of the object class attribute that identifies a user. The filter definition can be any valid LDAP search filter. For example:</p> <pre>(objectclass=person)</pre>
User Description Attribute	<p>(Optional) An attribute used to identify a user description in the SMC. The name you enter is the LDAP attribute to which you want to map the description. For example:</p> <pre>title</pre>
Full Name	<p>(Optional) Specifies the full name attribute, if available. For example:</p> <pre>cn</pre>
Additional Attributes	<p>Select All if you want all of the specified LDAP user attributes to be listed in the SMC. Select None if you want no additional attributes to appear.</p> <p>The specified attributes will be displayed in a tab when you select a user in the Users & Groups panel and open the Property Inspector.</p> <p> For more information, see “Accessing users and groups” on page 220.</p>

10. Click **Finish**.

The SMC displays the settings under the LDAP directory. You can view the new settings anytime by selecting **Users & Groups** in the Security options in the SMC.

Using NIS+ security

NIS+ (Network Information Services Plus) is a name service available on SunOS 5.x and higher operating systems. Users are contained in the **NIS+** table identified by `passwd.org_dir`, and groups by `group.org_dir`. After you have added users and groups, you can use them in security expressions for access control.

➤ **To set up NIS+ security:**

1. Start the SMC.
2. Select the **Security** icon from the toolbar.
3. Select **Security Providers**.
4. Select **NIS+** from the provider list, then click **Add**.
A wizard displays.
5. Select **NIS+** and click **Next**.
6. Type the name of the **NIS+** server in the following format:

`servername/nisDomain.com\username`

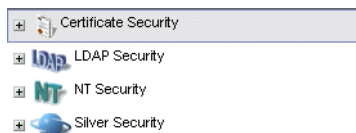
The server name must be recognized on your network.

Accessing users and groups

You can use the SMC to view users and groups that you have defined for any security provider.

➤ **To view users and groups:**

1. Start the SMC.
2. Select the **Security** icon from the toolbar.
3. Select **Users & Groups**.
4. Highlight the appropriate icon to view the users and groups known to the server.

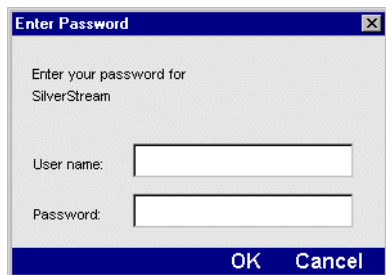


You can expand a selected item to show specific users and groups.

Using security provider login formats

SilverStream supports a number of security realms including SilverStream security, NT security, LDAP security, NIS+ security, and certificate security. All except certificate security involve establishing an identity by providing user name information and a password.

When users log in, they see a dialog similar to the following.



If the authentication dialog is being presented by a browser, the dialog—defined by the browser—will look a little different from the dialog shown above but will consist of the same fields.

Colons cannot be used in user names or passwords

User authentication in HTTP works by taking the user name and password separated by a colon (:). So make sure that user names and passwords don't contain colons. With LDAP distinguished names in particular, which can be quite long, make sure no component of the name contains a colon.

User name components

In SilverStream, a user name is composed of three parts delimited by backslash characters: Realm\Authority\Name.

Component	Description
Realm	SilverStream supports the following security provider realms for login: <ul style="list-style-type: none"> • SSSW (for SilverStream users) • NT • LDAP • NIS+
Authority	Authorities are as follows: <ul style="list-style-type: none"> • For SilverStream, there is no authority • For NT, the authority is the NT domain • For LDAP, the authority is the server • For NIS+, the authority is the server name and the domain name, separated by a forward slash (/)
Name	The user name

User name shortcut formats By default, the SilverStream server allows shortcuts to the full user login name, as follows:

- **SilverStream** If the user enters one part for a user name, a SilverStream user name is assumed. For example:
 - emilyh** is translated to **SSSW\emilyh**

NOTE For SilverStream security, the authority is an empty string—nothing between the two backslashes; the authority is not needed, because an external security system is not used.
- **Windows NT** If the user enters two parts to a user name, an NT user name is assumed, with the format *domain\userName*. For example:
 - mydomain\craigh** is translated to **NT\mydomain\craigh**

By default, LDAP and NIS+ names must be fully qualified, as follows:

- **LDAP** Login syntax for user name: **LDAP\serverName\distinguishedName**

The user must enter the entire path name. For example:

```
LDAP\myServer\cn=Nancy Smith,ou=SilverStream Software
```

- **NIS+** Login syntax for user name: **NisPlus\server/nisDomain\username**

The user must enter the entire path name. Note that the authority has two components, separated by a forward slash. For example:

```
NisPlus\myServer/domain1.com\jeanw
```

You can change these default login specifications, as described next.

Overriding defaults for login name components

You can override the defaults for login name components. So if you use only one type of external security system (and perhaps only one external security authority), you can allow users to provide a shortened name, thus simplifying the login procedure.

➤ To override the defaults for login name components:

1. Start the SMC.
2. Select the **Security** icon from the toolbar.
3. Select **General**.
4. Specify your default realm and (optionally) your default authority.

Field	Description
Default Security Realm	Defines the security realm for any login name that does not explicitly define a realm. Possible values are SilverStream, NT, LDAP, NIS+, or No Default Specified.
Default Security Authority	(Enabled only if a default security realm has been defined) Defines the authority for any login name that does not explicitly define an authority. SilverStream provides a list of valid authorities based on the selected default realm.

A full login name can always be specified, in which case the defaults are ignored.

Example Suppose your site uses security names from a single LDAP server. You could set the following defaults:

Field	What you specify
Default Security Realm	LDAP
Default Security Authority	<i>ServerName</i>

Users that exist in the LDAP server can now just use their LDAP user name and password when logging into the SilverStream server.

In this same example, a user that exists as part of the SilverStream security realm must now specify a full login name:

```
SSSW\SilverName
```

Simplifying login for LDAP users

In LDAP, a user name is specified relative to the LDAP naming hierarchy as a **distinguished name** (DN). The DN is a comma-separated list of nodes that contain attribute/name pairs from the leaf node where the user resides back to the root node.

By default, LDAP users logging in to the SilverStream server are required to enter the entire DN, which can be quite long. You can simplify login for LDAP users by specifying the User Login Attribute property when adding an LDAP server as a security provider. For more information on this property, see “Using LDAP security” on page 213.

If you have specified a User Login Attribute, when the user credentials are being verified during the login sequence, a search is performed for the specified User Login Attribute, with a value that matches the Name portion of the login user name. The search starts from a point in the LDAP hierarchy identified as User Location when defining the LDAP server to SilverStream (see “Using LDAP security” on page 213).

If the search is successful, the DN of the corresponding user (the first one found if there are multiple hits) is used to construct a fully qualified login name, and the login operation continues. If the search is not successful, the operation continues as though the Name field were the distinguished name of the LDAP user. This allows LDAP logins using either form when the attribute is set.

Example 1 Assume the following server properties are specified:

Field	What you specify
Default Security Realm (specified in Server Security panel in SMC)	LDAP
User Login Attribute (specified when defining the LDAP server to SilverStream)	mail
User Location (specified when defining the LDAP server to SilverStream)	o=SilverStream Software,c=US

In this example, a default security realm is defined and the login attribute is set to **mail**. At this site, each user's **mail** attribute is that user's e-mail address.

A user defined in an LDAP server named myServer, with a distinguished name of **uid=ecraig,ou=Development,ou=BillERICA,o=SilverStream Software,c=US** and an e-mail address of **ecraig@silverstream.com** (that is, a user whose **mail** attribute is **ecraig@silverstream.com**), could use a login name of either:

```
myServer\uid=ecraig,ou=Development,ou=BillERICA,o=SilverStream
Software,c=US
```

or

```
myServer\ecraig@silverstream.com
```

Example 2 Assume the following server properties are specified:

Field	Value
Default Security Realm (specified in Server Security panel in SMC)	LDAP
Default Security Authority (specified in Server Security panel in SMC)	myServer
User Login Attribute (specified when defining the LDAP server to SilverStream)	uid
User Location (specified when defining the LDAP server to SilverStream)	o=SilverStream Software,c=US

In this example, a default security authority has now been specified in addition to the default security realm. The login attribute has now been set to **uid**.

The same user as above (whose **uid** is **ecraig**) could use a login name of either:

```
uid=ecraig,ou=Development,ou=Billerica,o=SilverStream Software,c=US
```

OR

```
ecraig
```

Using certificates

Certificates are required when using HTTP with the Secure Sockets Layer (SSL) 3.0 protocol (HTTPS). HTTPS provides data encryption between the SilverStream client and server to ensure privacy and data integrity. Certificates can also be used to authenticate users of SilverStream applications.

This section contains the following topics:

- About certificates
- Creating and installing server certificates using the SMC
- Creating and installing server certificates from the command line
- Viewing server certificates
- Enabling RSA/DSA ports
- Turning off HTTP communications
- Restricting SSL cipher suites
- Managing Certificate Authorities
- Installing and managing client certificates
- Verifying SSL server certificates for Java clients

About certificates

A **certificate** (also called a public-key certificate, digital ID, or digital certificate) is a file that authenticates the identity of a user or a group. The certificate is a kind of license issued by a trusted organization called a Certificate Authority (CA). A CA may be an external company that offers certificate services (such as VeriSign) or an internal organization such as a (corporate MIS department).

For Internet applications, it is generally a good idea to have a server certificate that is signed by a widely recognized and trusted guarantor. For intranet applications, it may be sufficient to have the guarantor be the company in which the application is running.

Both users and servers can have certificates attesting to their identity. If you want to use SSL for privacy, the SilverStream server **must** have a server certificate. Once enabled, the server may request a client certificate from the browser attesting to the identity of the user.

Advantages of certificates

Certificates provide these important security services:

Service	Description
Enhanced authentication and security	Clients can be assured that they are communicating with who they think they are. Similarly, applications can be sure of who their users are. Certificate-based authentication is more secure than traditional methods of user authentication, such as user name/password.
Real-time encryption over SSL	SSL's encryption scheme requires the server to present its digital certificate to the client as part of the SSL handshake. Verifying the server's certificate gives the client a level of trust about the server's identity.
Convenience	Certificates allow a user to use a single user login over the intranet and across the Internet. Users log in once to the browser, for example (a local operation), and then the browser presents client certificates to servers as needed.

Certificate support in SilverStream

The following is a summary of SilverStream certificate support:

Support item	Description
Server certificates	<p>A server certificate is required for SSL/HTTPS. This allows clients to authenticate the server. There are two types of certificates, depending on the type of client:</p> <ul style="list-style-type: none"> • SilverStream supports RSA-encoded server certificates for HTTPS/SSL communications between the SilverStream server, Java clients, and HTML clients. • SilverStream supports Diffie-Hellman (DSA) server certificates for HTTPS/SSL communications between the SilverStream server and SilverStream Java clients and non-SilverStream Java clients. <p>NOTE SilverStream Java clients verify server certificates against a list of trusted CA certificates that are stored in a JAR file when establishing an SSL connection to the SilverStream server. See “Verifying SSL server certificates for Java clients” on page 264.</p>
Client certificates	<p>Client certificates are optional and are used for user authentication by the server. They are installed in a browser. You can get client certificates from a number of authorities, including VeriSign. Each client certificate includes the Certificate Authority (CA) certificate that generated it. The server must have a corresponding CA certificate.</p> <p>NOTE The SilverStream server does not support DSA client certificates.</p>
CA certificates on the server	<p>CA certificates represent trusted clients based on the CA who signed for them. CA certificates are required on the server for verifying corresponding HTML client certificates. The server will only authenticate client certificates that were generated and/or signed by one of its installed CA certificates.</p>

About global certificates

What are commonly called **global certificates** are actually **Global Secure Site IDs** from VeriSign. They are a form of digital ID that allows for 128-bit encryption worldwide. (Standard VeriSign digital IDs, now called Secure Site IDs, do not allow for U.S.-based companies to use 128-bit encryption outside the U.S.)

It is up to VeriSign to certify servers as supporting Global Secure Site IDs; it is not up to a server vendor—such as SilverStream—to declare support for Global Secure Site IDs.



For more information, see

<http://digitalid.verisign.com/server/global/help/globalFAQ.htm>.

Creating and installing server certificates using the SMC

You can use the SMC for these tasks:

- Generating an RSA server certificate
- Generating a DSA server certificate from the command line

Generating and installing RSA server certificates

The SMC provides the following functions for RSA server certificates:

Feature	Description
Generate Request	<p>Generates a certificate signing request (CSR) for an RSA certificate.</p> <p>The CSR is an encrypted file that contains information that identifies the organization running the SilverStream server. The CSR is submitted to a certificate issuer (such as VeriSign) that will use the information to create a certificate signed by the issuer.</p> <p>Generates the public/private key combination and stores the private key information in the database. Private key information is never passed from the server to the SMC, so this is extremely secure.</p> <p>This is similar to running the AgDigitalIDStep1 utility.</p>

Feature	Description
Install Certificate	<p>Prompts for the certificate information returned by the issuer and then installs the certificate (without private key information) into the SilverStream server.</p> <p>It does not install the private key information, because that already resides on the server from the Generate Request process.</p> <p>If the server already has an RSA certificate for the same DNS name, installing another certificate for the same DNS name will overwrite the existing one.</p> <p>This function is similar to running the AgDigitalIDStep2 utility.</p>
Import/Export Certificate	<p>Export allows you to export an installed certificate and private key to a specified file on the server. The certificate and private key are exported in the standard PKCS12 format. Used for backup.</p> <p>You are prompted to provide a password for the exported file.</p> <p>Import allows you to import an exported certificate and private key. Import installs the certificate on the server and will overwrite an existing certificate if the same DNS names are specified.</p> <p>When importing a certificate that was exported, you are prompted for a password. This is the same password used on the export.</p>
Export Private Key	<p>Writes the private key to the specified file on the server machine in the PKCS8 format. You are prompted for a password to protect the private key.</p> <p>Use this to back up your private key after using Generate Request.</p>

Feature	Description
Install with Private Key	<p>Prompts for the certificate information returned by the issuer and the private key file (generated via the Export Private Key function), then installs the certificate (with the private key information) into the SilverStream server.</p> <p>Use this feature when you've generated a CSR and are unable to use the private key installed via the Generate Request feature (for example, if the server's database has become corrupted before you receive the certificate back from the CA).</p>

NOTE To generate a certificate for a dispatcher, you must use the command line tool described in "Creating and installing server certificates from the command line" on page 240.

➤ **To generate an RSA server Certificate Signing Request (CSR):**

1. Start the SMC.
2. Select the **Security** icon from the toolbar.
3. Select **Certificates**.
4. Select the **RSA** tab.
5. Choose **Generate Request**.
6. Complete the items on the panel that displays as follows.

Field	What to specify
Server DNS Name	The TCP/IP hostname, which may be different from your machine name. (You can issue ping localhost from the command line to determine the TCP/IP name of the local host.)
Organization	Your company name in full unabbreviated legal form.
Organizational Unit	(Optional) Your department within the company.
City/Locale	(Optional) The city or locale where your company does business.

Field	What to specify
State/Region	The full name of the state or region where your company does business. Do not abbreviate.
Country	The country where your company does business. You must use the ISO two-letter country code. For example, the ISO code for the United States is US.

7. Click **Next**.

The following panel allows you to specify the size of the key pair to generate.

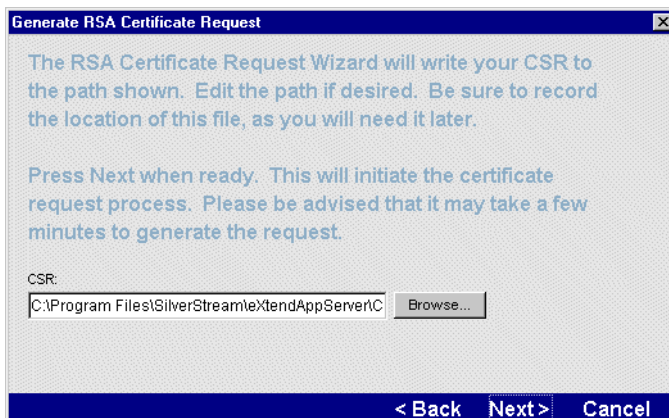
The 1024 bits option usually provides an acceptable level of security. Selecting a higher level decreases the speed of the initial connection. The 512 bits option provides a low level of security.



8. If prompted, specify the size of the key pair to generate.

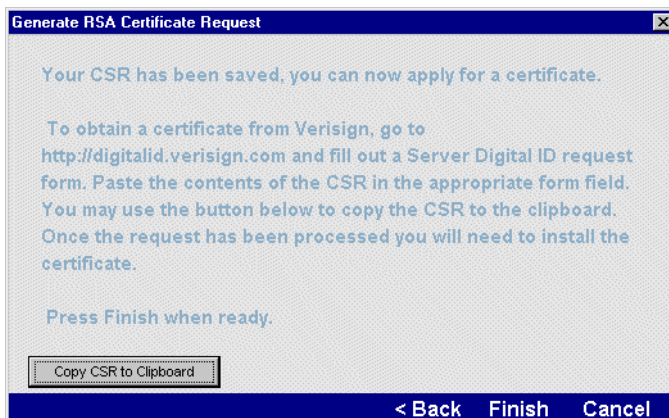
9. Click **Next**.

The following panel displays.



This panel shows the paths for the CSR (Certificate Signing Request). You may edit these paths if you choose. You will use this information later when installing the certificate.

10. Click **Next**.



11. Click **Copy CSR to Clipboard** to copy the contents of the CSR to the clipboard for use in the next step.
12. Follow the directions to request a certificate for your SilverStream server (for example, by using the VeriSign Web site <http://digitalid.verisign.com>). Once your request is approved, the certificate authority sends the new certificate back to you through e-mail.

13. Click **Finish**.

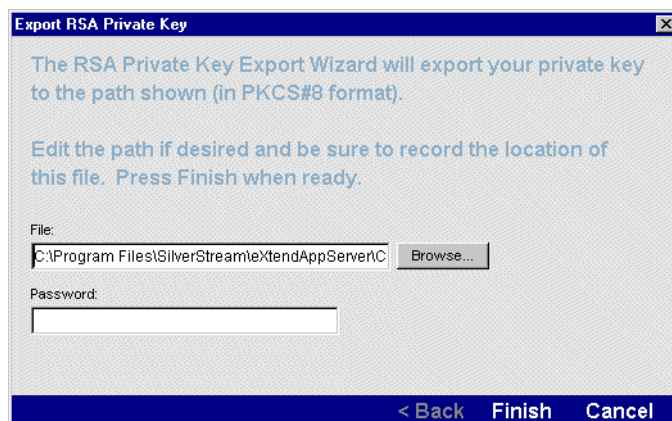
You may want to use the **Export Private Key** option to back up the private key for this CSR.

➤ **To back up the private key:**

After you have generated a CSR, you may want to back up the private key information in case the SilverMaster database gets corrupted before you get the CSR back from the certificate issuer. If you've saved the private key, you'll still be able to install the certificate.

1. Start the SMC.
2. Select the **Security** icon from the toolbar.
3. Select **Certificates**.
4. Select the **RSA** tab.
5. Choose **Export Private Key**. A message box displays advising you to use HTTPS.

The following panel displays.



6. Supply the path and file name where you want to store the RSA private key.
7. Supply a password for this file. This does not have to be the administrator's password—it just applies to the file containing the private key information.
8. Click **Finish**.

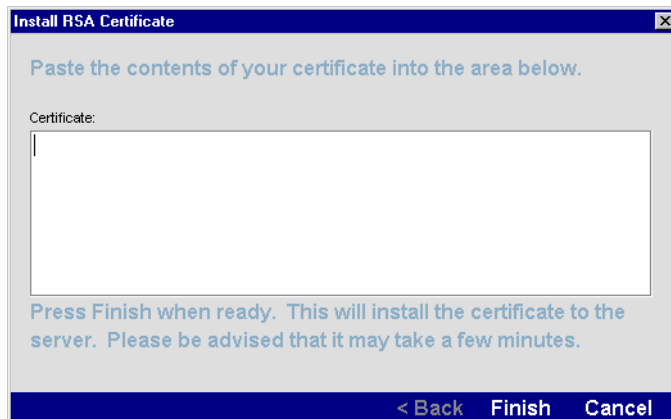
➤ **To install a certificate (with or without a private key):**

1. Start the SMC.
2. Select the **Security** icon from the toolbar.
3. Select **Certificates**.
4. Select the **RSA** tab.

5. Choose **Install Certificate** or **Install with Private Key**.

If you chose **Install with Private Key**, a message box displays and you are advised to use HTTPS for this procedure.

The following panel displays:



6. Paste the signed certificate into the text area and click **Finish**.

If you chose **Install with Private Key**, you are prompted for the file containing the private key and the password associated with the file.

1. Browse to the files location.
2. Enter the password and click **Finish**.

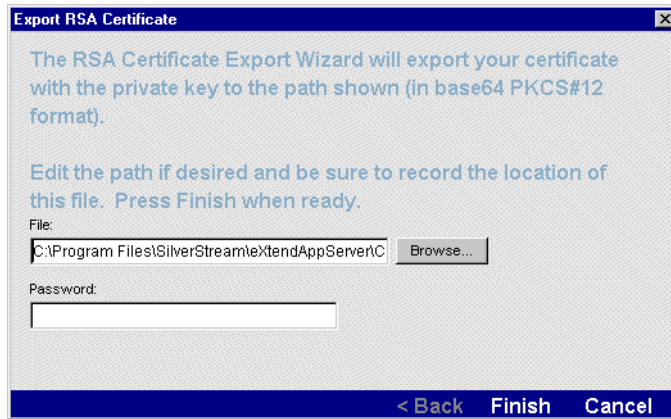
The SMC displays a message that the update was successful.

7. Click **OK**.
8. Choose **Restart** to have the changes take effect.

➤ **To back up an RSA certificate:**

1. Start the SMC.
2. Select the **Security** icon from the toolbar.
3. Select **Certificates**.
4. Select the **RSA** tab.

5. Choose **Export Certificate**. You are advised to run in HTTPS mode.
The following panel displays.



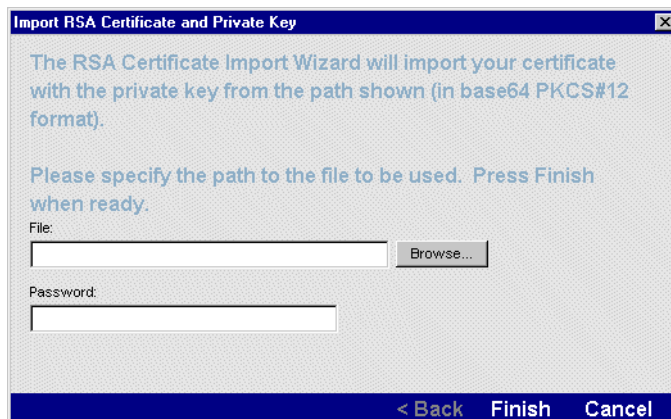
6. Specify the name and location of the backup file.
7. Specify a password to protect the file.
8. Click **Finish**.

➤ **To import (install) an RSA certificate:**

1. Start the SMC.
2. Select the **Security** icon from the toolbar.
3. Select **Certificates**.
4. Select the **RSA** tab.

5. Choose **Import Certificate**. You are advised to run in HTTPS mode.

The following panel displays.



6. Specify the name and location of the certificate file.
7. Specify the password used to protect the file. (This is the same password used to export the certificate.)
8. Click **Finish**.

Generating and installing DSA server certificates

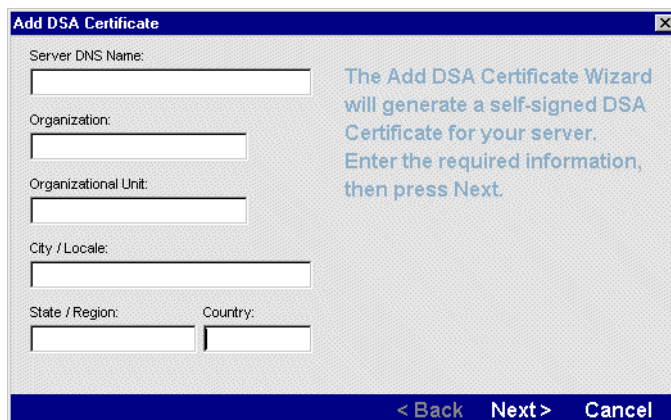
You can use the SMC to generate and install a DSA server certificate.

➤ To generate and install a DSA server certificate:

1. Start the SMC.
2. Select the **Security** icon from the toolbar.
3. Select **Certificates**.
4. Select the **DSA** tab.

5. Choose **Add Certificate**.

The following panel displays:

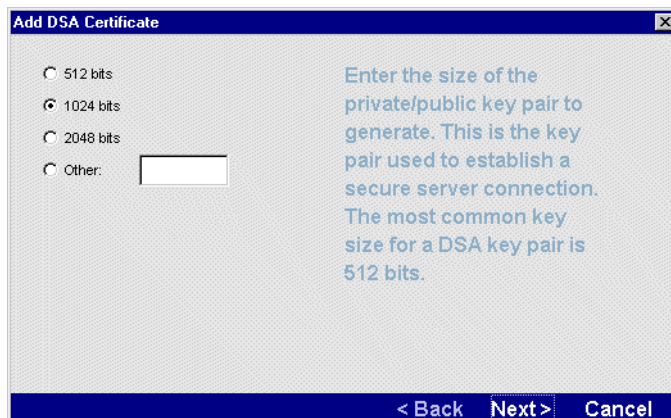


6. Complete the items on this panel as described below.

Field	What to enter
Server DNS Name	The TCP/IP hostname, which may be different from your machine name. (You can issue ping localhost from the command line to determine the TCP/IP name of the local host.)
Organization	Your company name in full unabbreviated legal form.
Organizational Unit	(Optional) Your department within the company.
City/Locale	(Optional) The city or locale where your company does business.
State/Region	The full name of the state or region where your company does business. Do not abbreviate.
Country	The country where your company does business. You must use the ISO two-letter country code. For example, the ISO code for the United States is US.

7. Click **Next**.

8. The following panel allows you to specify the size of the key pair to generate. The 1024 bits option usually provides an acceptable level of security. Selecting a higher level decreases the speed of the initial connection. The 512 bits option provides a low level of security.



9. If prompted, specify the size of the key pair to generate.
10. Click **Next**.

The following panel displays:



11. Click **Finish**.

Creating and installing server certificates from the command line

As an alternative to the SMC, you can create and install server certificates from the command line outside the SilverStream environment using the following executables:

Executable	Description
AgDigitalIDStep1	Used to prepare either of the following: <ul style="list-style-type: none"> • A Certificate Signing Request (CSR) for an RSA certificate. The CSR is an encrypted file that contains information that identifies the organization running the SilverStream server. The CSR is submitted to a certificate issuer (such as VeriSign) that will use the information to create a certificate signed by the issuer. • A self-signed DSA certificate that identifies the organization running the SilverStream server.
AgDigitalIDStep2	Prompts for the certificate information returned by the issuer as well as the private key used to protect the data, then installs the certificate (without private key information) into the SilverStream server.

The command line executables do not provide as many features as the SMC does. The following sections describe the command-line procedures:

- Generating an RSA server certificate
- Generating a DSA server certificate from the command line
- Installing the certificate

Generating an RSA server certificate

To enable HTTPS/SSL communications between the SilverStream server, Java clients, and HTML clients you need to install an RSA certificate on the server.

➤ To generate an RSA server certificate for use with Java and HTML clients:

1. Change the working directory to your SilverStream \bin directory.

- At the command line, specify the following command:

```
AgDigitalIDStep1
```

The following panel displays:

NOTE In UNIX, this utility runs using a GUI and cannot be run in a character terminal window. If you log in remotely to the UNIX machine, make sure you set your `DISPLAY` environment variable appropriately.

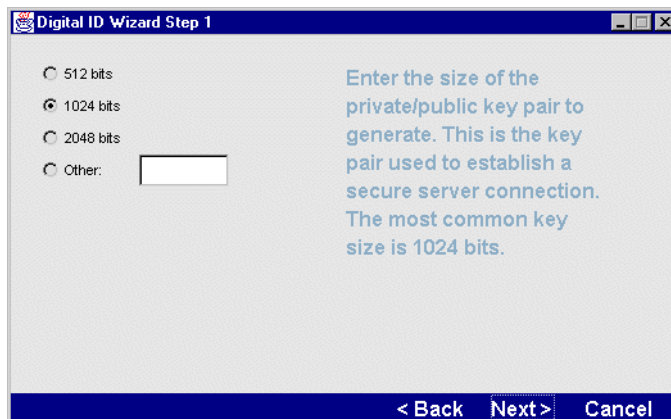
- Complete the items on this panel as follows.

Field	What to specify
Server DNS Name	The TCP/IP hostname, which may be different from your machine name. (You can issue ping localhost from the command line to determine the TCP/IP name of the local host.)
Organization	Your company name in full unabbreviated legal form.
Organizational Unit	(Optional) Your department within the company.
City/Locale	(Optional) The city or locale where your company does business.
State/Region	The full name of the state or region where your company does business. Do not abbreviate.
Country	The country where your company does business. You must use the ISO two-letter country code. For example, the ISO code for the United States is US .

4. Click **Next**.

The following panel displays for you to specify the size of the key pair to generate.

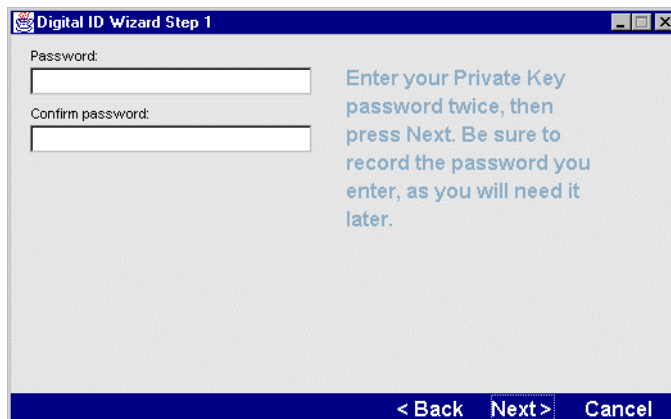
The 1024 bits option usually provides an acceptable level of security. Selecting a higher level decreases the speed of the initial connection. The 512 bits option provides a low level of security.



5. If prompted, specify the size of the key pair to generate.

6. Click **Next**.

The following panel displays.

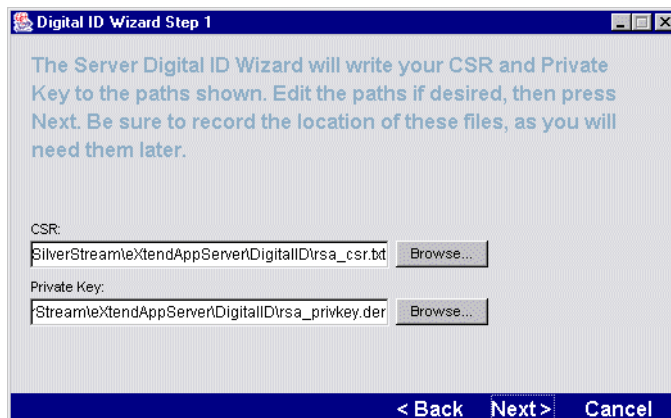


7. Enter a password, then confirm it. This password will be used to encrypt your private key.

TIP Be sure to record this password; you will need it to install the certificate into the server.

8. Click **Next**.

The following panel displays.

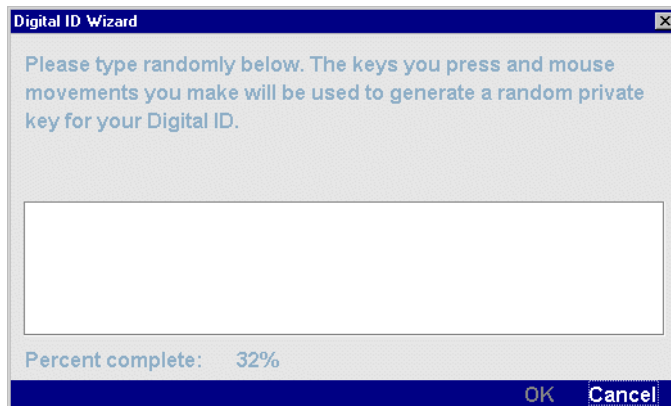


This panel shows the paths for the CSR (Certificate Signing Request) and the password-protected private key. You may edit these paths if you choose. You will use this information later when installing the certificate.

NOTE The file that contains the private key must be kept physically secure. Otherwise, anyone who can obtain the server's certificate can masquerade as the server.

9. Click **Next**.

If the wizard hasn't been able to collect enough randomness information to generate cryptographically good keys from the key presses and mouse movements you have made, the following panel displays.

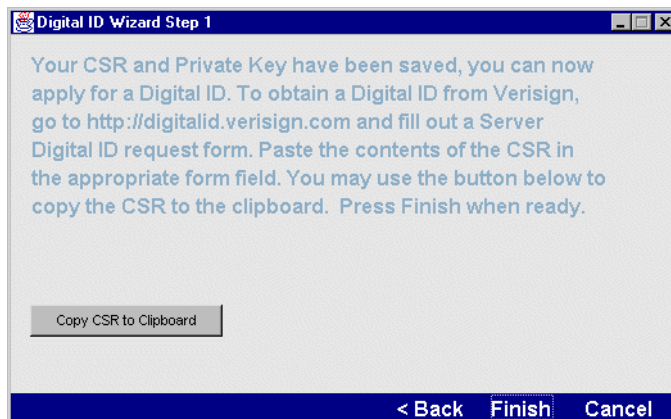


10. If prompted, type random characters in the edit box and move the mouse around to create an encrypted private key. When the wizard has enough randomness information, the **OK** button is enabled.

The wizard generates the certificate signing request and private key.

11. Click **OK**.

The following panel displays.



12. Click **Copy CSR to Clipboard** to copy the contents of the CSR to the clipboard for use in the next step.
13. Follow the directions to request a certificate for your SilverStream server (for example, by using the VeriSign Web site <http://digitalid.verisign.com>). Once your request is approved, the certificate authority sends the new certificate back to you through e-mail.
14. Click **Finish**.

You use AgDigitalIDStep2 to install the certificate. See “Installing the certificate” on page 248.

Generating a DSA server certificate from the command line

To enable HTTPS/SSL communications between a Java client and the SilverStream server, you can install either an RSA certificate (described above) or a DSA certificate as described here. The SilverStream server does not support DSA client certificates, and browsers do not support DSA server certificates.

NOTE You can use the SilverStream server RSA port to provide secure communications between the SilverStream server and all Java clients, HTML clients, and EJBs. To simplify your SSL communications configuration, you may want to use RSA certificates only.

➤ **To generate a DSA server certificate for use with Java clients:**

1. Change the working directory to your SilverStream \bin directory.
2. At the command line, specify the following command:

```
AgDigitalIDStep1 dsa
```

The following panel displays:

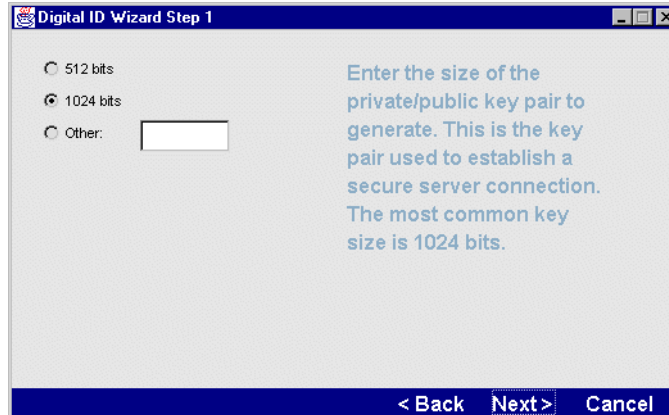
NOTE In UNIX, this utility runs using a GUI and cannot be run in a character terminal window. If you log in remotely to the UNIX machine, make sure you set your DISPLAY environment variable appropriately.

3. Complete the panel as follows.

Field	What to specify
Server DNS Name	The TCP/IP hostname, which may be different from your machine name. (You can issue ping localhost from the command line to determine the TCP/IP name of the local host.)
Organization	Your company name in full unabbreviated legal form.
Organizational Unit	(Optional) Your department within the company.
City/Locale	(Optional) The city where your company does business.

Field	What to specify
State/Region	The full name of the state or region where your company does business.
Country	The country where your company does business. You must use the ISO 2-letter country code. For example, the ISO code for the United States is US.

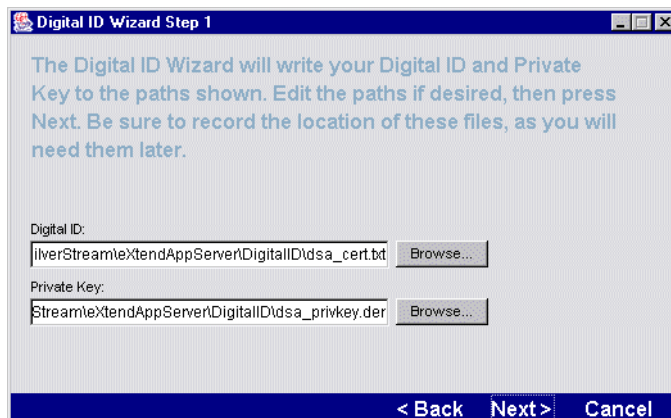
4. Click **Next**. The following panel allows you to specify the size of the key pair to generate. The 1024 bits option usually provides an acceptable level of security. Selecting a higher level decreases the speed of the initial connection. The 512 bits option provides a low level of security.



5. If prompted, specify the size of the key pair to generate.

6. Click **Next**.

The following panel displays.

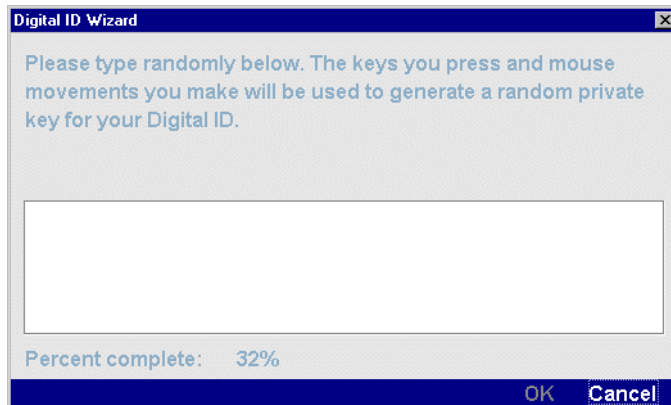


This panel shows the paths for the certificate and the private key. You may edit these paths if you choose. You will use this information later when installing the certificate.

NOTE The file that contains the private key must be kept physically secure. Otherwise, anyone who can obtain the server's certificate can masquerade as the server.

7. Click **Next**.

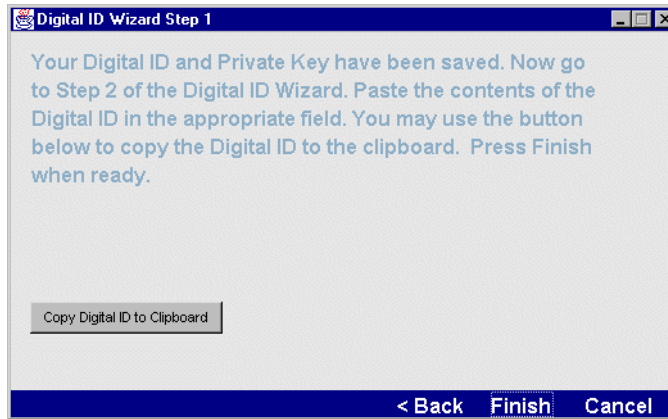
If the wizard hasn't been able to collect enough randomness information to generate cryptographically good keys from the key presses and mouse movements you have made, the following panel displays.



8. If prompted, type random characters in the edit box and move the mouse around to create an encrypted private key. When the wizard has enough randomness information, the **OK** button is enabled.

The wizard generates the certificate.

9. Click **OK**. The following panel displays.



10. Copy the certificate to the clipboard for use when installing the certificate.
11. Click **Finish**.

You use AgDigitalIDStep2 to install the certificate. See “Installing the certificate” next.

Installing the certificate

Once you receive your RSA certificate from the CA or have generated a DSA certificate using AgDigitalIDStep1, you can install the certificate using the AgDigitalIDStep2 program.

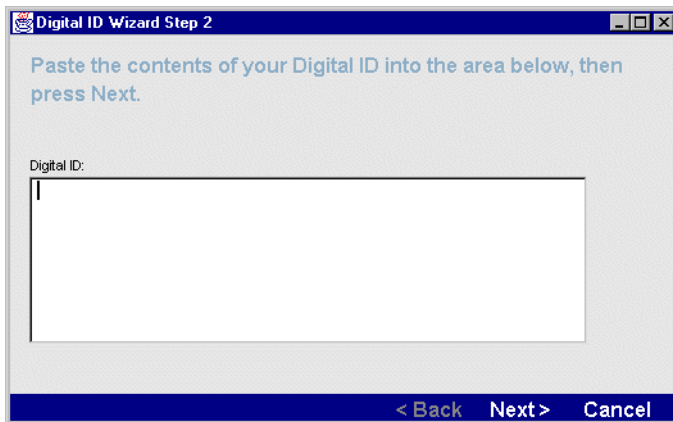
➤ To install the certificate:

1. Make sure the SilverStream server is running.
2. Change the working directory to your SilverStream **\bin** directory.

3. At the command line, specify the following command:

```
AgDigitalIDStep2
```

The following panel displays:



4. Paste the certificate into the panel.
5. Click **Next**.

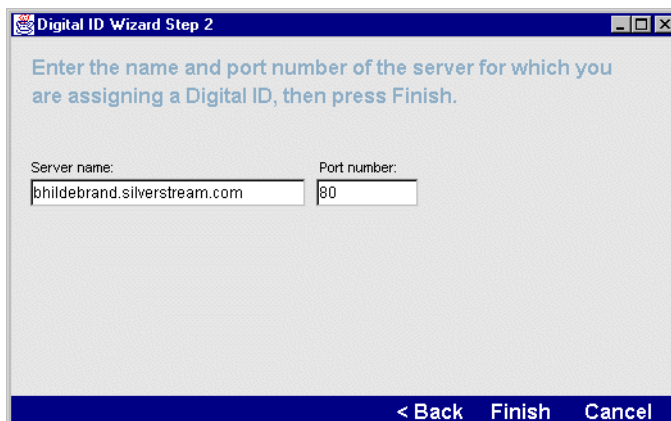
The following panel displays.



This panel asks for the path for the private key (you may need to edit the name of the private key, since the names are different for RSA and DSA certificates) and asks you to confirm your password (there is no password for a DSA private key).

6. Click **Next**.

The following panel displays.



7. Enter your server name and the HTTP port number to use.

If you have configured separate ports for different types of operations, specify your **administration** port. By default, the SilverStream server listens to port 80.

8. Click **Finish**. A confirmation message displays.

9. To activate the certificate, click the **Restart** (server) button on the SMC.

When the server is restarted, it is configured to listen using HTTPS at the RSA or DSA port(s), depending on which kind of certificate you just installed (in addition, by default, to listening for HTTP requests at the HTTP port).

 For more information, see “Enabling RSA/DSA ports” on page 251 and “Specifying general server properties” on page 110.

When you are ready for production, use the SMC to enable authorization. See “Enabling authentication” on page 266.

Viewing server certificates

➤ **To view certificates that have been installed on the server:**

1. Start the SMC.
2. Select the **Security** icon from the toolbar.
3. Select **Certificates**.

4. Select the **Certificate List** tab.
5. Choose a certificate from the dropdown.

Enabling RSA/DSA ports

By default, SilverStream specifies port 443 for RSA and DSA communications. You enable and modify ports for runtime, design, and administrative access for each of the following three security protocols: HTTP, HTTPS-RSA, and HTTPS-DSA. The server does not require you to configure unique port values for the different types of access; ports having the same value will share the same socket and will allow multiple operations.



For more information, see “Setting up separate ports” on page 106.

NOTE You can configure a SilverStream server RSA port to provide secure communications between the SilverStream server and all Java clients, HTML clients, and EJBs.

➤ To enable and change RSA/DSA ports:

1. Start the SMC.
2. Select the **Security** icon from the toolbar.
3. Select **Certificates** and choose either the **DSA** tab or the **RSA** tab.
4. In the Port Settings section of the tab, under Port Settings, select any of the check boxes to enable **Runtime**, **Design**, or **Admin** ports for DSA or RSA.
 - Enable RSA only after you have installed an RSA server certificate from a provider such as VeriSign.
 - Enable DSA only if you have installed a DSA certificate on the server.

NOTE After a server certificate has been installed, the ports are automatically enabled when the server is restarted.



For more information, see “Creating and installing server certificates from the command line” on page 240.

5. Change the RSA and DSA port numbers of the **Runtime**, **Design**, or **Admin** ports if necessary.

With a UNIX server, specify port numbers above 1024 if the server is not being run with root access (ports below 1024 are reserved for root access).



For information about cipher suites, see “Restricting SSL cipher suites” on page 253.

6. Select **Update**.

7. To activate the changes, click **Restart** (server).

Once ports are enabled, SilverStream Java clients will verify server certificates when establishing an SSL connection to the SilverStream server.



For more information, see “Verifying SSL server certificates for Java clients” on page 264.

Turning off HTTP communications

You can turn off HTTP communications and allow client communications using only HTTPS or RMI.



For more information, see “Specifying ORB settings” on page 116.

If you accidentally disable ports that prevent you from running the SMC (for example, if you disable the all the administration ports), you will need to edit the `httpd.props` file to re-enable the administration port. You can enable and disable the design and runtime ports using the SMC.

➤ To disable HTTP communications:

1. Start the SMC.
2. Select the **Configuration** icon from the toolbar.
3. Select **General**.
4. In the HTTP Ports section, deselect the **Enable HTTP Runtime Port** (or the HTTP Design or Admin ports) check box to disable.

Because the SilverStream server supports runtime ports for HTTP, HTTPS-RSA, and HTTPS-DSA, you must be careful how you disable runtime ports. The server will run even if you disable all runtime ports. Because many application and server objects involved with design or administration require runtime support, runtime operations can be performed on any port. When you disable HTTP runtime ports, the server checks to see if you have DSA or RSA runtime ports enabled and warns if you proceed to disable the HTTP runtime port.

5. Click **Update**.
6. To activate the change, click the **Restart** button. For more information, see “Restarting the SilverStream server” on page 100.

Now the server will not listen on the HTTP server port.

Restricting SSL cipher suites

When an SSL connection is initialized, the browser client and the server determine a common cipher value to be used for key exchange and encryption. Various cipher values offer different types of encryption algorithms and levels of security. The SilverStream server has a full set of cipher suites that can service a range of clients by providing low, medium, and high-level encryption.

You can restrict the levels of encryption (cipher values) used by the server when communicating in HTTPS through the RSA and DSA runtime ports. This allows you to have a server capable of high-level encryption while preventing connections from lower-level security clients.

During startup, the server reads the list of allowed cipher suites. By default, all cipher suites are allowed. You can use the SMC to deselect specific cipher suites. Only the selected cipher suites will be used to initialize the appropriate SSL socket(s).

➤ To specify which cipher suites are allowed:

1. Start the SMC.
2. Select the **Security** icon from the toolbar.
3. Select **Certificates**.
4. Choose the **RSA** or **DSA** tab.
5. Select the **Enable Runtime Port** check box to enable secure DSA (Java-client) and/or RSA (HTML and Java-client) communications.
6. Specify which cipher suites are allowed by clicking **Cipher Suites** and selecting and deselecting the cipher suites in the resulting panels, as described in the table below. If you select a cipher suite, its description displays in the panel.
7. Click **OK** to exit the Cipher Suites dialog and accept your changes.
8. Click **Update**.
9. To activate the changes, click **Restart** (server).

Table of cipher suites The following table lists the cipher suites (and levels of security) supported by the SilverStream server when using secure communications (HTTPS).

SSL cipher suite	Encryption and security level
RSA cipher suites	
SSL_RSA_WITH_NULL_MD5	0 bit (none)

SSL cipher suite	Encryption and security level
SSL_RSA_WITH_NULL_SHA	0 bit (none)
SSL_RSA_EXPORT_WITH_RC4_40_MD5	40 bit (low)
SSL_RSA_WITH_RC4_128_MD5	128 bit (high)
SSL_RSA_WITH_RC4_128_SHA	128 bit (high)
SSL_RSA_EXPORT_WITH_DES_40_CBC_SHA	40 bit (low)
SSL_RSA_WITH_DES_CBC_SHA	56 bit (medium)
SSL_RSA_WITH_3DES_EDE_CBC_SHA	168 bit (high)
DSA cipher suites	
SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA	40 bit (low)
SSL_DHE_DSS_WITH_DES_CBC_SHA	56 bit (medium)
SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA	168 bit (high)
SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA*	40 bit (low)
SSL_DHE_RSA_WITH_DES_CBC_SHA*	56 bit (medium)
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA*	192 bit (high)
SSL_DH_anon_EXPORT_WITH_RC4_40_MD5	40 bit (low)
SSL_DH_anon_WITH_RC4_128_MD5	128 bit (high)
SSL_DH_anon_WITH_DES_40_CBC_SHA	40 bit (low)
SSL_DH_anon_EXPORT_WITH_DES_CBC_SHA	56 bit (medium)
SSL_DH_anon_WITH_3DES_EDE_CBC_SHA	168 bit (high)

*These cipher suites require both an RSA and a DSA certificate.

Example To allow only greater than 40-bit encryption with RSA communications, select only the following cipher suites under **RSA Cipher Suites**:

- SSL_RSA_WITH_RC4_128_MD5
- SSL_RSA_WITH_RC4_128_SHA
- SSL_RSA_WITH_DES_CBC_SHA
- SSL_RSA_WITH_3DES_EDE_CBC_SHA

Managing Certificate Authorities

The SilverStream server maintains a list of Certificate Authorities (CAs) for verifying client certificates. This represents the list of guarantors the server will trust. There are three common CAs installed when the server is initially configured. These CAs are from VeriSign, Inc. and represent different levels of trust. Class 1 represents a certificate that has minimal trust. Class 3 represents the highest level of trust.

When a user with a client certificate attempts to access the server, the server first checks the list of CAs to verify that the certificate has been approved by a known party, then checks for a valid timestamp to verify that the certificate has not expired. After completing the verification, the server handles the connection request according to the Client Certificate Level parameter (which you can set using the SMC). This setting is described in “Installing and managing client certificates” on page 256.

SilverStream also lets the server extract the CA from an unrecognized client certificate. For more information, see “Installing and managing client certificates” on page 256.

➤ To install or delete a CA certificate:

1. Start the SMC.
2. Select the **Security** icon from the toolbar.
3. Select **Certificates**.
4. Select the **Authorities** tab.

This tab contains information about each CA and allows you to add new CAs to allow a broader range of clients to be trusted or to delete CAs to tighten the level of security on the server.

5. To add a CA, click **Add Certificate Authority** and select the file. To delete a CA, highlight it and click **Delete Certificate Authority**.

Installing and managing client certificates

In the HTTPS environment, a client certificate establishes the identity of a user when communicating with a server. Client certificates can be obtained from various sources, but to be useful the certificate must be signed by a guarantor (Certificate Authority or CA) that is trusted by the server.

The SilverStream server supports client certificates from standard Internet browsers (including Netscape and Internet Explorer) that use RSA encryption, and supports X.509 certificates, which is a particular implementation of the Certificate interface used by many certificate issuers.

Client certificates and EJBs Client certificates that use RSA encryption are also used to establish the identity of a user in the jBroker IIOP over SSL environment (used by EJBs). As with the HTTPS environment, the server supports client certificates from standard Internet browsers. If you need to add a new CA to the server, see “To install or delete a CA certificate:” on page 255.

Enabling and installing client certificates

Using the SMC, you can determine how the server will handle connection attempts from users with valid client certificates. Seven parameter options are available, each representing a different level of restriction. Two of the options will automatically install new certificates that are verified by the server and add them to the database as new users in the Certificate Security realm. You can also install the certificates manually (see “Manually installing client certificates” on page 260).

Each set of HTTPS ports (HTTPS-RSA HTTPS-DSA) has a single set of cipher suites associated with it. The cipher suite you select applies to all ports (runtime, design-time, and administration) of that type.

NOTE The SMC will not allow HTTPS ports to be enabled without a valid certificate. If you try to enable an HTTPS port using the SMC or by editing the props file without first installing a certificate, the server error will display on startup. See “The httpd.props File” on page 437.

➤ **To enable client certificates on the server:**

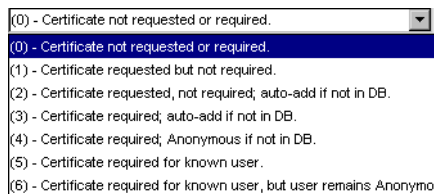
1. Start the SMC.
2. Select the **Security** icon from the toolbar
3. Select **Certificates**.

Each set of HTTPS ports (HTTPS-RSA HTTPS-DSA) has a single set of cipher suites associated with it. Although the cipher suite you select applies to all ports (runtime, design-time, and administration) within a protocol type, you can set each port to a different value.

NOTE You cannot enable an HTTPS port without first installing an HTTPS-RSA or HTTPS-DSA server certificate. If you try to enable an HTTPS port using the SMC or by editing the props file without first installing a certificate, the server error will display on startup.

The screenshot shows the 'Certificates' tab in the SMC configuration interface. The window has a title bar with tabs: General, Advanced, Permissions, Users & Groups, Certificates, and Security Providers. The 'Certificates' tab is active. The main content area is titled 'HTTPS Client certificate level:' and contains a dropdown menu with the selected option '(0) - Certificate not requested or required.'. Below this is the 'Accelerator Settings:' section, which includes a checkbox for 'Use Hardware Accelerator' (unchecked), a 'Vendors:' dropdown menu with 'ACS smart card' selected, and input fields for 'Slot number:' (containing '0') and 'PIN:'. The 'Trusted Clients:' section is titled 'List of clients to be trusted:' and features a large empty text area on the left and three buttons ('Add...', 'Edit...', 'Delete') on the right. A blue bar at the bottom right of the window contains the 'Update' button.

- Select an option from the dropdown list labeled **HTTPS Client certificate level**, then click **Update**.



You need to restart the server **only if** you changed from level 0 to another level.

Your selection determines how the server will handle all valid client certificates. The certificate verification table (below) describes each option. The options are numbered from 0 (no verification) to 6 (most restrictive level).

Certificate verification table

Certificate verification level	Description
Certificate not requested or required (0)	<p>Turns off the use of certificates for establishing a client's identity.</p> <p>A client that connects via HTTPS will still benefit from secure communications and can check the identity of the server, but will not be asked to present its own certificate.</p>
Certificate requested but not required (1)	<p>The server will request a certificate from the client. If the client does not have a certificate or has a certificate that has not been validated by the server, a connection is allowed but the user remains anonymous.</p> <p>If a certificate is presented for an established user, the client takes on the identity of the Certificate Security user.</p>

Certificate verification level	Description
Certificate requested, not required; auto-add if not in database (2)	<p>The server will request a certificate from the client. If the client does not have a certificate, a connection is allowed but the user will remain anonymous. If a legitimate certificate is presented but does not yet correspond to a client validated by the server, the server will automatically add the certificate, and the client takes on the identity of the newly created Certificate Security user.</p> <p>If a certificate is presented for an established user, the client takes on the identity of the previously established Certificate Security user.</p>
Certificate required; auto-add if not in database (3)	<p>The server will request a certificate from the client. If the client does not have a certificate, the connection is denied. If a legitimate certificate is presented but does not yet correspond to a client validated by the server, the server will automatically add the certificate, and the client takes on the identity of the newly created Certificate Security user.</p> <p>If a certificate is presented for an established user, the client takes on the identity of the previously established Certificate Security user.</p>
Certificate required; Anonymous if not in database (4)	<p>The server will request a certificate from the client. If the client does not have a certificate, the connection is denied. If a legitimate certificate is presented but does not yet correspond to a client validated by the server, the client is allowed to connect but remains Anonymous.</p> <p>If a certificate is presented for an established user, the client takes on the identity of the previously established Certificate Security user.</p>

Certificate verification level	Description
Certificate required for known user (5)	<p>The server will request a certificate from the client. If the client does not have a certificate or has a certificate that does not correspond to a client previously validated by the server, the connection is denied.</p> <p>If a certificate is presented for an established user, the client takes on the identity of the previously established Certificate Security user.</p>
Certificate required for known user, but remain Anonymous (6)	<p>The server will request a certificate from the client. If the client does not have a certificate or has a certificate that does not correspond to a client previously validated by the server, the connection is denied.</p> <p>If a certificate is presented for an established user, the client is allowed to connect but still remains Anonymous.</p>

Manually installing client certificates

The previous section describes parameter options that automatically add new validated certificates to the database on the server. You may prefer to install the certificates separately. This section describes how to manually add client certificates to the database. It consists of two procedures, one for the client machine and the other for the server.

➤ To install a client certificate—client machine:

NOTE For this procedure to work, the certificate validation level on the server must be set to 1, 2, or 4 (see “Certificate verification table” on page 258).

1. Make sure you have a valid client certificate installed on your browser.
2. Open your browser and go to the following URL:


`https://server/SilverStream/Meta/Certificates?action=data`

where *server* is the name of your server.

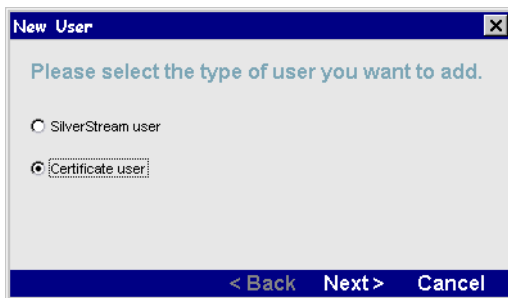
The server will extract the user information from the certificate and send it back to the client.

3. Save the file that the server presents to an appropriate area.

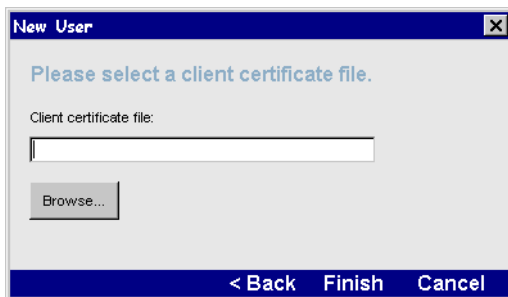
➤ **To install a client certificate—server machine:**

1. Start the SMC.
2. Select the **Security** icon from the toolbar.
3. Select **Users & Groups**.
4. Select **Certificate Security**.
5. Choose the **New User**  icon at the bottom of the right pane.

The following panel displays.



6. Choose **Certificate user**.



7. Enter the file name obtained by the client from the previous procedure (“To install a client certificate—client machine:” on page 260) and click **Finish**.

This adds the certificate as a new user to the database on the server.

Extracting a CA from a client certificate

The SilverStream server also allows you to extract the CA from a client certificate if the CA is not installed on the server. This section consists of two procedures, one for the client machine and the other for the server.

➤ **To extract the CA from a certificate—client machine:**

NOTE For this procedure to work, the certificate validation level on the server must be set to 1 or 2 (see “Certificate verification table” on page 258.)

1. Make sure you have a valid client certificate installed on your browser.
2. Open your browser and go to the following URL:

`https://server/SilverStream/Meta/Certificates?action=dataCA`

where *server* is the name of your server.

The server will extract the CA from the certificate and send it back to the client.

3. Save the file that the server presents to an appropriate area.

➤ **To extract a CA from a client certificate—server:**

1. Start the SMC.
2. Select the **Security** icon from the toolbar.
3. Select the **Certificates** panel.
4. Select the **Authorities** tab.
5. Click **Add Certificate Authority**.
6. Browse for the location of the extracted CA, then click **Open**.
The program installs the new CA to the database.
7. To activate the new CA, restart the server by clicking the **Restart** (server) icon.

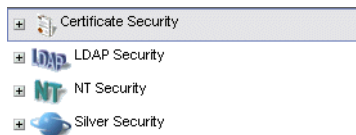
Accessing certificate users

Certificate users are added to a security realm called Certificate Security. This realm is included in the lists of users and groups supported by the SilverStream server.


➤ **To access certificate users:**

1. Start the SMC.
2. Select the **Security** icon from the toolbar.
3. Select **Users & Groups**.

4. Navigate to a user from the **Certificate Security** list.



5. Use this panel to view current users or add users to groups.

 For more information about users and groups, see “Setting Up Users and Groups” on page 127.

You can use certificate users in security expressions exactly as you use other users in SilverStream. For more information about security expressions, see “Authorization and access control” on page 271.

Updating client certificates

An X.509 certificate includes a start date and an end date. A client is not allowed to connect to the server with a certificate that has expired. You can update the certificate representing an established Certificate Security user by obtaining a new certificate and assigning it to a user already known to the SilverStream server. This ensures that any security expressions involving the existing user will continue to work properly.

Normally, an existing certificate user would present the updated certificate to the server using the URL described in “Manually installing client certificates” on page 260.

NOTE If you are running the server with one of the autoinstall parameters described previously (level 2 or 3) and an existing client certificate holder attempts to access the server with a new (updated) certificate that the server validates, that client will be installed as a new user. This client may be restricted from resources available to the previous user.

➤ To update an existing certificate user:

1. Select the **Security** icon from the toolbar.
2. Select the **Users & Groups** panel.
3. From the Certificate Security domain, navigate to the user you want to update.

4. Click the Properties button:



A form with three tabs displays. The General tab shows general information concerning the user. The Additional Attributes tab displays information about the certificate. The Update tab allows you to update the certificate for the selected user.

5. Select the **Update** tab.
6. Enter the file name for the updated certificate, then click **Update**.

This replaces the old certificate with the new version without changing the identity of the user on the server.

Verifying SSL server certificates for Java clients

SilverStream Java clients verify server certificates against a list of trusted CA certificates (that are stored in the `agrootca.jar` file) when establishing an SSL connection to the SilverStream server. If a certificate cannot be verified, the SSL connection cannot be established.

To use SSL communications, you must have one of the following server certificates installed on your SilverStream server:

- An RSA certificate purchased from an external company that offers certificate services (such as VeriSign).

You use SilverStream's SMC or command-line tool (`AgDigitalIDStep1`) to generate an RSA Certificate Signing Request (CSR). The encrypted CSR file is submitted to an external CA, which identifies the organization running the SilverStream server and creates a signed certificate. If you use an external CA and RSA encryption, see "Using the `agrootca.jar` file to verify RSA certificates" on page 265.

- A self-generated RSA certificate (you act as your own CA using your internal organization's tool, such as Netscape Certificate Server).

If you generate your own certificate, you need to distribute the `agrootca.jar` file as described in "Using the `agrootca.jar` file to verify RSA certificates" on page 265.

- A self-signed DSA certificate created using the SMC.

Organizations using a self-signing DSA server certificate should use the command-line option, as described in "Command-line option for self-signing DSA server certificates" on page 265.

Using the agrootca.jar file to verify RSA certificates

SilverStream server clients verify the SilverStream server certificate against a list of trusted CA certificates that are stored in the **agrootca.jar** file. At startup, the SilverStream client reads in the list of CAs and then checks all server certificates against contents of agrootca.jar file to verify that certificates are signed.

You should include all trusted root CAs in the agrootca.jar file and remove any untrusted CAs. SilverStream server clients will only trust server certificates that were signed by the root CA certificates in the JAR file.

If you generate your own RSA certificates using a tool such as Netscape Certificate Server, you should put the CA certificate of the Certificate Server into the agrootca.jar file and then distribute the JAR file to each client machine running SilverStream Designer, SilverJRunner, and SilverJ2EEClient. The agrootca.jar file should be stored in the **\lib** directory of the client's SilverStream installation directory.

NOTE The agrootca.jar file is used by the SilverStream server only when user code tries to connect to another server using SSL (such as in the case of a servlet).

Simplified secure port configuration You can use the SilverStream server RSA port to provide secure communications between the SilverStream server and all Java clients, HTML clients, and EJBs. To simplify your SSL communications configuration, you may want to use RSA certificates only.

Command-line option for self-signing DSA server certificates

If your organization uses self-signed DSA certificates to provide data encryption between the SilverStream server and clients, you should use the **+Dsssw.ssl.nocacheck** command-line option when you run SilverStream Designer, SilverJRunner, and SilverJ2EEClient. This option prevents the SilverStream client from attempting to verify a DSA server certificate. For example:

```
SilverDesigner +Dsssw.ssl.nocacheck
SilverJRunner +Dsssw.ssl.nocacheck server database formorview
SilverJ2EEClient +Dsssw.ssl.nocacheck server database warfile
```

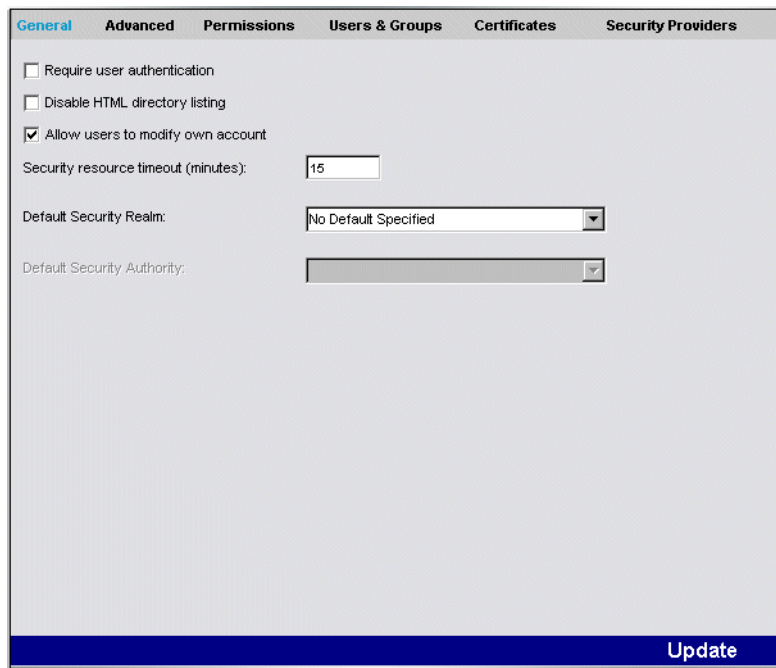
Enabling authentication

You can require user authentication and activate related settings. You set authentication settings when you are ready to deploy applications that require authentication for both HTTP user/password protection and HTTPS-RSA client certificates.

 For more information, see “Setting up separate ports” on page 106.

➤ **To enable user authentication:**

1. Start the SMC.
2. Select the **Security** icon from the toolbar.
3. Select **General**.





The screenshot shows a window titled "General" with tabs for "General", "Advanced", "Permissions", "Users & Groups", "Certificates", and "Security Providers". The "General" tab is active. It contains the following settings:

- Require user authentication
- Disable HTML directory listing
- Allow users to modify own account
- Security resource timeout (minutes):
- Default Security Realm:
- Default Security Authority:

An "Update" button is located at the bottom right of the window.

4. Activate the security options as follows.

Item	Description
Require user authentication	<p>Requires users when first accessing the server to authenticate themselves, either through a certificate or user name/password. Once authenticated, users do not have to be authenticated again during the session.</p> <p>Setting this option disallows Anonymous users; if you don't set this option, a user can access the server without logging in or sending a certificate, in which case the user is known as Anonymous.</p> <p>If you are using this option with HTTPS, activate Client certificate level in HTTPS.</p> <p>You can also require login on a per-object basis. See "Changing access" on page 276.</p>
Disable HTML directory listing	<p>Disables the listing of server directory contents in a browser when the browser points to the directory's URL. Prevents users from seeing directory contents. If this option is turned on, the server returns a FORBIDDEN error.</p>
Allow users to modify own account	<p>By default, users can change their own user properties. You can turn off this privilege by deselecting this check box. If this privilege is turned off, only administrators (users with Read Server Configuration and Modify Server Configuration permissions) can change user properties.</p> <p> For more information, see "Editing user properties" on page 131.</p>
Security resource timeout	<p>Specifies how frequently the SilverStream server will upload current lists of users and groups from the NT, LDAP, and/or NIS+ servers, which would include any updates to these lists.</p> <p> For more information, see "Resetting the security resource timeout" on page 210.</p>

5. For changes to take effect, click **Update**.

Using Cryptographic Hardware Integration


Cryptographic Hardware Integration (CHI) enables significant application server SSL encryption/decryption performance enhancements on machines with:

- SilverStream eXtend Application Server (Version 4)
- A supported hardware accelerator card

 For information about the supported cards, see the server's *Release Notes*.

To use CHI, you need to:

- Install a supported hardware accelerator card on the machine where the SilverStream server is installed

 For information about installing and setting up the card, see your hardware card's documentation.

- Install CHI on the machine (see “To install CHI:” on page 268)
- Configure the server using the SMC (see “To configure the SilverStream server to use the hardware accelerator card:” on page 269)

➤ To install CHI:

1. Invoke **chiVersionInstall.exe** (the CHI installer).
2. Follow the prompts to install CHI into your SilverStream server's installation directory.

You can later uninstall CHI separately if you want.

➤ To configure the SilverStream server to use the hardware accelerator card:

1. Start the **SMC**.
2. Select the **Security** icon from the toolbar.
3. Select **Advanced**.
4. Select **Use Hardware Accelerator**.
5. Specify your accelerator card.
6. Specify the slot number and PIN, which you can get or set up from the utility software that came with the card.
7. Click **Update**.
8. Restart the server.

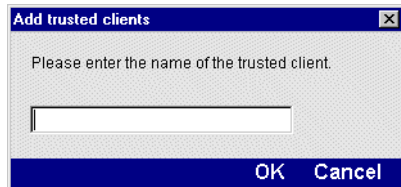
Managing trusted clients

You can use the SMC to set up a list of clients that are trusted by a SilverStream server when receiving an asserted identity in EJB calls. This is a cluster-level property—any trusted clients set up on a single server are propagated to all servers in the cluster.

➤ **To add a client to the list of trusted clients:**

1. Start the **SMC**.
2. Select the **Security** icon from the toolbar.
3. Select **Advanced**.
4. In the Trusted Clients section of the Advanced tab, choose **Add**.

You are prompted for the client name.



5. Enter the client's host name.

The entry can contain the asterisk wild card character(*), for example:

`*.mydomain.com`

or

`server*.mydomain.com`

or

`*`

If you use the asterisk, it should be the last character in a section of the URL. Any characters following the asterisk in a section are ignored. In the example shown above, if `server*` was shown as `server*1`, the 1 would not be included in the list.

10 Using Security

This chapter describes ways to provide and restrict access to application databases, a server, and/or an entire cluster. This chapter contains sections on:

- Authorization and access control
- Permission types
- Making secure application objects executable
- Default server and object security
- Locking down servers, clusters, and applications
- Securing the production server
- Security checklist
- Securing the development server
- Excluding robots

Authorization and access control

Access control specifies what operations users are authorized to perform on different types of application objects and resources. Authorization occurs after users authenticate themselves while using the application.

Authorization and access control

SilverStream authorization and access control security settings work together to let you manage access to the server, application data, and application objects. Authorization uses security expressions to determine what an identified user is allowed to access.

You can modify access settings while the application is running by selecting individual resources on the server (tables, forms, views, and so on) and specifying who is allowed to access them.

SilverStream supports the Java security Access Control List (ACL) API, allowing developers to programmatically control access to any SilverStream object, including data and application objects. This Java security works with the security provided by the SMC and the SilverStream eXtend Application Server Administration API.

 For more information about the SilverStream API, see Chapter 13, “Using the Server Administration API”.

You can use the SMC to control SilverStream server administration at the server or cluster level and to control access to the following:

- A SilverStream database
- A deployed J2EE archive
- The design of individual SilverStream database tables
- Rows within a table
- Directories, pages, forms, and views
- Media objects
- Business objects

You specify access by setting the permissions described in “Permission types” on page 272.

Permission types

SilverStream allows you to set up access control on various **administration operations** and **database objects** through permissions.

Administrative server permissions

The **administration resource** controls your ability to view, modify, and change administrative settings (and permissions to access settings). Once you have secured the administration resource, unauthorized users will not be able to perform administrative operations.

NOTE Only members of the Administrators group can access administration settings. To prevent unauthorized users from accessing all administrative information such as who is logged in, session information, permission settings for users and groups, number of connections, and so on, it is important to retain this restriction.

Each of the permission tabs listed in the following table represents an aspect of the administration resource that should remain restricted.

Permission	Description
Read Server Configuration	Limits who can view administration settings and connection parameters.
Modify Server Configuration	Limits who can administer the server or cluster by changing connection parameters such as who is logged in, permission settings for users and groups, number of connections, and so on.
Read Directory Listing	Limits who can browse the server directories. Does not affect whether a user can access an object.
Read Users and Groups	Limits who can see users and groups that have been defined on the server. Users must have this permission in order to set permissions; otherwise, they cannot see the users and groups.
Set Permissions	Limits who can change all of the permissions that control the administration resource. Anyone with Locksmith privilege will bypass the check for this permission and be able to change the security properties on any object.

Database object permissions

Secure your application databases by setting permissions on directories and objects such as pages, views, business objects, and J2EE archives. Decide whether to restrict access by user or group, using simple lists or advanced expressions. Permission settings apply to the directory, all subdirectories, and objects in the directory or any subdirectory. Each object or resource within a hierarchy must be secured.

 For more information, see “Step 10: Secure your application databases” on page 298 in the security checklist.

The following permissions apply to database objects (such as tables, forms, views, and so on). The permission names on the tabs vary slightly depending on which object is selected.

Permission	Description
<p>Read (Called Read Design for tables)</p>	<p>For an object, limits who can view an object's design information. For a database or directory, limits who can access child objects.</p> <p>To easily prevent users from accessing objects in a database or directory, deny them Read access at the database or directory level.</p> <p>NOTE Users who need to run an object in a subdirectory must have Read access to all of its parent directories. For more information, see "Making secure application objects executable" on page 285.</p>
<p>Modify (Called Modify Design for tables) (Called Write for media objects such as images and sounds)</p>	<p>For an object, limits who can change the object. For a database or directory, limits who can add new child objects.</p> <p>To avoid adverse changes to your server, carefully review who has Modify permission to a directory.</p> <p>With a classic application, for example, a user with Modify access to the Pages directory can create new page objects. For classic applications, you should only assign Modify access for directories to developers and/or administrators.</p> <p>For a J2EE application, a user with Modify permissions can change the contents of the deployed archive (the deployment plan, for example). You probably want to restrict this type of access.</p>
<p>Execute (Called Default Execute for databases and directories)</p>	<p>Limits who can run application objects (such as pages, forms, views, J2EE archives, and so on) within the selected database or directory. End users typically have Execute permission on objects for both classic and J2EE applications.</p>
<p>Set Permissions</p>	<p>Limits who can alter access control information on an object. For example, you may decide that only administrators should be able to change the security properties on this object.</p>

Permission	Description
Select (on tables only)	<p>Lets the user set up row-level retrieval access on individual tables by creating selection criteria through the Expression Builder. This is useful if you want to set up row-level security in a table.</p> <p>This is for classic applications only.</p> <p>To restrict Write access to the table's data, you need to write a business object. For more information, see the chapter on table-modified business objects in the <i>Programmer's Guide</i> of the server's Classic Development Help.</p>

How access works

Different types of resources (servers, tables, objects, images, and so on) have different sets of permissions that correspond to their functions.

- By default, access to **server administration operations and database objects** is restricted to members of the SilverStream Administrators group. See “Default server and object security” on page 288.
- Most **directory levels** in the hierarchy allow you to set Read, Write, Execute, and Set Permissions access rights. Each object or resource within a hierarchy can be secured in an identical fashion.
- If access rights are defined at the **database level**, you see all databases but can only expand or open the ones for which you have Read access.
- If you have **Read access to a particular directory**, you can see the named contents of that directory.
- Requests for the contents of the named **object** may be restricted based on expressions set on that object.
- You can change **security access rights** only if you have Set Permissions rights to the SilverStream database directory or object. You can then use specific access expressions to further limit modifications.
- You can **secure multiple directories** by selecting the Apply To This Directory And All Descendants radio button at the bottom of the permission tabs. You can enable the Require Login For Access check box for selected objects as needed.

To **restrict Write access to rows** on INSERT, UPDATE, or DELETE, you can create business objects that enforce access restrictions. For more information, see the chapter on table-modified business objects in the *Programmer's Guide* of the SilverStream server's Classic Development Help.



For more information, see “Making secure application objects executable” on page 285.

Changing access

You can change object security at the server or cluster level, at the directory level, or at the object level by setting permissions in the SMC.

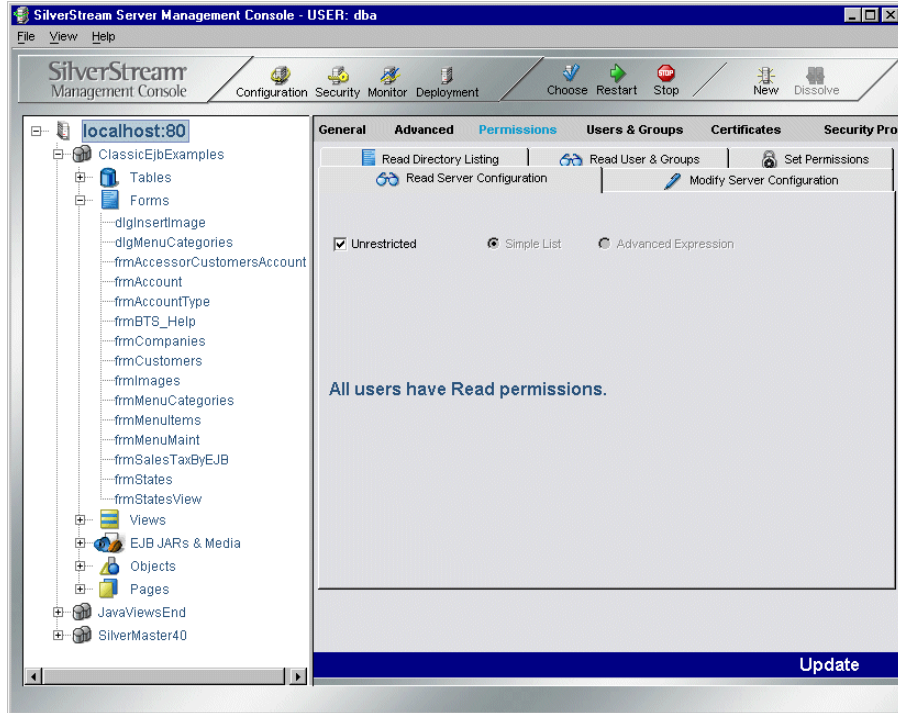
NOTE In addition to setting permissions in the SMC, you can also set permissions at the command line SilverCmd SetSecurity described in the SilverCmd Reference in the *Facilities Guide*.

➤ To change user access:

1. Start the SMC.
2. Select **View>Display Classic Settings**.
3. Select the **Security** icon from the toolbar.

4. Select the **Permissions** panel.

The SMC view changes: all the clusters, servers, and databases being administered by the SMC are now expandable, allowing you to list their contents in order to set permissions at any level you want.



5. Select an object or directory and set your permissions as described in “Permission types” on page 272.

Setting permissions on multiple objects You can set permissions on more than one object at a time as long as all the objects are in one directory (that is, the objects are all the same type, such as forms or pages, and on the same server): select Shift+Click for contiguous entries or Ctrl+Click for noncontiguous entries and specify the permissions for all the selected objects.

Setting the scope of the permissions and whether login is required The radio buttons that appear at the bottom of the Permissions panel depend on whether you selected a directory or an object. The following is a description of each button option that might appear.

Radio button option	Description
Apply to this directory only	Displays for directory selections only. Applies permissions only to the selected directory and becomes the default setting for new objects in the directory structure. Does not change permissions for existing objects in the directory.
Apply to this directory and all descendants	Displays for directory selections only. Applies permissions to the selected directory and all existing objects within that directory (including subdirectories) as well as new objects. Overrides any existing settings for existing objects.
Require login for access	Displays for object selections only. Requires the user accessing the object to be authenticated, either by a client certificate or by being logged on. Sets authentication on a per-object basis. You can also require authentication at the server level. See “Enabling authentication” on page 266.

Restricting permissions

Each tab on the form represents a permission type that applies to the selected directory or object. To restrict access, select a tab, deselect **Unrestricted**, then select either **Simple List** or **Advanced Expression**.

CAUTION *If you deselect the Unrestricted check box, be sure to modify either the Simple List or Advanced Expression; otherwise, no one will have access. This problem will also occur if you clear all previous Simple List entries. If you find yourself in the situation where no one has Read access to the server, you can run SilverMasterInit using the -a command-line option. For more information, see “Using the SilverMasterInit program” on page 416.*

Using simple lists

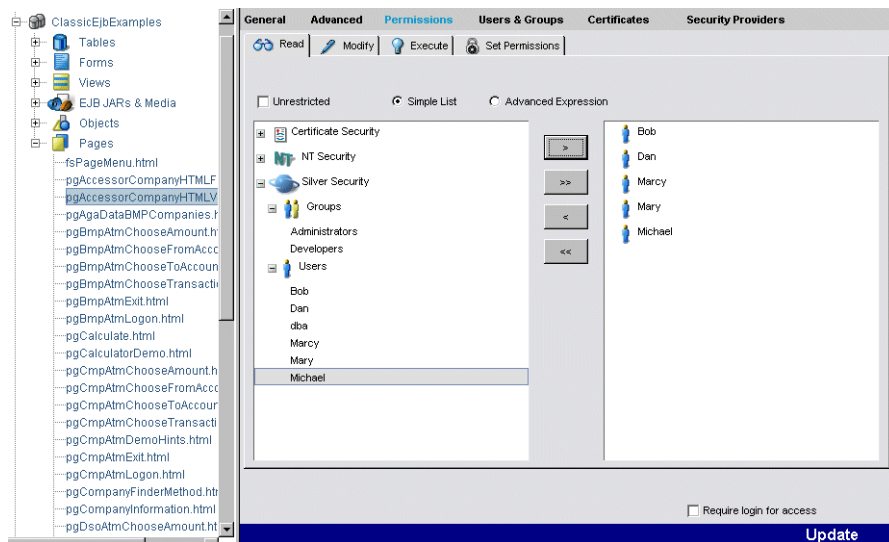
This option allows you to specify users and groups that are known to a security provider and have access permission as defined in the provider setup.

➤ To use simple lists:

1. In the SMC, make sure that **View>Display Classic Settings** is checked.
2. Select the **Security** icon from the toolbar, then select the **Permissions** panel.
3. Select the objects or directory you want to set permissions for on the left side of the SMC.
4. In the Permissions panel, select the tab for the permission type you want to restrict.
5. To activate the Simple List button, turn off **Unrestricted** (if selected).
6. Simple List is the default selection. From the form, select the users and groups to whom you want to grant permission, then click >. To select all users or groups, click >>. Selected users and groups are listed in the list box on the right.

NOTE A local group that contains a global group (that you defined with the NT User Manager) will appear in the SMC with its individual member names listed (instead of the group name). For more information, see “Using NT security” on page 211.

To remove the permission from a user or group, select the user or group and click <. To remove all users and groups, select <<.



Using advanced expressions

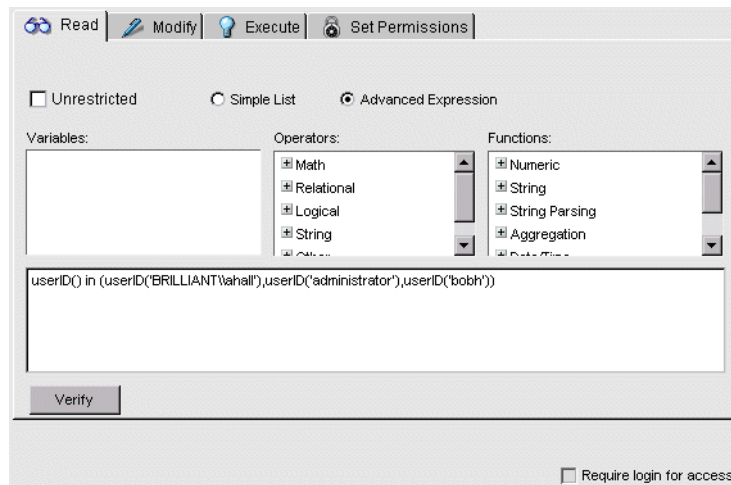
Use this option to build expressions that let you specify access according to particular criteria. To use advanced expressions, select **Advanced Expression** instead of Simple List (see “To use simple lists:” on page 279). Expressions can include any of the following:

- Identity/group membership
- Built-in functions, which can perform logical and relational operations
- Database content, for row-based access control

➤ To use advanced expressions:

1. Select the objects or directory you want to set permissions for on the left side of the SMC.
2. In the Permissions panel, select the tab for the permission type you want to restrict.
3. If selected, turn off **Unrestricted**; then select **Advanced Expression**.

The SilverStream Expression Builder displays selection panes for variables, functions, and operators.



Examples of advanced expressions The following are examples of advanced security expressions.

- Use the day() method to disallow access to the object on weekends (the day() function returns a number from 0 to 6, where 0 indicates Sunday, 1 indicates Monday, and so on):

```
day(now()) >= 1 and day(now()) <= 5
```


- Similarly, the following expression disallows access to an object except between 9:00 and 5:00 Monday through Friday:

```
(day(now()) >= 1) and
(day(now()) <= 5) and
(hour(now()) >= 9) and
(hour(now()) <= 17)
```



For more information about specifying expressions, see the chapter on expressions in the *Programmer's Guide* of the SilverStream server's Classic Development Help.

UUID support

The Expression Builder supports **Universally Unique Identifiers** (UUIDs) for security expressions. These provide a more secure system than simple integer IDs. UUIDs are used by default in simple security expressions and are available in advanced expressions.

You can enter UUID expressions directly in the Expression Builder, or choose them from the ID section in the Expression Builder's Functions panel.

Function	Description
userID()	Finds the UUID of the user currently logged on.
userID('name')	<p>Finds the UUID of a specific user.</p> <p>To create a qualified name, start with the security realm (SSSW, NT, LDAP, or NIS+), add security authority (such as the NT domain name or LDAP/NIS+ server name), then add the user name. Separate each component with two backward slashes (\). (You need to escape backslashes in user names by doubling the backslashes.)</p> <ul style="list-style-type: none"> • By default, if both the realm and authority are missing, a SilverStream user is assumed. • By default, if the realm is missing, an NT user is assumed, with the first component assumed to be the NT domain. <p>For example:</p> <ul style="list-style-type: none"> • userID('bobh') refers to the SilverStream user bobh (user defined in Silver Security) • userID('DEVA\\JWilkins') refers to the NT user JWilkins, who is in the NT domain DEVA.

Function	Description
<code>groupID('name')</code>	<p>Finds the UUID of a specific group.</p> <p>Group names, like user names, consist of a qualified name.</p> <p>For example:</p> <ul style="list-style-type: none"> • groupID('Administrators') refers to the Silver Security Administrators group • groupID('DEVA\\Accounting') refers to the NT Accounting group in the DEVA domain
<code>UUID('uuidString')</code>	<p>Finds the UUID corresponding to the string representation of a UUID. This form is used when a group ID or user ID cannot be resolved.</p> <p>SilverStream translates names into corresponding UUIDs before storing them in the AgAccessRights system table. Similarly, when the client views the expression, the internal form is translated back into human-readable names. If an internal UUID cannot be translated (for example, if it has been deleted), <code>UUID()</code> is used.</p>

Examples The following are examples of security expressions using UUID.

- This example restricts access to the Silver Security users **administrator** and **bobh**:

```
userID() in (userID('administrator'),userID('bobh'))
```
- This example restricts access to the Silver Security user **bobh** and the NT user **JWilkins**, who is in the NT domain DEVA:

```
userID() in (userID('bobh'),userID('DEVA\\JWilkins'))
```
- This example restricts access to the SilverStream user **administrator** and any user in the SilverStream Developers group:

```
userID() in (userID('administrator')) or userID() userin (groupID('Developers'))
```
- This example restricts access to the SilverStream user **administrator** and any user in the NT Accounting group, which is in the DEVA domain:

```
userID() in (userID('administrator')) or userID() userin (groupID('DEVA\\Accounting'))
```



For more information about specifying expressions, see the chapter on expressions in the *Programmer's Guide* of the SilverStream server's Classic Development Help.

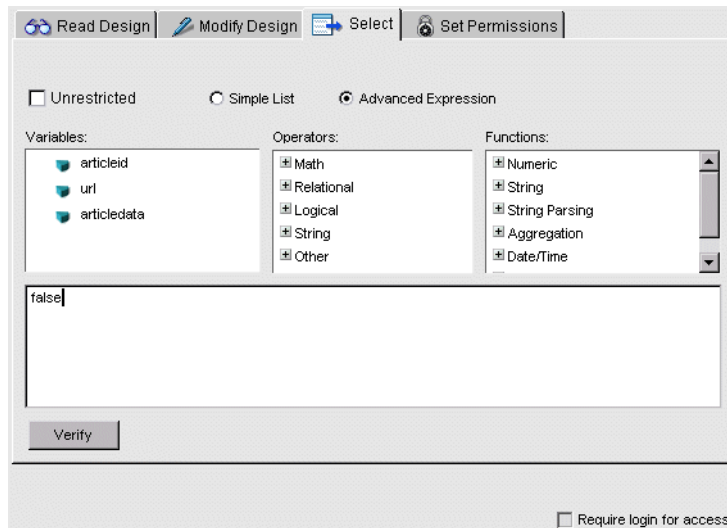
Row-level expressions

This is for classic applications only. You can perform database table and row-based access control through the SELECT expression on a table. Row-level expressions work as follows:

- The expression may involve any number of columns.
- The SilverStream server (or the database) evaluates this expression during a SELECT operation.
- Only rows that pass the SELECT expression are returned.

➤ To use row-level security:

1. Select the table for which you want to set permissions.
2. On the Permissions panel, select the **Select** tab.
3. If selected, turn off **Unrestricted**; then select **Advanced Expression**.



4. Select the column in the left panel and write an expression.

Examples of SELECT expressions The following are examples of row-level SELECT expressions.

- This example uses a group and a table column called SalesCode. The SELECT expression will show orders whose sales codes are between 100 and 200 only to users who are in the Northeast sales group.

```
userID() userin (groupID('NorthEast-Sales')) AND orders.SalesCode
BETWEEN 100 AND 200
```

- This example uses an if...then statement to use a conditional expression. The following SELECT expression will show employees whose status is 3 only to managers.

```
if (userID() userin(groupID('Managers'))
then (employees.status>0)
else (employees.status<>3)
endif
```



For more information about specifying expressions, see the chapter on expressions in the *Programmer's Guide* of the SilverStream server's Classic Development Help.

The INLIST operator

The INLIST operator is a SilverStream construct that can be used in an advanced SELECT security expression. For example, you may have special columns in your tables that contain information about who can view them. The rows are retrieved and then filtered through this expression using the column's values.

Example For example:

- The Project table has a column called Allowed_Groups, which has a type **varchar(100)**.
- The first row has the following value for Allowed_Groups: “**Administrators Developers Designers**”.
- The second row has the following value for Allowed_Groups: “**Administrators**”.
- The third row has the following value for Allowed_Groups: “**Administrators Developers**”.

Suppose a user named Dev1 belonging to the Developers group logs in to the server. The following shows the SELECT security expression on table Project:

```
user() userin (inlist(PROJECT.ALLOWED_GROUPS))
```

Next, Dev1 creates a form against table Project and runs the form. The server retrieves three rows and applies the SELECT expression to every row before giving the form back to the client. Because Dev1 belongs to the Developers group, the server shows only the first and third rows—but not the second row, which is restricted to the Administrators group.

Making secure application objects executable

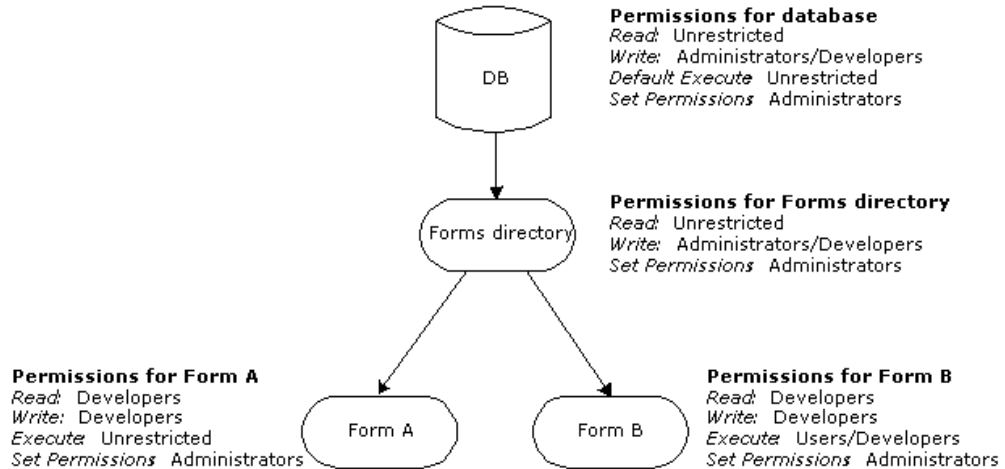
When you are securing application databases, you also need to make sure that users can run database objects such as forms and pages for classic applications, and EARs, WARs, and EJB JARs for J2EE applications.

Typical permissions for classic applications

Typically, administrators and developers have Read and Write permissions at the database and parent directory level, along with Read, Write, and Execute permissions on database objects. Because Read and Write permissions allow users to be able to see and (respectively) change the source code, you would typically assign these permissions only to developers and/or administrators. The Set Permissions access is usually limited to server administrators.

To allow users to run forms, pages, objects, and so on, you usually assign them unrestricted Execute permission at the directory level (including all descendants) and on all objects they need to run. Because users who need to run an object in a subdirectory need Read access to **all** of its parent directories, you must take special care when setting access to objects in a database or directory. For example, if a subdirectory contains a form or page that users need to run, you must grant them unrestricted Read access to both the database and the parent directory. This situation is typically the only time you assign users unrestricted Read access. You may also want to publish to your production server without source to prevent users from viewing source code and design information.

As shown in the following figure, anyone can run Form A, but only members of the Users and Developers groups can run Form B. Because Form A has unrestricted Execute access, Read access must also be set to unrestricted on the Forms directory. If Read access on the Forms directory were to be limited to Developers and Users, only those users would be able to run Form A.



Servlet URLs With servlets and pages, you can create your own directory hierarchies for their URLs. Because the SMC cannot show their directory hierarchies, you cannot use the SMC to set permissions on these types of objects. You can set these permissions in either of the following two ways:

- By using the permission settings inherited from the parent database. If you correctly set permissions on the database before you save the servlet or page and create the URL hierarchy of the servlet or page, the correct permission settings will be inherited.
- By using SilverCmd. Once the servlet or page is created, use SilverCmd to specify the URL path hierarchy in the XML command file.

When setting permissions, you typically start at the top of the directory structure and set broad restrictions that you then refine. You then work your way down through the directory structure and open up object Read access to appropriate users and groups as needed. Enable the **Require login for access** check box for selected objects as needed.

Typical permissions for J2EE application objects

J2EE archives are deployed to the EJB Jars & Media directory of the deployment database, so the deployer needs **Write** permissions to that directory.

To allow users to run the deployed archives, you usually assign them unrestricted Execute permission at the directory level (including all descendants).

➤ **To make secure database objects executable:**

1. Log in to the SMC as an Administrator or user with Locksmith privilege.
 2. Select the **Security** icon from the toolbar.
 3. Select **Permissions**.
 4. Select the server and application database you want to change.
 5. Set the initial group permissions as follows:
 - Assign Read and Modify permissions to developers and administrators.
 - Assign Set Permissions access to administrators.
 - Assign unrestricted Execute permission to end users.
- NOTE** Select the **Apply to this directory and all descendants** check box on all of the preceding tabs.
6. Click **Update**.
 7. Select an object directory (like the form, page, or table for a classic application or the EJB JARS & Media directory for J2EE applications).
 8. Select the **Unrestricted** check box on the **Read** permission tab, but do **not** click **Apply to this directory and all descendants**.
 9. Click **Update** to save these settings.
 10. Apply the same **Read** permission settings (as in Step 8) to all major directories of your application that you want users to access. While these settings give users access at the parent directory level, they also let you restrict individual objects (such as pages, forms, tables, and EJB JARS & Media) within those directories.
 11. Review individual objects to determine if the **Require login for access** option should be enabled. (The **Require login for access** option has no effect for J2EE applications.)

Default server and object security

If you choose the installation default, the SilverStream server initializes the SilverMaster database. When access to SilverMaster is restricted, all users (except those in the Administrators group) are unable to access administration operations, add databases, and browse directory listings. If your SilverStream server is running in a restricted production environment, all users will be required to authenticate themselves when accessing the server.

If you did not choose the **Restrict Access to the SilverStream Server** installation option, your server resources are not locked down. If you are developing SilverStream applications, you may want to run the server in an unrestricted design environment until you are ready to deploy the application. Installing the SilverStream server with **unrestricted** access means that unauthorized users can perform administrative operations and browse directory listings until you lock down access by setting permissions.



See “Ways to lock down a server” on page 290 for other ways to lock down the server.

Default object security

Object security defaults are as follows:

- Users have runtime Read and Execute access to all objects (except administration objects.)
- If you define a particular access at the **directory** level, it becomes the default access for any new objects created within that SilverStream directory. A directory is a container for objects of one type—for example, all forms are in the Forms directory. When you have selected the Permissions panel in the SMC, you can expand directories to see their contents (assuming you have Read permission on that directory). Any new object takes on the access security of the immediate parent object. For example, if you create a new form in a SilverStream database, the form is saved in the Forms directory and takes on the security settings of the Forms directory.

Default group permissions

During installation, SilverMasterInit creates two predefined groups (Administrators and Developers) and sets permissions for both groups. The server requires all users to login. Default group permissions for a locked-down server are as follows:


- Access to server administration operations is restricted to members of the SilverStream Administrators group. Access to administration resources (such as who can use the SMC, view session and statistical information, add and remove users and groups) is only granted to members of this group.

- Developers can access design resources (directory listings) so they can use the SilverStream Designer. Developers, however, will not be able to add a database in the Designer until you allow access.

You typically separate who is allowed to read an object (that is, view design-time information) from who is allowed to write it. You may also want to define a separate group (such as end users) with Execute permission.

 For more information about predefined SilverStream groups, see “About Silver Security users and groups” on page 127.

NOTE By default, users will have Read access to the top level of the SilverMaster database and directories that will enable them to log in and access any existing application databases.

 For more information, see “Making secure application objects executable” on page 285.

Locking down servers, clusters, and applications

At some point in the development and deployment process, you will want to **lock down** an application database, a server, and/or an entire cluster. Locking down these items helps secure your production and deployed application environment by making sure the appropriate access permissions are set.

For example, when deploying an application, you might want to lock down the application database so no one can access it except you, then progressively unlock it as appropriate in the production environment.

CAUTION *If, when you installed the SilverStream server, you did not choose the **Restrict Access to the SilverStream Server** option, your server resources are not locked down. Even if you aren't ready to lock down and deploy your applications, you should nevertheless **lock down** your server.*

Ways to lock down a server

The following sections describe different ways you can restrict server access:

Section	Information
“Default server and object security” on page 288	Describes default SilverStream server installation settings
“Using the SMC to lock down the server or an application” on page 290	Describes how to use the SMC to lock down the server or an application
“Using SilverCmd to lock down the server, an application, or a cluster” on page 291	Describes how to use SilverCmd to lock down the server, an application, or a cluster
“Security checklist” on page 292	Provides an important security overview
“Securing the development server” on page 299	Describes additional considerations when configuring an application development environment

Using the SMC to lock down the server or an application

➤ To lock down the specified server:

1. Restrict the **Read Server Configuration**, **Modify Server Configuration**, **Read Directory Listing**, **Read User & Groups**, and **Set Permissions** permissions at the server or cluster level to members of the Administrators group.
2. Select the SilverMaster database and restrict **Read**, **Modify**, and **Set Permissions** permissions to members of the Administrators group and apply that restriction to all descendants.
3. Set unrestricted **Read** access to the SilverMaster database to allow users (with appropriate permissions) to access other databases and log in to the server.
4. To restrict retrieval access to the SilverMaster database from within SilverStream, restrict the **Select** permission to administrators at the Tables directory level and apply to all descendants.



For more detailed instructions, see “Step 9: Secure the SilverMaster database” on page 297 in the security checklist.


➤ **To lock down the selected application:**

1. Restrict the **Read, Modify, Default Execute**, and **Set Permissions** permissions at the database level to members of the Administrators group and apply the restriction to all descendants.
2. To restrict retrieval access to the database from within SilverStream, restrict the **Select** permission to administrators at the Tables directory level and apply to all descendants.

Using SilverCmd to lock down the server, an application, or a cluster

Included with the SilverStream server (in \Samples\SilverCmd in the SilverStream installation directory) are three sample XML files that are input files for the SilverCmd SetSecurity command. These files show how you can lock down an application, a server, or a cluster using SilverCmd. All three files are well commented with usage notes.

Secure file	Shows you how to lock down
secure_server_sample.xml input file	An entire server You might want to apply this file to a server when putting it into production
secure_appication_sample.xml input file	An application. You might want to apply this file to an application database just before deploying it in order to secure it properly
secure_cluster_sample.xml input file	A cluster You might want to apply this file after you have secured each server in a production cluster using secure_server_sample.xml

 For information on using SilverCmd, see the SilverCmd Reference in the *Facilities Guide*.

Securing the production server

The levels and types of security that you implement depend on the scale and demands of your particular enterprise. Once your application is ready to deploy (or even if it is already deployed), you should go through each step in the following security checklist to secure your production environment.

If you chose the installation default, the SilverStream server restricts access to the SilverMaster database. When access to SilverMaster is restricted, unauthorized users will not be able to access administration operations and browse directory listings.

During application development, you may have opened up access to specific application objects and directories. Once your applications are built, it is up to you to make sure that **all** resources are protected from unauthorized access.

Security checklist

TIP To test your site's security, run any accompanying tests for each step.

Step 1: Design firewalls

Make sure the SilverStream server is installed **behind** any firewalls your company uses.



For more information, see “Server configurations” on page 27.

Step 2: Set up a unique database account

Set up a unique database account for the SilverStream server to use to connect to each database.



For more information, see Chapter 4, “Data Source Configuration”.

Step 3: (Optional) Set up SSL

If you intend to use SSL communications, obtain a server certificate and install it on the server. You also need to enable the RSA and/or DSA ports and disable HTTP if you want to require only SSL.



For more information, see “Using certificates” on page 226.

Step 4: (Optional) Set up unique ports

For added security, configure separate runtime, design, and administration ports. The server supports ports for the following three security protocols: HTTP, HTTPS-RSA, and HTTPS-DSA. Each unique port you configure excludes URLs and operations that are not associated with it. The separate ports are designed to work in conjunction with your server permission settings.



For more information, see “Setting up separate ports” on page 106.

Step 5: Set up users, groups, and security providers

User and group information can be defined in SilverStream or can be obtained from an external security system. For SilverStream-defined entities, all information is stored in the SilverMaster database catalog. For external security, all information is obtained from the external system. In either case, define access to directories and objects.



For more information, see “Managing Silver Security users and groups” on page 130.

Step 6: Require authentication at the server

If you restrict the **Execute** access to all resources in your SilverMaster database, you **must** enable **Require user authentication** at the server level. Otherwise, requiring user authentication is optional.

When a user can access the server without logging in or sending a certificate, the user is considered Anonymous. It is often a good idea to prohibit Anonymous user access by forcing users to authenticate themselves when they first access the server, through either a certificate or a user ID/password pair. You can also require login on a per-object basis.

➤ **To test for unauthorized user access:**

- To assess Anonymous user’s ability to access to your site, enter an URL to your application into a browser such as:

```
http://localhost/
```

If the browser displays a **login dialog**, your site is requiring anonymous users to authenticate themselves.

If the **default page** or a **listing** of your site’s directory contents (or a Read Access Denied message) displays, your site is allowing anonymous access and you may want to complete the steps the following procedure to require user authentication.

➤ **To require user authentication:**

1. Start the SMC and select a server from the left pane.
2. Select the **Security** icon from the toolbar.
3. Select **General**.
4. Check the **Require user authentication** check box.
5. Click **Update** to save the settings.



For more information, see “Enabling authentication” on page 266.

Step 7: Restrict the directory listing on the server

SilverStream displays a directory listing when users request certain URLs from a browser or from the Designer. Although a listing of directory entries may not seem terribly critical to your site’s security, you may still want to prevent these lists from appearing.

To see whether unauthorized users can access the HTML and non-HTML directory listings, run the following two procedures.

➤ **To check whether the HTML directory listing is enabled (and optionally disable it):**

1. From your browser, enter the URL of a directory on your server such as:

`http://localhost/SilverStream/Meta/`

If your HTML directory listing is **protected**, the following error message displays:

You are not allowed to read the specified resource.
Additional technical details about this error are available.

If an HTML directory list displays, you may want to disable the directory listing access as described in the following steps.

2. Open the SMC and select the **Security** icon from the toolbar.
3. Check the **Disable HTML directory listing** check box.
4. Click **Update** to save the settings.

➤ **To check whether the non-HTML directory listing is enabled (and optionally disable it):**

1. From your browser, enter an URL from a browser such as:

```
http://localhost/SilverStream/Meta/?access-mode=text
```

If your non-HTML directory listing is **protected**, the following error message displays:

```
You are not allowed to read the specified resource.
Additional technical details about this error are available.
```

If you see the directory contents listed in plain text, you need to disable directory listing access as described in the following steps.

2. Open the SMC and select the **Security** icon from the toolbar, then select **Permissions**.
3. Set the **Read Directory Listing permission** to limit access as described in “To check the Read Server Configuration setting of your administration resource:” on page 296.
4. Click **Update** to save the settings.



For more information, see “Enabling authentication” on page 266.

Step 8: Secure the administration resource

The **administration resource** controls your ability to view, modify, and change administrative settings (and permissions to access settings). You secure server objects by restricting permissions on the SilverStream server administration resource. Once you have secured the administration resource, unauthorized users will not be able to perform administrative operations.

Run the following two tests to check access to your server. But even if the test URL does not access protected server information, you should use the SMC to verify that your administration resource security settings properly restrict access to the appropriate users.

➤ **To check the general accessibility of your administration resource:**

1. Open the SMC.
2. As an anonymous user, try to view and change application objects.

On a **secure** server you see:

```
Read Access Denied.
Additional technical details about this error are available.
```

3. If you are able to view or change application objects, you should **promptly** run the procedure “To protect administration access:” on page 296.

➤ **To check the Read Server Configuration setting of your administration resource:**

1. Enter the following URL for your Web site:

`http://localhost/SilverStream/Administration/`

On a **secure** server you see:

Read Access Denied.

Additional technical details about this error are available.

If you see a listing of site-specific server properties (similar to the following text sample) listed in plain text, your site is **not** protected.

```
#Written by SilverStream Server
#Mon Oct 16 19:08:36 EDT 2000
com.sssw.srv.fulltextsearch.noindexonsearch=false
com.sssw.loadbalancer.connect.tryInterval=30
com.sssw.srv.server=SilverStream\ Server/4.0
com.sssw.srv.http.ClientPool.minIdle=0
com.sssw.loadbalancer.connect.sleepCount=10
com.sssw.srv.http.ClientPool.minFree=10
```

2. If the administration resource at your site is not protected, you should **promptly** run the procedure “To protect administration access:” on page 296.

CAUTION *If you discover that your administration resource was left unrestricted, you should immediately change the user name and password for each database as well as the password for each mail account the server connects to. These steps will help ensure that anyone who may have accessed this information while the resource was unrestricted can no longer use it.*

➤ **To protect administration access:**

1. Start the SMC and select a server from the left pane.
2. Select the **Security** icon from the toolbar.
3. Select **Permissions**.
4. On each permission tab, deselect the **Unrestricted** check box to restrict access.

It is especially important that you restrict the **Read Server Configuration** permission by deselecting the **Unrestricted** check box.

5. Select either **Simple List** or **Advanced Expression**.

6. Select the users and groups or define an expression that specifies who will be assigned the permission on the selected tab. You should limit access on all five tabs (listed in the table in “Administrative server permissions” on page 272) to members of the **Administrator** group.



For more information, see “Permission types” on page 272.

Step 9: Secure the SilverMaster database

Special care should be taken when securing the SilverMaster database. In addition to storing resources such as user and group information, SilverMaster stores the login resource and references to all other application databases configured in the server.

Unless your applications are tightly controlled, you don't typically apply Read restrictions to the SilverMaster database. You must allow Read access to the top level of the SilverMaster database in order for users to be able to access anything on the server. You may want to lock down all of the resources below the main directory level.

If you accidentally restrict Read access to your SilverMaster database, enable **Require user authentication** on the Server Security panel. If you forget to require authentication, run SilverMasterInit with the -a option to enable user authentication from the command line. You will need to restart the server for the authentication change to take effect.

➤ To test access to your SilverMaster database:

- To see whether your SilverMaster database can be accessed by unauthorized users, follow “Step 10: Secure your application databases” on page 298.
In Step 4 of that procedure, select the SilverMaster database that you think should be restricted. Select an object that is stored beneath the parent level of the directory structure of the SilverMaster database to test access.

➤ To lock SilverMaster resources below the main directory level:

1. Open the SMC as an administrator and select the **Security** icon from the toolbar.
2. Select **Permissions**.
3. In the left panel, select the SilverMaster database beneath the desired server.
4. Restrict **Read**, **Modify**, and **Set Permissions** to members of the Administrators group and then click the **Apply to this directory and all descendants** radio button. For more information about permission settings, see the table in “Administrative server permissions” on page 272.
5. Click **Update** to save the settings.

6. In the left pane, expand the **SilverMaster** database and select the **Tables** directory.
7. Restrict **Select** access to members of the Administrators group and click the **Apply to this directory and all descendants** radio button.
8. Click **Update** to save the settings.
9. Reselect the SilverMaster database and set unrestricted **Read** access.
10. Click **Update** to save the settings.
11. Repeat Steps 9 and 10 for each of the directories and subdirectories expanded beneath SilverMaster (such as **Tables, Forms, Views, EJB JARs &Media, and Pages, General, Images, Jars, and Sounds**).
12. Click **Update** to save the settings.



For more information, see “Configuring the SilverMaster database” on page 58.

Step 10: Secure your application databases

You should restrict user access to application databases by setting permissions on directories and objects (such as pages, views, and business objects) using simple or advanced expressions.

Permission settings apply to the directory, all subdirectories, and objects in the directory or any subdirectory. For each application, review every object and specify who is allowed Read, Write, Default Execute, and Set Permission access on each directory level in the SilverStream hierarchy. Each object or resource within a hierarchy must be secured.

Unless you set restrictions, all users have full access permissions on **objects**. If you define a particular access on a **directory**, it becomes the default access for any new objects created within that directory. A directory is a container for objects of one type—for example, all forms are in the Forms directory. When you have selected the Permissions panel in the SMC, you can expand directories to see their contents (assuming you have Read permission on that directory). You can secure multiple directories by selecting the **Apply to this directory and all descendants** radio button at the bottom of the permission tabs. You can enable the **Require login for access** check box for selected objects as needed.

NOTE If your application objects contain EJBs, you must unrestrict Read access to the parent directory of the objects directory (and to all descendants), since the object directory contains classes that need to be read by the application.

➤ To test access to database objects (and optionally secure them):

1. Log in to the SMC as an Administrator or user with Locksmith privilege.
2. Select the **Security** icon from the toolbar.
3. Select **Permissions**.

4. In the left panel, select a page (or other object) that you think should be restricted that is stored beneath the parent level of the directory structure.
5. Click each tab and see who has access to this object. Complete the following steps if the permission restrictions are not set correctly.
6. Click each tab and select either **Simple List** or **Advanced Expression**.
7. Select the users and groups or define an expression that specifies who will be assigned the permission on the selected tab.
8. Make sure that someone in the **Administrator** group has **Set Permission** access to this object.
9. Click **Update** to save the settings.



For more information, see “Database object permissions” on page 273 and “Making secure application objects executable” on page 285.

Step 11: Map J2EE security roles

When deploying J2EE archives, map the security roles specified in the deployment descriptor to security principals in the production environment. For example, if the bean developer has defined three security roles, these same roles must be mapped to security groups or individual users on the target server. If you decide to map roles to groups (for ease of maintenance), be sure to verify group membership.

Securing the development server

Many of the guidelines in the preceding security checklist also apply to securing the development server. For example, you can use secure communications between the SilverStream server and Java clients such as the SilverStream Designer and SMC through a DSA or RSA certificate.


➤ To secure your development server:

1. Add your developers to the Developers group and restrict access to this group as described in the following steps.
2. (Optional) Configure a separate port for use by your developers.
3. Restrict any objects from specific developers as needed. You can do this at the server, cluster, or object level. You need to be logged in to the server as an administrator with the Locksmith privilege.

4. You can require login for forms, tables, views, and pages by activating **Require login for access**.
5. Secure access by directory or object type—for example:

Directory or object type	Allow write access to
The Tables directory	(For classic applications only) Only designated database developers
The Form, View, and Pages directories	(For classic applications only) Only developers
The Objects directory	(For classic applications only) Only designated business object developers The ability to write a business object is like superuser access. A business object can bypass other SilverStream security and shut down the server or otherwise compromise data.
The EJB JARS & Media directory and its subdirectories	For J2EE applications, only deployers. For classic applications, only developers.

NOTE If necessary, set up row-level security on selected tables.

 For more information, see “Authorization and access control” on page 271 and “Ways to lock down a server” on page 290.

Excluding robots

Robots are programs, such as search engines and Web crawlers, that can traverse many pages on a Web site by recursively retrieving linked pages. The SilverStream server supports the commonly used **robot exclusion protocol** that allows you to specify exactly what files and directories robots that conform to the exclusion protocol can access on your Web site.

NOTE The robot exclusion protocol is purely voluntary. There is no guarantee that a robot conforms to the protocol, but most do.

In this exclusion protocol, the access policy for robots is specified in a file called **robots.txt**. The robots.txt file shipped with SilverStream tells robots not to proceed below the root of the server (that is, it disallows all access to all robots that conform to the exclusion protocol). So by default, conforming robots are disallowed all access to the Web site you have developed with SilverStream.

You can modify robots.txt to specify the kind of access you want to provide to robots. For example, you might want to allow search-engine access to some directories but not to others.

➤ To modify robot access:

- Edit **robots.txt** in the **SilverStream\resources** directory.



For information about the protocol's format and semantics, including examples, see <http://www.robotstxt.org/wc/robots.html>.

The next time a conforming robot requests the access policy from the server, the updated robots.txt file will be read and the information sent back to the robot.

11 Tuning the Server

This chapter describes how to use the SilverStream Management Console (SMC) to manage server parameters to improve the overall performance and efficient operation of your system.

This chapter contains sections on:

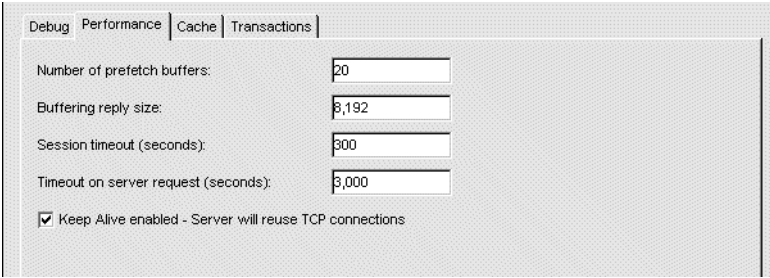
- Setting performance parameters
- Managing client connections
- Managing the server content cache
- Managing connection pools for J2EE applications
- Managing database connections for classic applications

Setting performance parameters

To improve performance, you can use the SMC to set buffer sizes as well as server timeout parameters.

➤ To set performance parameters:

1. Start the SMC.
2. Select the **Configuration** icon from the toolbar.
3. Select **Advanced**.
4. Select the **Performance** tab.



The screenshot shows the SMC configuration interface with the 'Performance' tab selected. The interface includes a tabbed menu at the top with 'Debug', 'Performance', 'Cache', and 'Transactions'. Below the menu, there are four input fields for performance parameters: 'Number of prefetch buffers' (value: 20), 'Buffering reply size' (value: 8,192), 'Session timeout (seconds)' (value: 300), and 'Timeout on server request (seconds)' (value: 3,000). At the bottom, there is a checked checkbox labeled 'Keep Alive enabled - Server will reuse TCP connections'.

5. Edit the settings as needed.

Field	Description
Number of prefetch buffers	The total number of prefetch buffers to use with this server. See “Using prefetch buffers” on page 325.
Timeout on server request	<p>The maximum total server-side time for processing a client request, from the instant the request is received until the reply is sent back. This allows the server to time out in the event of a persistent problem.</p> <p>You should set this value to be substantially longer than the longest expected server response time.</p>
Session timeout	<p>The time after which the server will terminate a session with a client after the client goes idle. The default value is five minutes. Java clients and the SilverStream Designer running in the browser ping the server every four minutes to keep their session alive.</p> <p>You probably don’t want to reduce this value below four minutes, as it might interfere with the Designer, SilverJRunner, SilverJ2EEClient, and Java form pinging, resulting in the server dropping active connections.</p> <p>The session timeout value affects how long session object variables are maintained. The longer the timeout, the longer the memory used for a session will be consumed by the server. In deployed applications that do not use session objects, you probably want to use a short session timeout.</p>

Field	Description
Buffering reply size	<p>The size of the packet (in bytes) in which larger replies are gathered, copied, and sent back to the client.</p> <p>Setting this to a larger size might provide better performance, but it is best to limit the size to 8KB or 16KB, depending on available server physical memory and the number of client connections.</p>
Keep alive enabled	<p>When checked client connections can be reused for additional client requests to the server.</p> <p>The server uses an HTTP(s) listener that accepts client requests via the TCP/IP protocol. The server is configured to have a certain number of threads that handle the client TCP/IP connections (see “Managing client connections” on page 306).</p> <ul style="list-style-type: none">• If Keep alive enabled is checked (true) after handling a new TCP/IP request, the server reads the next client request from the same client TCP/IP connection.• If Keep alive enabled is disabled, after handling a new TCP/IP request, the server closes the client TCP/IP request.

6. Click **Update**.
7. To activate the new setting(s), click the **Restart** (server) button.

Managing client connections

This section describes how a client connection is established on the server, and how you can modify connection parameters to improve performance.

NOTE **Client** connections are not the same as **database** connections. Database connections are described in “Managing database connections for classic applications” on page 318.

Client sessions and threads

When a client first connects with the server through HTTP, the server establishes the connection by allocating a **thread** (if available) from the connection pool.

About threads

The thread is a lightweight background process that:

- Creates a server session
- Executes the HTTP request by hooking up to the appropriate server components and passing the result back to the client

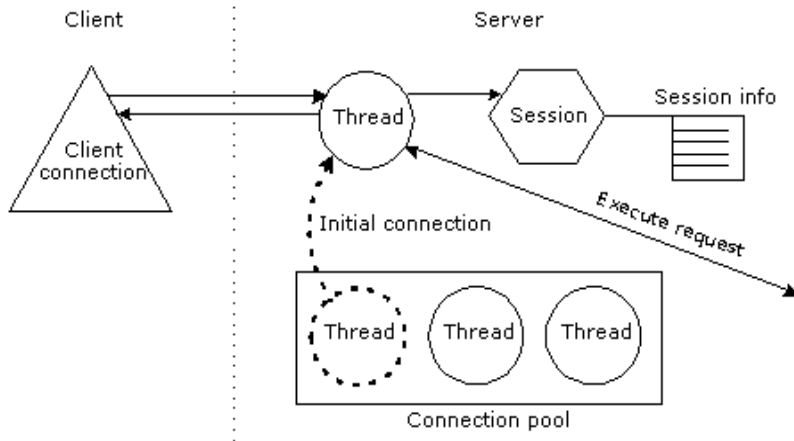
The thread is associated with the client connection until either the client or the server closes it. The server will close the connection according to the settable connection parameters. When the connection is closed, the thread is returned to the connection pool.

About session objects

The **session** is an object on the server that does the following:

- Stores information about the client (such as user ID, login, and hostname).
- Stores the application data included programmatically in the client’s session object (such as in a shopping cart application, which needs to store purchases and other information for each client along the way).
- Remains alive for a specified time (session timeout period) after all threads have been returned to the pool. (In certain cases there can be more than one client thread associated with a session.) This allows the client to reconnect to the session within the specified time. The default timeout is five minutes. Once a session object has been deleted by the server, its contents are gone.

The following diagram shows a client connection with the server session and its associated thread.



Client connection parameters

The SilverStream server imposes a limit on the total number of allowable client connections (threads). Within this allowance, you can modify connection parameters in order to manage the server load in your production environment. The server defines client connections in terms of individual connection states and overall loads.

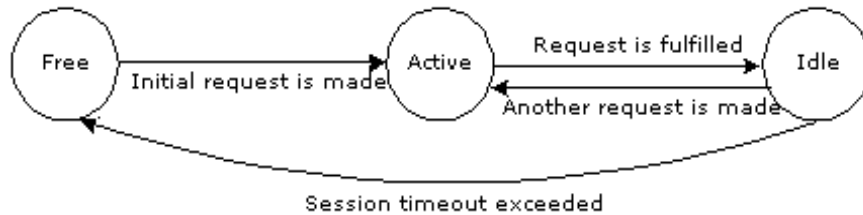
Connection states

Each client connection (thread) exists in one of the following states:

- **Free**, if it is not in use
- **Active**, if it is connected, in use, and a request is in progress
- **Idle**, if it is connected and in use, but no request is in progress on the connection

Once a thread has been idle for the session timeout period, SilverStream closes the connection and returns the thread to the connection pool. (You can configure the timeout period. See “Setting performance parameters” on page 303.)

A thread typically has the following life cycle:



Load levels

The server has two load levels: **light** and **busy**. It behaves differently when requests for new connections are made, depending on whether its load is light or busy.

How the SilverStream server operates based on load levels

Here is what happens when the server receives a request for a new connection:

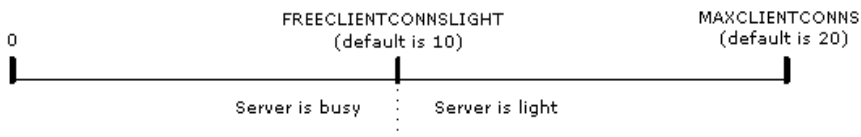
1. A request is made for a new connection to the server.
2. SilverStream responds based on its load level.

Load level	What the SilverStream server does
Light	The new request is allocated a free thread. When an active connection goes idle, it is kept open until it times out.
Busy	The server closes older idle connections (enough to bring the server to the light load level if possible), returning the threads to the connection pool, then allocates a free thread to the new request. If there are no idle connections (that is, all the allowable number of connections are active), the new request is refused.

Specifying how the load level is determined

You can specify how the server determines whether the load is **light** or **busy** by setting the following properties. In the descriptions that follow, the terms in **bold** refer to the label for the properties in the SMC and the terms in parentheses are the names of the corresponding properties in the AgiAdmServer interface in the Server Administration API (without the PROP_ prefix).

- **Maximum number of client connections** (MAXCLIENTCONNS). The total number of allowable connections. The default is 20.
- **Free client connections for “light”** (FREECLIENTCONNSLIGHT). Determines whether the server is considered to be light or busy. The default is 10.
 - If the number of free connections is less than FREECLIENTCONNSLIGHT, the server is considered to be busy.
 - If the number of free connections equals or exceeds FREECLIENTCONNSLIGHT, the server is considered to be light.



NOTE Two other properties—**Free client connections for “busy”** (FREECLIENTCONNSBUSY) and **Idle client connections for “light”** (IDLECLIENTCONNSLIGHT)—are used internally and should be set as described in “Setting the connection parameters” on page 310.

Determining an appropriate value for MAXCLIENTCONNS

You should set MAXCLIENTCONNS to the maximum number of connections that will be active simultaneously on your server, plus approximately 10 percent for peak times. The server creates a separate thread for handling each active connection.

Setting this parameter too high will result in excessive consumption of system resources by your server, and setting it too low will result in connections being refused.

A general guideline is to set this parameter to the maximum number of simultaneous sessions you expect your server to serve (plus 10 percent or so) assuming there will be about one active connection per session (but remember that complicated forms may require more than one connection, so there may be more connections than users).

Determining an appropriate value for FREECLIENTCONNSLIGHT

Typically you will set FREECLIENTCONNSLIGHT low if your application has a stable user load, such as with an intranet application. Setting FREECLIENTCONNSLIGHT low means that the application is usually in a light load, so idle connections are maintained longer—allowing existing users to maintain their connections longer.

For Internet applications, where the user load can vary dramatically, you might want to set FREECLIENTCONNSLIGHT relatively high, to better ensure that new users can get a connection. Idle connections are terminated more quickly.

Setting the connection parameters

You can set the connection parameters in the SMC.

➤ To set client connection parameters:

1. Start the SMC.
2. Select the **Configuration** icon from the toolbar.
3. Select **Connections**.

Maximum number of client connections:	<input type="text" value="20"/>	(The total number of allowable connections)
Free client connections for "busy":	<input type="text" value="5"/>	(This setting is used internally and should be set to the same value as Free client connections for "light")
Free client connections for "light":	<input type="text" value="10"/>	(Determines whether the server is considered to be light or busy - server is busy if the number of free connections is less than this setting)
Idle client connections for "light":	<input type="text" value="20"/>	(This setting is used internally and should be set to the same value as Maximum number of client connections)

4. Specify settings as follows.

Field	Description	What to specify
Maximum number of client connections	MAXCLIENTCONNS. Determines the maximum number of connections.	Set as described in “Determining an appropriate value for MAXCLIENTCONNS” on page 309.
Free client connections for busy	FREECLIENTCONNSBUSY. Used internally.	Set to the same value as FREECLIENTCONNSLIGHT.
Free client connections for light	FREECLIENTCONNSLIGHT. Determines when the server is considered to be light or busy.	Set as described in “Determining an appropriate value for FREECLIENTCONNSLIGHT” on page 310.
Idle client connections for light	IDLECLIENTCONNSLIGHT. Used internally.	Set to the same value as MAXCLIENTCONNS.

5. Click **Update**.

The new settings take place immediately.

Managing the server content cache

SilverStream uses server-side caching for various purposes, including resource naming and attribute information, database schema information, and prefetch buffering. SilverStream also stores (or caches) the contents of file resources, such as media store objects, in memory or on disk. Caching is mostly invisible to users except that it affects performance.

The two caches

There are two separate caches for every SilverStream server:

Cache	Description
Memory cache	This cache is held completely in memory, and is intended for holding smaller files
Disk cache	This cache is on disk, and is intended for larger files

What you can configure

Each cache has two settings you can configure:

Setting	Description
The maximum content size	The maximum size of any individual file allowed into the cache
The maximum cache size	The maximum size of the cache itself (that is, the total size of all the files in the cache)

You can also specify the directory used for disk caching.

How SilverStream uses the caches

The server uses the most appropriate cache for each file, based on the file's size. Small files are stored in the memory cache, and larger files are stored in the disk cache. Very large files are not cached at all. The server uses an LRU (least recently used) algorithm within each cache, throwing out old (not recently used) files when the cache becomes full.

➤ To set cache settings:

1. Start the SMC.
2. Select the **Configuration** icon from the toolbar.
3. Select the **Advanced** option.
4. Select the **Cache** tab.

The following panel displays.

Debug | Performance | **Cache** | Transactions

Content Cache Enabled

Maximum size of the disk cache (bytes):

Maximum size of any file that will be cached in the disk cache (bytes):

Maximum size of the in-memory cache (bytes):

Maximum size of any file that will be cached in the in-memory cache (bytes):

Directory for disk cache entries:

5. Specify the settings as follows.

Field	Description
Content cache enabled	Enables or disables the content cache. If unchecked, all content caching is turned off. Caching frequently requested file resources will improve the response time for these resources. Turn off only for debugging purposes.
Maximum size of the disk cache	The maximum size of the disk cache, in bytes. The total size of all files in the cache is always less than or equal to this. If set to 0, the disk cache is disabled. In general, make this cache large enough to store the remainder of frequently used files after you have determined the in-memory cache size (see the description of “Maximum size of the in-memory cache” below).
Maximum size of any file that will be cached in the disk cache	The maximum size of any file that is cached in the disk cache, in bytes. Files less than or equal to this size but too large for the in-memory cache are cached in the disk cache. Files greater than this size are not cached.

Field	Description
Maximum size of the in-memory cache	<p>The maximum size of the in-memory cache, in bytes. The total size of all files in the cache is always less than or equal to this. If set to 0, the in-memory cache is effectively disabled.</p> <p>In general, make this size as large as possible without causing excessive paging activity on your system.</p>
Maximum size of any file that will be cached in the in-memory cache	<p>The maximum size of any file that is cached in the in-memory cache, in bytes. Files less than or equal to this size will be cached in the in-memory cache.</p>
Directory for disk cache entries	<p>The directory in which the disk cache entries are stored. By default, this is a subdirectory of the SilverStream installation directory.</p> <p>The server tries to create this directory if it doesn't exist. The server also tries to clear out all the cache files from this directory when it starts up. (You might want to have a directory that is dedicated to the cache.)</p> <p>If the server cannot find or create this directory, the disk cache is disabled.</p>

6. Click **Update**.

The new settings take effect immediately.

Managing connection pools for J2EE applications

This section contains the following topics:

- About connection pool connections
- Setting the number of connection pool connections

About connection pool connections

The SilverStream server handles all connections between a J2EE application and a data source via a connection pool. The connections in the connection pool are typically TCP/IP connections, established between the SilverStream server and the data source through JDBC or a RAR.

Most database or EIS servers place a limit on the number of connections that can be open simultaneously. You can use the SMC to set the minimum (initial) and maximum number of connections used by the SilverStream server for each data source.

For maximum performance, the maximum number of open SilverStream data source connections you set will normally equal the maximum number of users who will be simultaneously querying or updating the data source.

Determining the correct number of connections

Setting a maximum that is too small will result in degraded performance: if a client attempts to access the data source when all connections are in use, the client will be blocked until an in-progress query or update is completed. You might want to experiment with different maximum settings to optimize your server performance.

If other applications will be establishing connections to the data source server independently of the SilverStream server, you may need to reduce the maximum number of open SilverStream connections. This ensures that data source connections are available for non-SilverStream users.

Consult your DBMS or EIS documentation and ensure that your data source has been configured to accept the number of client connections specified in the SMC, taking into account other applications that might be accessing the data source.

Setting the number of connection pool connections

You can use the SMC's Pools tab to manage both JDBC and Connector connection pools including:

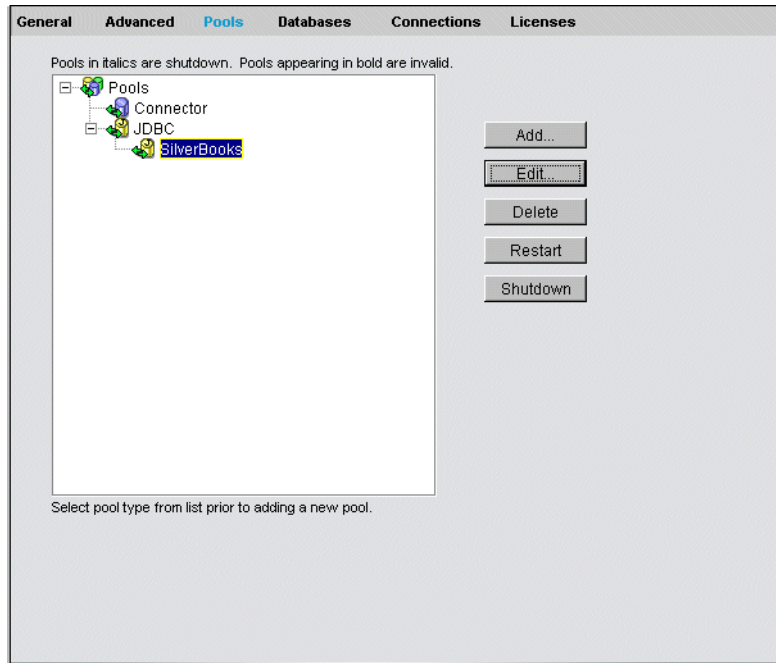
- Minimum and maximum number of connections
- Idle timeouts
- Connection timeouts
- Logging levels

You specify these values for each individual connection pool.

➤ To updating the connection pool settings:

1. Start the SMC.
2. Select the **Configuration** icon from the toolbar.
3. Select **Pools**.
4. Select either **Connector** or **JDBC** to display the available connection pools.

5. Highlight the connection pool whose connection values you want to modify, and choose **Edit**.



The JDBC Edit Connection Pool Wizard displays. You can review various settings for the connection pool, but can only update the minimum and maximum connection values and the connection pool timeout values.

6. Choose **Next** to proceed through the wizard until you reach the panel that displays the minimum and maximum connection values shown here:

Field	Description
Minimum connections	<p>The default minimum number of connections for a connection pool. The SilverStream server will immediately establish this many connections when it is started, and keep them open as long as the server is running.</p> <p>NOTE If the data source crashes and then restarts, SilverStream will drop the old connections and reestablish new ones as needed. It will not immediately establish what had been specified as the minimum number of connections.</p>
Maximum number connections	<p>The default maximum number of connections for connection pool. The SilverStream server will open connections on demand up to this number.</p> <p>Make sure that the maximum number of open SilverStream connections is less than or equal to the total number allowed by the EIS; otherwise, the SilverStream server will get errors from the data source server when it attempts to open a connection.</p>


Field	Description
Idle Connection Timeout	The idle timeout in seconds. The default is 60 seconds. When set to -1, idle timeout is disabled and no idle connections are ever closed
Connection Wait Timeout	The connection wait timeout in seconds. The default is 30 seconds. When set to -1, clients are forced to wait until a connection becomes available.
Log Level	The levels are: 0—Logging is turned off 1—Logs basic connection pool operations 2—Level 1 with more detailed operations and error messages 3—Level 2 with exception stack traces and trace output produced by JDBC driver or Connector resource adapter. Messages are written to the server console.

7. Click **Update**.
8. To activate the new settings, click the **Restart** button.

Managing database connections for classic applications

This section contains the following topics:

- About database connections and performance
- Setting the maximum and minimum number of database connections
- Using prefetch buffers

 For information about the SilverStream data access architecture in SilverStream Classic applications, including its data cache objects, see the data access chapter in the *Programmer's Guide* of the SilverStream server's Classic Development Help.

About database connections and performance

The SilverStream server handles all connections between a SilverStream client and a database for SilverStream Classic applications. Each database connection is typically a TCP/IP connection, established between the SilverStream server and the database server through JDBC.

Most database servers place a limit on the number of database connections that can be open simultaneously. You can use the SMC to set the minimum (initial) and maximum number of connections used by the SilverStream server for each database.

The maximum number of open SilverStream database connections you set will normally equal the maximum number of users who will be simultaneously querying or updating the database.

Exception

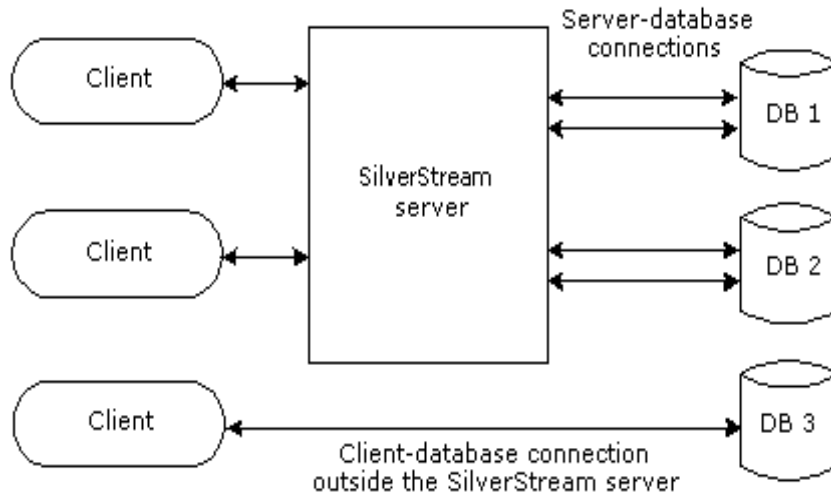
Complicated forms may require more than one database connection, so the number of connections may exceed the number of users.

Determining the correct maximum number of connections

Setting a maximum that is too small will result in degraded performance: if a client attempts to access the database when all connections are in use, the client will be blocked until an in-progress query or update is completed. You may want to experiment with different maximum settings to optimize your server performance.

If other database users will be establishing connections to the database server independently of SilverStream, you may need to reduce the maximum number of open SilverStream connections. This ensures that database connections are available for non-SilverStream users.

You should consult your DBMS documentation and ensure that your database has been configured to accept the number of client connections specified in the SMC, taking into account other applications that may be accessing the database.



Setting the maximum and minimum number of database connections

You can set the default maximum and minimum number of database connections as well as the number of connections for specific databases. You do this on the Databases panel in the SMC.

Setting the default number of connections

You can set the default minimum and maximum number of database connections. These values are used by:

- Subsequently added databases.
- All databases currently on the server that are using the default minimum or maximum number of connections (that is, databases whose minimum or maximum number of connections have not been changed from the default).

NOTE To override the defaults for a specific database, see “Setting the number of connections for a specific database” on page 323.


➤ **To set the default maximum and minimum number of database connections:**

1. Start the SMC.
2. Select the **Configuration** icon from the toolbar.
3. Select **Databases**.

The screenshot shows the 'Databases' configuration tab in the eXtend Application Server Administrator's Guide. The window has a title bar with tabs for 'General', 'Advanced', 'Pools', 'Databases', 'Connections', and 'Licenses'. The 'Databases' tab is active. The main area contains the following fields and controls:

- At the top, there are two spinners for 'Min # of database connections (Applies to all databases):' (set to 2) and 'Max # of database connections (Applies to all databases):' (set to 10). Each spinner has a 'Reset all' button to its right.
- Below these is an 'Add Database...' button.
- The 'Database settings:' section includes a 'Database:' dropdown menu currently showing 'SilverMaster40'.
- Below the dropdown are 'Username:' and 'Password:' text boxes. The 'Username' box contains 'dba' and the 'Password' box contains '*****'.
- Below the password box are two spinners for 'Min Connections:' (set to 2) and 'Max Connections:' (set to 10). Each spinner has a 'Reset' button to its right.
- To the right of the 'Max Connections' spinner is a 'Remove Database...' button.
- Below these are several text labels and their corresponding values:
 - 'Database platform:' Sybase Adaptive Server Anywhere 6
 - 'Driver set:' SilverStream JDBC-ODBC bridge
 - 'JDBC Driver:' com.sssw.jdbc.mss.odbc.AgOdbcDriver
 - 'JDBC URL:' jdbc:sssw:odbc:SilverMaster40
 - 'JDBC URL attributes:'
- At the bottom of the configuration area are two buttons: 'Synchronize Database Schema' and 'Delete Idle Connections'.
- A large blue 'Update' button is located at the bottom right of the window.

4. Edit the fields at the top of the panel as needed.

Field	Description
Min # of database connections (Applies to all databases)	<p>The default minimum number of connections for a database. The SilverStream server will immediately establish this many connections when it is started, and keep them open as long as the server is running.</p> <p>NOTE If the database crashes and then restarts, SilverStream will drop the old connections and reestablish new ones as needed. It will not immediately establish what had been specified as the minimum number of connections.</p>
Max # of database connections (Applies to all databases)	<p>The default maximum number of connections for a database. The SilverStream server will open connections on demand up to this number.</p> <p>Make sure the maximum number of open SilverStream connections is less than or equal to the total number allowed by the database server; otherwise, the SilverStream server will get errors from the database server when it attempts to open a connection.</p> <p> For more information, see “About database connections and performance” on page 319.</p>

5. Click **Update**.

Resetting the defaults

You can reset the default minimum and maximum number of connections. This applies to **all** databases on the server.

➤ To reset the defaults:

- Click **Reset all** next to Min # or Max # of database connections at the top of the Databases panel.

The default minimum or maximum number of connections is reset, and **all** databases on the server are reset to use the new default values.

Setting the number of connections for a specific database


You can override the default numbers of database connections for a specific database.

➤ To override the default minimum/maximum connections for a specific database:

1. Start the SMC.
2. Select the **Configuration** icon from the toolbar.
3. Select **Databases**.
4. Select the database you want to configure from the list box (under Database settings).

General	Advanced	Pools	Databases	Connections	Licenses
Min # of database connections (Applies to all databases):		2		Reset all	
Max # of database connections (Applies to all databases):		10		Reset all	
Add Database...					
Database settings:					
Database: SilverMaster40					
Username: kba		Password: *****			
Min Connections: 2		Reset		Max Connections: 10	
				Reset	
Remove Database...					
Database platform:		Sybase Adaptive Server Anywhere 6			
Driver set:		SilverStream JDBC-ODBC bridge			
JDBC Driver:		com.sssw.jdbc.mss.odbc.AgOdbcDriver			
JDBC URL:		jdbc:ssw:odbc:SilverMaster40			
JDBC URL attributes:					

5. Set the minimum and maximum number of database connections for the selected database by editing the fields below the database name.

Field	Description
Min connections	<p>The SilverStream server will immediately establish this many connections when it is started, and keep them open as long as the server is running. Setting a value here overrides the default value.</p> <p>NOTE If the database crashes and then restarts, SilverStream will drop the old connections and reestablish new ones as needed. It will not immediately establish what had been specified as the minimum number of connections.</p>
Max connections	<p>The SilverStream server will open additional connections on demand up to this number. Setting a value here overrides the default value.</p> <p>Make sure the maximum number of open SilverStream connections is less than or equal to the total number allowed by the database server; otherwise, the SilverStream server will get errors from the database server when it attempts to open a connection.</p> <p> For more information, see “About database connections and performance” on page 319.</p>

6. Click **Update**.

Restoring the values to the defaults

You can reset the minimum or maximum number of connections for a specific database to be the default values.

➤ **To restore minimum/maximum connections to the defaults:**

1. Select the database you want to configure from the list box.
2. Click **Reset** next to Min Connections or Max connections for the selected database.

Database settings:

Database: SilverMaster40

Username: dba Password: *****

Min Connections: 2 Max Connections: 10

Reset Reset Remove Database...

The minimum or maximum number of connections is reset to be the default value (shown at the top of the panel).


Using prefetch buffers

With SilverStream classic applications uses **prefetch buffers** to speed database operations. A prefetch buffer is initiated by a database query from a SilverStream form or view. When a table is returned and displayed in a form or view, the rows are visible to the user one screen at a time. However, the server will fetch more than one screenful of rows and store them in the prefetch buffer, in anticipation of the user wanting to view the entire set.

NOTE Prefetch buffers are not used with pages (except when a Java form or view is included on the page), with business objects, or with J2EE applications.

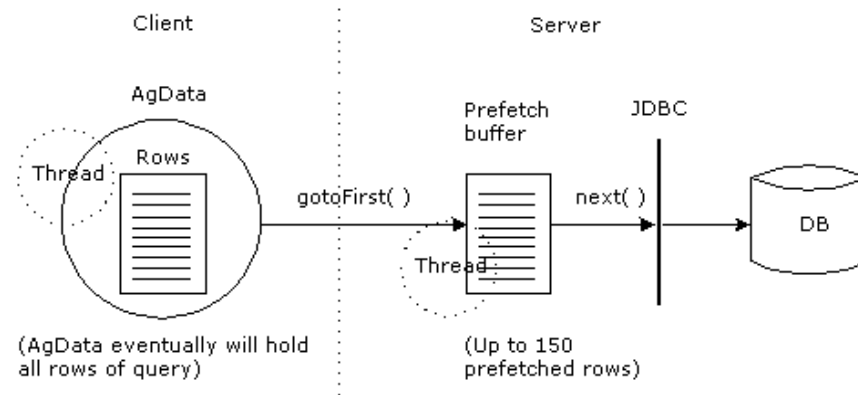
Important note about prefetch buffers

Prefetch buffers are temporary; they do not hold data until it is requested. Unless the creator of the form or view specified the **Limit number of returned rows** feature, **all** the rows resulting from the query will go to the client, even if there are millions of rows. As soon as the client asks for any data, it will eventually get all of it. (This is true also with pages; all the data will eventually go to the client unless the user cancels the operation.)

 For more information about the **Limit number of returned rows** feature, see the Form Designer, Page Designer, or View Designer in the *Tools Guide* of the SilverStream server's Classic Development Help.

How prefetch buffers are managed

The following diagram and the description below it illustrate how SilverStream manages prefetch buffers.



At the start of a client query, SilverStream allocates a prefetch buffer that fetches 150 rows from the database. The following describes a generic sequence.

1. Using the client interface, the user does `gotoFirst()` on the client `AgData`.
2. A background thread starts on the client, requesting 100 rows from the server.
3. 100 rows are moved from the prefetch buffer to the communication line. A prefetch thread gets 100 more rows from the database.
4. Eventually the client thread gets the last row from the prefetch buffer, which frees the prefetch thread.

You can set the number of prefetch buffers for a server. Typically, you want to allocate one prefetch per user (matching the number of client connections). You might see a performance improvement if you set the number higher than the default. Before you do so, make sure you have the server capacity (sufficient memory) to handle the additional memory overhead of the additional prefetch buffers.

Here is a rule of thumb for calculating how much memory a prefetch buffer might consume:

$$\text{Average row size in bytes} * 150 \text{ rows}$$

where Average row size is the sum of the sizes of each of the columns being accessed by an `AgData` in your application.

12 Administering a Cluster

This chapter describes how SilverStream uses server clustering to implement load balancing and failover and explains how to set up and maintain a clustered environment.

The chapter contains the following sections:

- Server clustering
- Cluster components
- Component failover
- Setting up a server cluster
- Administering a server cluster
- Specifying a server's relative load weight
- Managing failover
- Dissolving a cluster
- Changing the clustering components' properties
- Installing certificates in a cluster
- Setting up Fulcrum in a cluster

Server clustering

In order to accommodate high processing loads, SilverStream implements load balancing using server clustering.

What is a server cluster?

A **server cluster** is a set of servers running on different hosts that share the processing load. In the SilverStream server environment, a cluster is a group of independent systems working together as a single system connected to the same SilverMaster database catalog. In this configuration, a Web client interacts with a cluster as though that cluster were a single high-performance, highly reliable application server.

NOTE A cluster of servers can handle demand over time more efficiently than one large machine, because the sum of the bandwidth and resources of multiple machines is greater (and less expensive) than that of a single large machine.

Benefits of server clustering

SilverStream server clustering provides the following benefits:

Benefit	Description
Scalability and better performance	You can increase the number of requests that get processed over a period of time. When the overall load exceeds the capabilities of the systems in the cluster, additional systems can easily be added to the cluster.
Cache management	If a change is made to application logic or presentation, the SilverStream Cache Manager ensures that the change is propagated to each server in the cluster.
Load management	SilverStream provides a Load Manager with a dispatcher that distributes client requests to servers in order to optimize overall performance. You can also use a third-party load manager.
Failover	SilverStream provides failover capability for each component of the server cluster software. Specific system failover capabilities are described in “Managing failover” on page 354.

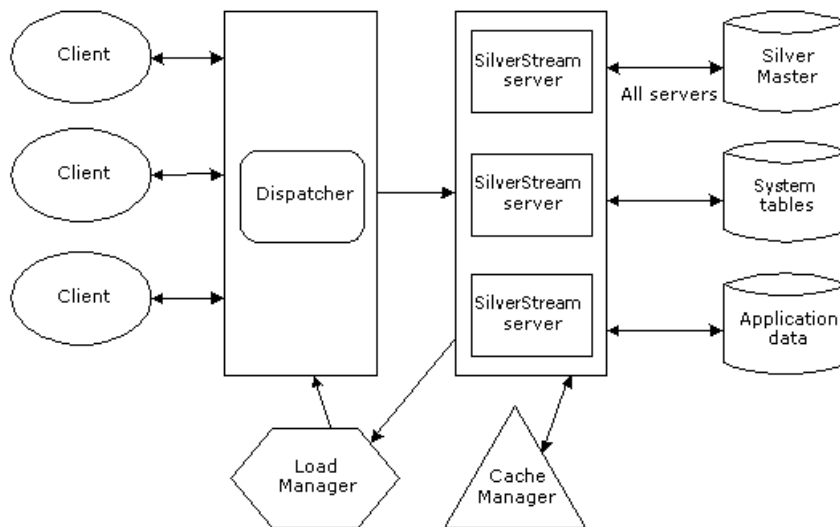
Cluster components

A SilverStream server cluster has five components.

Component	Number	Purpose
SilverMaster database	One	Keeps track of cluster membership
SilverStream server	One or more (usually at least two)	Serves applications
Load Manager	One (optional but requires Dispatcher when used)	Manages the activity of the Dispatchers

Component	Number	Purpose
Dispatcher	One (optional but requires Load Manager when used)	Used by the Load Manager to redirect requests to SilverStream servers
Cache Manager	One	Keeps server caches in sync

The cluster components are configured as follows.



Cluster requirements A cluster configuration must meet these general requirements:

- Each server in the cluster must communicate with the same SilverMaster catalog.
- Each server must have a unique address and port.
- Each component in the cluster must be able to communicate over TCP/IP.
- A client machine capable of running the SilverStream Management Console (SMC) must be available to create and configure the cluster.

Cluster options You have the following options within the cluster configuration:

- Servers can reside on any machine in the network.
- You can have an unlimited number of servers per cluster.
- Servers and components can run on different platforms.

NOTE When servers operate in a clustered environment, they may affect the way scheduled and server-triggered business objects work. For more information, see the *Programmer's Guide* of the server's Classic Development Help.

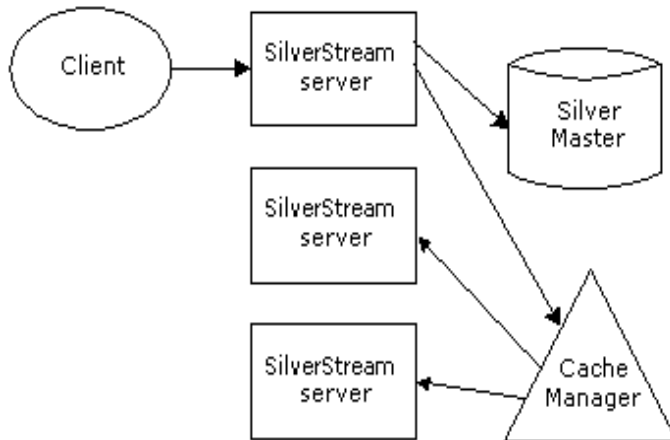
The Cache Manager

Each SilverStream server has an intelligent caching mechanism for storing commonly accessed data in memory (such as HTML pages, forms, and security information). Reading this information from the database for each request would be very inefficient, so the server maintains this information in cache memory. Whenever the information is updated on the server (for example, a page or form has changed, a new security permission is applied, or a J2EE archive is deployed), the server's cache is updated as well.

Maintaining this cache across multiple servers in a cluster is important for application and data consistency. In a clustered environment, multiple servers can simultaneously change the same data in the SilverStream system tables. This situation might leave data cached in memory in a corrupted or inconsistent state. The Cache Manager is responsible for preventing such conflicts by ensuring that servers in the cluster are notified when another server invalidates cache objects. Servers then discard their invalid cache entries and get an updated version of the resource the next time the object is needed.

How the Cache Manager works

The SilverStream Cache Manager can run on a separate machine or on any machine in the server cluster. The Cache Manager must be up and running before any servers in a cluster start. As part of server startup, the server reads the clustering information about the setup from the SilverMaster catalog and initiates contact with the Cache Manager. The Cache Manager registers the server's existence and identifies it as a member of the cluster.



If a server modifies an existing object, it notifies the Cache Manager that the specific object must be invalidated on other servers. Since all objects in a SilverStream server environment can be denoted by an URL, the server calls the Cache Manager to invalidate the specified URL for all servers. The Cache Manager calls each registered server except the one that initiated the invalidation, and tells it to invalidate the object identified by the URL.

For example, a developer makes a change to a servlet in the development environment and deploys the new servlet to the cluster. The Cache Manager sees that the class has been updated in the database and instructs all SilverStream servers in the cluster to invalidate their caches so that the new object can be read from the database and the updated copy retained in memory.

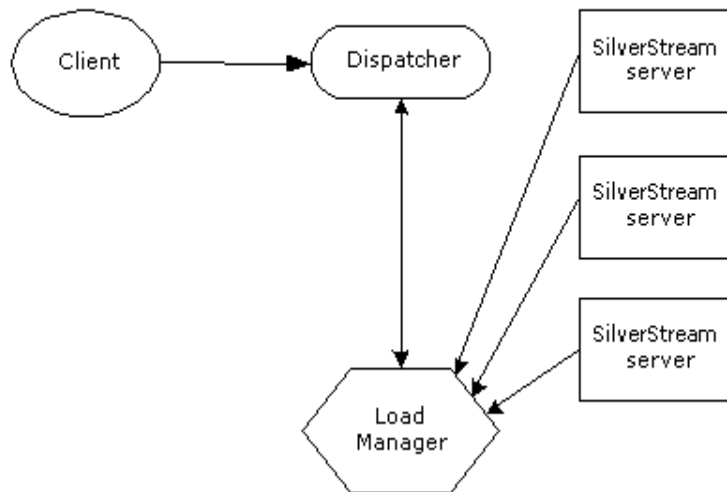
The Load Manager

The Load Manager is a program that can run as a service or regular process on any machine in the same network as the cluster servers. The Load Manager's role is to keep track of each active server and its relative processing load (or weight) in the cluster. Based on this information, the Load Manager generates a distribution map, which it transmits to the Dispatcher in the configuration.

How the Load Manager works

The SilverStream Load Manager must be up and running before any servers in a cluster start. As part of server startup, the server reads the clustering information about the setup from the SilverMaster catalog and initiates contact with the Load Manager to register the server's existence. The Load Manager first registers that the server is up, then determines how to contact the server and factors the server into the distribution map. As part of Load Manager startup, the Load Manager reads the Dispatcher information about the setup from its property file and initiates contact with the Dispatcher so that it can transmit the distribution map in the future.

The figure below shows the Load Manager.




Distribution mapping

The distribution map, which is dynamically generated by the Load Manager, determines the processing load for each server in the cluster. The distribution weight for each server is represented by an integer between 1 and 10.

How the load is distributed By default, a **round-robin system** is used, which means that each server in the cluster gets an equal number of hits over time and has the same weight (or no weight).

You can modify this distribution. For example, if you set Server 1 at 8, Server 2 at 4, and Server 3 at 2, over a period of time Server 1 would get twice as many hits as Server 2, and Server 2 would get twice as many hits as Server 3. The table that follows shows other examples of weight settings.

Servers in clusters	Weight	Result
1, 2, 3	null	Round robin
1, 2, 3	1, 1, 1	Round robin
1, 2, 3	2, 4, 6	Server 1 is least hit, Server 2 eventually has twice as many hits, and Server 3 eventually has three times as many hits as Server 1
1, 2, 3	1, 1, 10	Server 3 eventually has 10 times as many hits as Server 1 and Server 2

 For information on modifying the distribution, see “Specifying a server’s relative load weight” on page 353.

The Dispatcher

The Dispatcher serves as the entry point for Web clients into a server cluster: clients initially access the cluster through the Dispatcher’s URL.

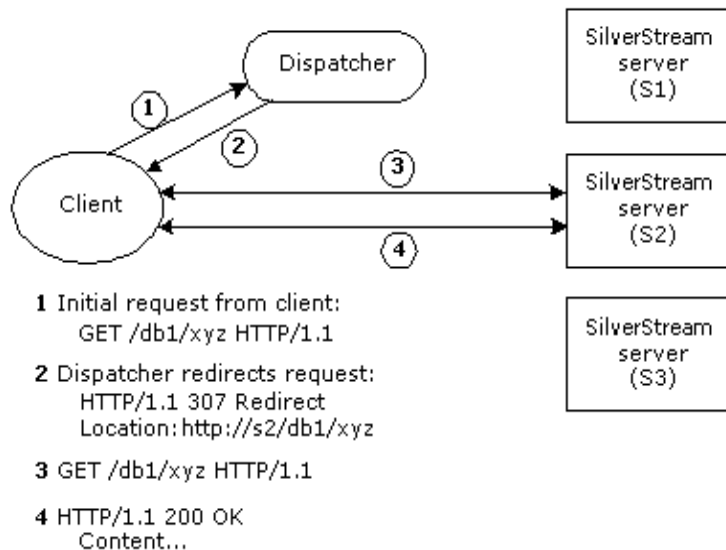
The Dispatcher is a lightweight HTTP/HTTPS-supported program that communicates with other cluster components to dispatch client requests to servers according to the distribution map the Load Manager provides. The Dispatcher can be run as a service or as a regular process on any machine in the same network as the cluster servers. The Dispatcher and Load Manager can run on separate machines and on separate platforms, and do not need access to the SilverMaster database.

How the Dispatcher works

The first time the Dispatcher is called from the client, it finds the best server for the request, based on the distribution map it receives from the Load Manager. Then, using a standard process called HTTP redirect, it redirects the request back to the client. The client then sends the request directly to the designated server.

Once a session has gone through a Dispatcher to a particular server, all client communication is sent directly to the server, not through the Dispatcher; and the redirection is not masked (meaning that the user sees the URL for the server, not the URL for the Dispatcher, in the browser).

The following figure shows the sequence in the HTTP redirect process.



The Dispatcher supports HTTPS (as well as both HTTP 1.0 and HTTP 1.1). This means you can install server certificates after you set up a cluster. See “Administering a server cluster” on page 348.

Using third-party dispatching solutions

While the SilverStream Dispatcher is an excellent solution for many load-balancing and failover needs, you may require functionality that is not available in the SilverStream Dispatcher.

You may want to use a third-party dispatching solution for the following reasons:

Reason	More information
Complex load-balancing algorithm	When you want more control over how load distribution is calculated, you may want to use another dispatching solution.
DNS masking	<p>Since the SilverStream Dispatcher performs a simple HTTP redirection to an available server in the cluster, the browser must be able to establish a direct connection to that server. All clients therefore must be able to resolve all TCP/IP host names for server members. On the other hand, dispatching solutions with DNS masking capabilities process all incoming and outgoing traffic, so all servers can be resolved using one common host name.</p> <p>This capability is particularly useful in Internet applications where it is desirable for users to see only one host name (such as <code>www.company.com</code>), instead of a host name for each server and dispatcher.</p>
Session-level failover	For J2EE applications whose deployment plans indicate failover support (WARs marked as distributable and EJB JARs marked as recoverable) and classic SilverStream applications (developed using the SilverStream <code>AgoPersistentStateManager</code> object), many third-party dispatchers can automatically reroute a session to another server in the cluster completely transparently to the user.
Global dispatching	You may want to have dispatchers that can do more complex routing—for example, to different sites around the world.

In situations like these, you may want to substitute a third-party dispatching solution for the SilverStream Load Manager and Dispatcher. You would still use the SilverStream Cache Manager to maintain a consistent cache and would still use the SMC to manage the server.

Any standard HTTP server load-balancing solution should work with the SilverStream server.

Component failover

SilverStream provides system failover and recovery in the event of transient failure and persistent failure. If any component in a cluster fails, the failure is detected by a background program called **SilverMonitor**.

About SilverMonitor

SilverMonitor observes the state of the daemons, processes, and services running on the system. Each component of the server cluster is monitored by SilverMonitor. If any component fails, SilverMonitor detects the failure and attempts to restart the component.

SilverMonitor usually ensures that failed components recover quickly. If a failure is persistent (due, for example, to a hardware failure), SilverMonitor gives up after a predefined number of attempts in order to conserve system resources.

NOTE SilverMonitor runs in each cluster by default. It is also provided as a server startup option through which you can define certain program parameters. For more information, see “Using SilverMonitor” on page 414.

If SilverMonitor cannot restart a server

When a server in a cluster goes down and is not restarted promptly, the following sequence occurs:

1. The Cache Manager detects the failure.
2. The Load Manager detects the failure and removes the server from the distribution map.
3. The Load Manager sends an updated map to the Dispatcher so that new clients are not redirected to the failed server.

When the server is restarted

When the server is restarted, the following sequence occurs:

1. The server reconnects to the Cache Manager.
2. The server reconnects to the Load Manager.
3. The Load Manager rebuilds the distribution map and sends it to the Dispatcher.
4. The server begins to receive client requests.

When a server fails, connected clients lose their connections. They need to either restart or send another request from the browser to the Dispatcher.

Persistent failure

If restarting a server takes a significant amount of time, the cluster components force a reconfiguration of the cluster so that incoming client requests can be serviced as efficiently as possible. During a persistent failure, the active components respond as follows:

When this goes down	This happens
SilverStream server	The failure is detected by the Load Manager, which instructs the Dispatcher(s) to redirect traffic to active servers, as described above
Load Manager	The distribution map cannot be updated, but the Dispatcher still works using the cached version of the map
Dispatcher	No new connections can be made. Existing sessions are not affected
Cache Manager	Applications whose logic or properties have not changed continue to run correctly

Setting up a server cluster

The SilverStream clustering components (Cache Manager, Load Balancer, and Dispatcher) are installed with SilverStream Enterprise Edition, but not when installing the Developer Edition.

➤ To set up a server cluster:

1. Using the SilverStream installation program, install the first server and create the SilverMaster catalog against which all of the server members will run. If you have already completed the installation, make sure the SilverMaster setup is appropriate for the cluster.



For information about installing SilverStream and configuring the SilverMaster database, see the *Installation Guide*.

2. Using SilverCmd or the SMC, add any application data sources your system will need.

3. Using the installation program, install the clustering components on one or more machines. You can use a combination of platforms (UNIX platforms and Windows NT). The SilverStream clustering components (Cache Manager, Load Balancer, and Dispatcher) can also be installed separately in the Enterprise Edition install by choosing Custom Install and selecting the Cluster Managers option.



For more information, see the *Installation Guide*.

4. Start the clustering components.



For more information, see “Starting the clustering components” next.

If you installed the components as daemons or services, you can stop and restart them in this mode.

5. Using the installation program, install each of the additional servers that will run in the cluster.



For more information, see “Installing cluster servers” on page 340.

6. Start the SMC and create the cluster profile.



For more information, see “Creating the cluster profile” on page 342.

7. After creating the profile, restart all servers and components to activate the cluster.



For more information, see “Restarting the clustered servers” on page 347.

Starting the clustering components

If the load-balancing software is not currently running as a service on your network, you need to run the programs manually on the resident machine(s). If you are using the SilverStream Load Manager, execute the commands in the order shown in the procedure that follows.

➤ To start the cluster components:

NOTE The programs in the steps below are all in the SilverStream installation directory.

1. Start the Cache Manager program.

- For Windows NT:

```
\bin\SilverCacheMgr.exe
```

- For UNIX:

```
/bin/SilverCacheMgr
```

2. Start the Dispatcher program.
 - For Windows NT:


```
\bin\SilverDispatcher.exe
```
 - For UNIX:


```
/bin/SilverDispatcher
```
3. Start the Load Manager program.
 - For Windows NT:


```
\bin\SilverLoadMgr.exe
```
 - For UNIX:


```
SilverStreamInstallDir/bin/SilverLoadmgr
```

You can specify the VM to start with each of these executables. For more information, see “Specifying the VM to use” on page 98.

Using startup parameters with SilverDispatcher You can start SilverDispatcher with the following parameters:

Parameter	Description
<i>-p propfile</i>	<p>The name and location of the an alternate Dispatcher.ddl file.</p> <p>The Dispatcher.DDL file is created by the SMC at cluster configuration time. It specifies the default ports for RMI, HTTP, and HTTPS for the dispatcher and it is located in the server's Resources directory.</p> <p>Using an alternate Dispatcher.ddl file is not recommended. Editing this file outside of the SMC is also not recommended.</p>
<i>-c upload-certificate</i>	<p>Run the SilverDispatcher in upload-certificate mode.</p> <p>For more information, see “Installing certificates in a cluster” on page 358</p>
<i>-h host</i>	<p>The host name specified here is converted to an IP address via:</p> <pre style="margin-left: 2em;">InetAddress.getByName(host_name);</pre> <p>When specified, the dispatcher will open a serversocket that is listening on this ip address only. If not specified, the socket will listen on all ipaddresses of the local machine</p>

Parameter	Description
+cp:p <i>path</i>	Prepends specified <i>path</i> to the class path. Don't use this debugging option without first contacting SilverStream Technical Support. Instead, use AGCLASSPATH to make additional Java classes available to SilverStream server applications. For more information, see "Setting the AGCLASSPATH variable" on page 138.
+cp:a <i>path</i>	Appends specified <i>path</i> to the class path. This option makes additional Java classes available to SilverStream applications by appending the specified path to the class path. NOTE It is recommended that you use the AGCLASSPATH environment variable to extend Java classes. For more information, see "Setting the AGCLASSPATH variable" on page 138.

Installing cluster servers

After you have installed the server that will contain the cluster's SilverMaster database, use the SilverStream installation program to install additional servers. When installing additional servers, you need to point them to the first server you installed (which contains the SilverMaster database), since all servers in a cluster need to use the same SilverMaster.

This section provides installation instructions for Windows NT and UNIX. See the *Installation Guide* for more detailed information.

➤ To install cluster servers (Windows NT):

1. Start the Setup program on the machine you want to add to a cluster.
2. Select **Typical**, **Full**, or **Custom** (if you choose Custom, make sure the SilverStream server and Common components are selected).
3. When prompted, make sure **Configure SilverStream Server (execute SilverMasterInit)** is unchecked. Instead, you will use the same SilverMaster that was used by the first server you installed. Also, make sure that **Install a new Server configuration file** is checked. Click **Next**.
4. Select the database type for the SilverMaster. The database type must match the database type used by the first server you installed, which contains the SilverMaster database.

5. If you selected Sybase Adaptive Server Anywhere, you are asked whether you want to use an existing database or create a new database. Select **Use an existing database for SilverMaster** and click **Next**.
6. Enter the SilverMaster name and the other database information that you are prompted for. You must specify the same SilverMaster used by the first server you installed for the cluster.

Each machine in the cluster must be a copy of the other. This means that:

- You must have the identical ODBC names for the SilverMaster and any additional databases that your SilverMaster references on all servers in a cluster.
 - Any necessary DBMS client software that is required must be installed on all systems.
 - JDBC drivers must reside on all servers participating in the cluster.
7. Complete the installation program.
 8. When prompted, enter your license string for the server.

If an error displays, you probably misentered information about the existing SilverMaster database. Go back to Step 4 and respecify that information.

After adding each server to the cluster, you can use the SMC to create the cluster profile, as described in “Creating the cluster profile” on page 342.

➤ **To install cluster servers (UNIX):**

1. Execute the SilverStream install script (**install.sh**) on the machine you want to add to the cluster.
2. Enter the location where you want to install SilverStream.
3. Enter the database type and user and password information for your cluster's SilverMaster.

NOTE Your ability to include a UNIX server in a cluster with NT servers may be affected by the availability of database drivers on both UNIX and NT.

Each machine in the cluster must be a mirror copy of the other. This means that:

- You must have the identical ODBC names for the SilverMaster and any additional databases that your SilverMaster references on all servers in a cluster.
 - Any necessary DBMS client software that is required must be installed on all systems.
 - JDBC drivers must reside on all servers participating in the cluster.
4. When prompted to upgrade your databases, answer **no**.
 5. To bypass rerunning SilverMasterInit, answer **no** when prompted to configure your SilverStream server.

6. Enter the port number for your server.

NOTE If you have already added this server to your cluster using the SMC, the port value you specify here must match the value you entered when you added the server to the cluster.

7. Enter your license string.

If an error displays, you probably misentered information about the existing SilverMaster database. Go back to Step 3 and respecify that information.

After adding each server to the cluster, you can use the SMC to create the cluster profile, as described next.

Creating the cluster profile

Once you have multiple servers pointing to a single SilverMaster, you are ready to configure the cluster profile using the SMC. A cluster profile defines which servers, ports, and security protocols are included in the cluster. A SilverStream server can only be included in one cluster.

You must use an HTTP port to create a cluster. Furthermore, if you have configured an administration port, you must also use that port to create the cluster.



For more information, see “Using separate ports with your firewall” on page 107.

➤ **To create a cluster profile:**

1. Make sure the Cache Manager, Dispatcher (if used), and Load Manager (if used) are running.

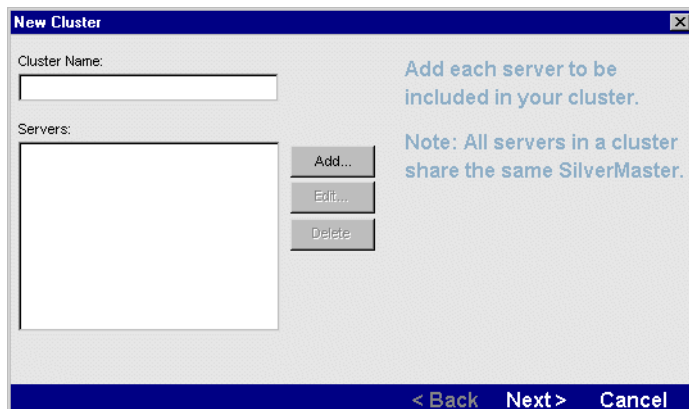


For more information, see “Starting the clustering components” on page 338.

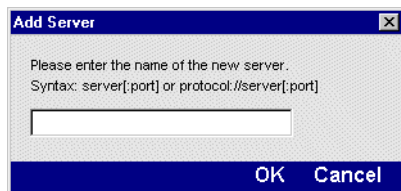
2. Start the SMC.

3. Click **New** (cluster) from the toolbar.

The New Cluster Wizard displays.



4. Enter a cluster name, then click **Add**. The following panel displays.



5. Enter a fully qualified name followed by a port number and then click **OK**. For example:

`agserver.silverstream.com:50001`

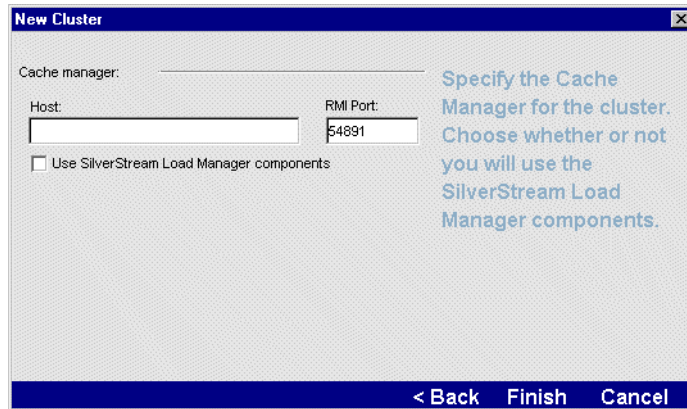
Be sure the first server you add is the one containing the SilverMaster. All subsequent servers you add must be configured to use the same SilverMaster.

If your port(s) are set to 80, you don't need to specify a port number. However, if you have defined an administration port, be sure to specify that port number. If SilverStream is not your primary Web server, you must change the port to a number above 5000. For instructions on changing ports, see "Specifying general server properties" on page 110.

6. Click **Add** again and enter the server name for as many servers as you intend to add to the cluster. Each server is listed in the New Cluster form as you add it to the cluster.

7. Click **Next** to set up your Cache Manager.

The following panel displays.



8. Enter the host name of the Cache Manager. The default RMI port number for the Cache Manager is 54891. When initially creating the cluster profile, you should specify the default port, but you can later change the port if necessary.



For information about changing the default ports, see “Changing the clustering components’ properties” on page 357.

9. If you plan to use the SilverStream Load Manager, select the **Use SilverStream Load Manager components** check box and go to the next step.

If you do not plan to use the Load Manager at this time, click **Finish** and go to “Restarting the clustered servers” on page 347.

- Enter the host name of the Load Manager. The default RMI port is 54891. When initially creating the cluster profile, you should specify the default port, but you can later change the port if necessary.

New Cluster

Load manager: _____

Host: RMI Port:

Dispatcher: _____

Host: RMI Port:

The load manager and dispatcher must be running before completing this wizard.

Add... Edit... Delete

< Back Finish Cancel



For information about changing the default ports, see “Changing the clustering components’ properties” on page 357.

- Click **Add** to add the Dispatcher.

The following panel displays.

Add Dispatcher

Host:

RMI Port:

The dispatcher must be running in order to be added to the cluster environment.

HTTP Settings:

Runtime Port: Design Port: Admin Port:

RSA Settings:

Runtime Port: Design Port: Admin Port:

DSA Settings:

Runtime Port: Design Port: Admin Port:

OK Cancel


- Enter the host name for the Dispatcher.

13. Specify port settings for any or all of the ports in each protocol type you want the Dispatcher to listen on.


The Dispatcher listens on all configured port types: HTTP, RSA, and DSA for any of the following unique server ports that you have already configured and enabled:

- A runtime port for users executing application objects in the cluster
- A design port for developers using servers in the cluster
- An administration port for use with SMC operations.

You can disable HTTP port(s) and use HTTPS or RMI client communications. For more information, see “Turning off HTTP communications” on page 252.

 For information about default port settings for each security protocol, see “Port types” on page 109.

The following table describes the port settings that appear in the panel. When initially creating the cluster profile, you should specify the default ports for each type unique port and protocol, but you can later change the ports if necessary.

Item	Description
RMI Port	The port for Dispatcher communications with the Load Manager. Used by the Dispatcher, Cache Manager, and Load Manager. The default is 54891.
HTTP Settings:	The HTTP port(s) for unencrypted communications with clients used by the Dispatcher. You can configure up to three HTTP ports.
Runtime Port	By default, the cluster runtime, administration, and design ports use the same default port number: 54892.  See “Turning off HTTP communications” on page 252.
Design Port	
Admin Port	

Item	Description
RSA Settings: Runtime Port Design Port Admin Port	The RSA port(s) for encrypted communications (using HTTPS) used by the Dispatcher with Java clients and HTML clients. By default, the Dispatcher uses the same default port number (54893) for all RSA runtime, administration, and design ports in the cluster.
DSA Settings: Runtime Port Design Port Admin Port	The DSA port(s) for encrypted communications (using HTTPS) used by the Dispatcher with Java clients. By default, the Dispatcher uses the same default port number (54894) for all DSA runtime, administration, and design ports in the cluster.



For information about changing the default ports, see “Changing the clustering components’ properties” on page 357.

14. Click **OK**. You return to the New Cluster panel.
15. Click **Finish**.

The new cluster is configured.

Restarting the clustered servers

After defining the cluster profile, you must restart each server. If you are running the SilverStream Load Manager, you must also restart the Load Manager and Dispatcher.

➤ To restart a server:

1. Select the server in the left panel in the SMC.
2. Click **Restart** (server).



For more information about restarting servers, see “Restarting the SilverStream server” on page 100.

➤ To restart the Load Manager and Dispatcher:


- See “Starting the clustering components” on page 338.

Administering a server cluster

After creating a cluster profile, the SMC updates to display options specific to a clustered environment.

About properties in a clustered environment

When you are working in a clustered environment, you need to be aware of three types of properties:

Property type	Description
Server local properties	<p>Properties that are specific to an individual server and stored in the server's <code>httpd.props</code> file. These are properties that are needed by a server in order to start, so they must be defined externally to be available when the server starts. These properties are not stored in the SilverMaster database.</p> <p> For a listing of these properties and information on setting them, see Appendix A, "The <code>httpd.props</code> File".</p>
Server stored properties	<p>Properties that are specific to an individual server and stored in the SilverMaster database (in the <code>AgProperties</code> table).</p> <p>For a listing of these properties, select a server in a cluster in the SMC and look at the properties listed in the panels. The listed properties include the server stored properties as well as the server local properties that are configurable in the SMC. All server stored properties are configurable in the SMC.</p>
Cluster shared properties	<p>Properties shared by all servers in the cluster. These properties are stored in the SilverMaster database (in the <code>AgProperties</code> table).</p> <p>These are cluster-level properties: all servers in the cluster share the same values for the cluster shared properties. Most of the security properties are cluster shared properties.</p> <p>For a listing of these properties, select a cluster in the SMC and look at the properties listed in the panels. All cluster shared properties are configurable in the SMC.</p>


How properties are set when a cluster is created or dissolved

When a cluster is created, any server in the cluster retains its **server local** and **server stored** properties from when it was originally configured as a standalone server. You can choose to retain these settings or change them at the server level.

 For more information, see “Setting server-level properties in a cluster” on page 352.


However, a server included in a cluster does **not** retain the values for properties that are defined as **cluster shared** properties. Because individual servers in the cluster may not have the same **cluster shared** property values as they did when they were standalone servers, you must reconfigure the **cluster shared properties** at the cluster level.

Accordingly, when you create a new cluster profile, SilverStream sets all the **cluster shared** values to the default values. You can keep these settings or change them at the cluster level. Once you change a cluster-level property, the new value will be applied to all the servers in the cluster. When a cluster is dissolved, all servers in the cluster become standalone servers.

 For more information, “Setting cluster-level properties” on page 349 next.


Setting cluster-level properties




As mentioned above, when working in a clustered environment, some properties exist at the cluster level and some exist at the server level. If you select the cluster at the left-hand side of the SMC, you see the cluster-level properties. Most cluster properties are the same as those set for standalone servers. The following table provides cross-references to the documentation for the cluster-level properties.

 To learn about the API for cluster-level properties, see `com.sssw.rts.adminapi.AgiAdmCluster` in the Server Administration API.



Configuration properties

The cluster-level configuration properties are grouped in several SMC panels.

Panel	Description
General	RMI/ORB and SSL for remote object properties for the cluster  For more information, see “Specifying ORB settings” on page 116.







Panel	Description
Advanced	<p>Performance, Cache Manager, and Load Balancer properties.</p> <ul style="list-style-type: none"> • Performance Timeout on server request; session timeout. See “Setting performance parameters” on page 303. • Cache Manager These properties exist only in clusters. See “Managing failover” on page 354. • Load Balancer These properties exist only in clusters. See “Managing failover” on page 354.
Licenses	<p>License information.</p> <p> For more information, see “Managing licenses” on page 136.</p>
Managers	<p>Cache Manager, Load Manager, and Dispatcher properties. These are the properties you specified when defining the cluster profile. You can edit these properties after creating a cluster.</p> <p> For information about these properties, see “Creating the cluster profile” on page 342.</p> <p> For information about changing these properties after you have created the profile, see “Changing the clustering components’ properties” on page 357.</p>
Servers	<p>Lets you add servers to and remove servers from the existing cluster profile and change a server’s load weight.</p>

For Classic SilverStream applications, there are also the following cluster-level configuration properties:

Panel	Description
Full Text	<p>Properties for handling Fulcrum SearchServer full-text search engine</p> <p> For more information, see “Setting Fulcrum full-text properties” on page 161.</p>
Mail	<p>Properties for handling e-mail.</p> <p> For more information, see “Setting up mail on the server” on page 159.</p>

Security properties

The cluster-level security properties are grouped in the following SMC panels:

Panel	Description
General	General security settings for the cluster.  For more information, see “Setting Up Security” on page 199.
Advanced	Client certificate levels and trusted clients list for the cluster.  For more information, see “Setting Up Security” on page 199.
Permissions	Read cluster configuration, modify cluster configuration, and set permissions for the cluster.  For more information, see “Setting Up Security” on page 199.
Users & Groups	Managing Silver Security and certificate users and groups for the cluster.  For more information, see “Setting Up Users and Groups” on page 127.
Certificates	View certificates that have been installed on the server and recognized Certificate Authorities.  For more information, see “Setting Up Security” on page 199.
Security Providers	Configure SilverStream to recognize external security providers, including Windows NT directory services, LDAP, NIS+, and certificate issuers.  For more information, see “Setting Up Security” on page 199.

Monitor properties

The cluster-level monitor properties are a subset of the properties for standalone servers. For more information about the monitor properties, see “Monitoring server activity” on page 146.



Setting server-level properties in a cluster



When a standalone server is added to a cluster, many of its server-level properties become cluster-level properties.

If you select a server in a cluster at the left-hand side of the SMC, you see the properties for a **server** in a cluster. Most properties for servers in a cluster are the same as those set for standalone servers. The following section provides cross-references to the documentation for the properties of servers in clusters.

Configuration properties

The configuration properties of a server in a cluster are grouped in the following SMC panels.

Panel	Description
General	<p>General settings for the server.</p> <p> For more information about the general properties, see Chapter 5, “Running the Server”.</p>
Advanced	<p>Debug, Performance, Cache, Transactions, Cache Manager, and Load Balancer properties.</p> <ul style="list-style-type: none"> • Debug See “Low-level debugging” on page 408. • Performance Number of prefetch buffers; buffering reply size. See “Setting performance parameters” on page 303. • Cache See “Managing the server content cache” on page 311. • Transactions See “Managing J2EE transactions” on page 144 • Cache Manager These properties exist only in clusters. See “Managing failover” on page 354. • Load Balancer These properties exist only in clusters. See “Managing failover” on page 354.
Pools	<p>Allows you to create, delete, restart, and shut down Connector and JDBC connection pools. It also allows you to edit pool properties.</p> <p> For more information, see “Configuring connection pools” on page 69.</p>

Panel	Description
Connections	Client connection properties.  For more information, see “Managing client connections” on page 306.
Databases	Information about the databases in the cluster (databases known to the cluster’s SilverMaster). You can modify the minimum and maximum number of database connections for each server in the cluster.  For more information, see “Configuring the SilverMaster database” on page 58.

Security properties


The accelerator settings for a server in a cluster are administered at the server level. For more information about accelerator settings, see “Using Cryptographic Hardware Integration” on page 268.

Monitor properties

The monitor properties in a cluster are the same as the properties for standalone servers. For more information about the monitor properties, see “Monitoring server activity” on page 146.

Specifying a server’s relative load weight

You can specify the relative processing weight for each server in a cluster. The Load Manager uses that information to generate a distribution map that determines the processing load for each server at runtime.

 For a description of how weighting works, see “Distribution mapping” on page 333.

➤ To specify a server’s relative load weight:

1. Select the cluster in the SMC.
2. Select the **Configuration** icon from the toolbar, then select **Servers**.
3. Select a server from the list and specify an integer in the **Server Load Weight** field.
4. Click **Update**.

5. Select the other servers and specify appropriate relative values.
6. To activate the new server weight settings, click the **Restart** (server) button for the servers.

Managing failover

The SMC provides access to properties that control how the Cache Manager and Load Manager respond in the event of a system failure. Normally you do not need to edit these properties.

The Cache Manager and Load Manager properties exist at both the cluster and server level. You can set the properties at the cluster level, which sets the values for each of the servers in the cluster. You can then override the values for any of the server in the cluster; but note that if you later change any of the properties at the cluster level, it will override settings you made at the server level.

Cache Manager properties

The Cache Manager properties determine how the Cache Manager will respond when its connection with a server fails.

➤ To set Cache Manager properties:

1. Start the SMC.
2. Select the cluster to set properties at the cluster level, or select a server within a cluster to set properties at the server level.
3. Select the **Configuration** icon from the toolbar.
4. Select **Advanced**.
5. Select the **Cache Manager** tab.

Start sleep interval (seconds):	<input type="text" value="30"/>
Reconnect sleep interval (seconds):	<input type="text" value="10"/>
Start try count:	<input type="text" value="10"/>
Reconnect try count:	<input type="text" value="10"/>

- Reset any of the properties.

Property	Description
Start sleep interval	The number of seconds the Cache Manager will delay before starting a series of attempts to reconnect with the server.
Reconnect sleep interval	The number of seconds between each series of reconnection attempts.
Start try count	The number of times to start a series of reconnection attempts before generating an error.
Reconnect try count	The number of times to attempt to reconnect in a series.

- Click **Update**
- To activate the new properties, click the **Restart** (server) button.

Load Manager properties

The Load Manager properties determine how the Load Manager will respond if its communication with a server fails.

➤ To set Load Manager properties:

- Start the SMC.
- Select the cluster to set properties at the cluster level, or select a server within a cluster to set properties at the server level.
- Select the **Configuration** icon from the toolbar.
- Select **Advanced**.

5. Select the **Load Balancer** tab.

Connect try interval (seconds):	<input type="text" value="30"/>
Connect sleep interval (seconds):	<input type="text" value="10"/>
Connect try count:	<input type="text" value="10"/>
Connect sleep count:	<input type="text" value="10"/>

6. Reset any of the properties.

Property	Description
Connect try interval	The number of seconds to delay after each retry series.
Connect sleep interval	The number of seconds to delay after each connection retry in a series.
Connect try count	The number of times the Load Balancer should begin a series of connection attempts with the server.
Connect sleep count	The number of connection retries in a series.

7. Click **Update**.
8. To activate the new properties, click the **Restart** (server) button.

Dissolving a cluster

You can dissolve a cluster, which does the following:

- Deactivates the cluster
- Deletes the cluster profile

➤ **To dissolve a cluster:**

1. Click **Dissolve** (cluster) in the SMC toolbar.
You are asked to confirm your action.
2. Click **OK** to dissolve the cluster and delete the cluster profile.

Changing the clustering components' properties

You can change which hosts are running the clustering components: Load Manager, Dispatcher, and Cache Manager. You can also change which ports the clustering components will use.

Changing hosts

After creating a cluster profile, you may decide to run the clustering components on different hosts.

➤ **To change the hosts:**

1. In the SMC, select the cluster.
2. Select the **Configuration** icon from the toolbar.
3. Select **Managers**.
4. Update the hosts as needed for the clustering components.
5. Click **Update**.
6. Stop and restart each server and each clustering component in the cluster.

Changing ports

By default, all clustering components use port 54891 for their RMI port. In addition, the Dispatcher uses ports 54892, 54893, and 54894 for its HTTP, RSA, and DSA ports respectively. Normally you don't change these port values unless you want to define unique runtime, design, or administration ports.

But if you need to, you can change the ports after you create the cluster profile.

NOTE All clustering components must use the same RMI port.

➤ **To change the ports:**

1. In the SMC, select the cluster.
2. Select the **Configuration** icon from the toolbar.
3. Select **Managers**.
4. Update the port specifications.
5. Click **Update**.


6. Stop and restart each server and each clustering component in the cluster.

If you changed the port for the Cache Manager, you must start it with the following command line:

```
SilverCacheMgr -p portNumber
```

Installing certificates in a cluster

For a SilverStream cluster to listen and serve on the HTTPS port, you must install a server certificate in each SilverStream server in the cluster. If you are using the SilverStream Dispatcher (**SilverDispatcher**), you also need to install a certificate for it.

 For more information about certificates and HTTPS/SSL, see “Using certificates” on page 226.

About the server certificates

If you are using SilverDispatcher for the cluster, each SilverStream server must have a server certificate with the DNS name matching the **server’s** host name (and, as mentioned, the SilverDispatcher machine must have its own server certificate).

If you are using a third-party hardware dispatcher (such as Cisco LocalDirector) that does URL masking—that is, masks all URLs such that the browser hitting any of the servers in the cluster sees the dispatcher’s host name—then every SilverStream server must have a server certificate with the DNS name matching the **dispatcher’s** host name.

For example, say you have a SilverStream cluster with:

- A Dispatcher with the host name **www.myhost.com**
- Three servers with host names **server1**, **server2**, and **server3**

If you are using SilverDispatcher, you need to create four server certificates: one for the SilverDispatcher machine (with the DNS name **www.myhost.com**) and one for each of the servers (with DNS names **server1.myhost.com**, **server2.myhost.com**, and **server3.myhost.com**).

If you are using a third-party URL-masking dispatcher, you need to create only one server certificate (with the DNS name **www.myhost.com**) and upload it to each of the servers.

How to do it

The following procedures describe exactly what you need to do.

➤ To generate server certificates:

- Use the SMC to generate an RSA or DSA certificate.
 - If you are using SilverDispatcher, generate a different certificate for each server and the SilverDispatcher used in the cluster, as described above.
 - If you are using a third-party dispatcher, generate one certificate with the dispatcher's DNS name, as described above.



For more information, see “About certificates” on page 227.

➤ To install certificates if you are using SilverDispatcher:

1. Invoke **AgDigitalIDStep2** to install the appropriate certificate on a server in the cluster.



For more information, see “Installing the certificate” on page 248.

2. When prompted, specify the server and the HTTP port that the server is listening on (default: 80 in NT, 8080 in UNIX).
3. After the certificate has been installed, restart the server.

The server is now listening on its HTTPS port (default: 443 for RSA certificates and 444 for DSA certificates).

4. Repeat Steps 1 through 3 for each server in the cluster.
5. To install the certificate on SilverDispatcher, start SilverDispatcher using the **-c** startup option. This puts the Dispatcher in the mode in which it can upload a certificate.



For more information, see “Starting the clustering components” on page 338.

6. Invoke **AgDigitalIDStep2** to install the certificate on the machine containing the Dispatcher.



For more information, see “Installing the certificate” on page 248.

7. When prompted, specify the machine containing the SilverStream Dispatcher and specify the HTTP port that the Dispatcher is listening on (default: 54892).
8. After the certificate has been installed, stop the Dispatcher, then restart it as normal (without the **-c** startup option).

The Dispatcher is now listening on its HTTPS port (default: 54893 for RSA certificates and 54894 for DSA certificates).

➤ **To install certificates if you are using a third-party dispatcher:**

1. Invoke **AgDigitalIDStep2** to install on a server the certificate that references the dispatcher's DNS name.



For more information, see “Installing the certificate” on page 248.

2. When prompted, specify the server and the HTTP port that the server is listening on (default: **80** in NT, **8080** in UNIX).
3. After installing the certificate, stop the server.
4. Add the following line to the server's **httpd.props** file (in SilverStream\resources):

```
http-server.com.sssw.srv.https.cert.hostname=DispatcherName
```

where *DispatcherName* is the DNS name of the dispatcher.



For more information about httpd.props, see Appendix A, “The httpd.props File”.

5. Restart the server.
The server is now listening on its HTTPS port (default: 443 for RSA certificates and 444 for DSA certificates).
6. Repeat the preceding steps for each server in the cluster.

Setting up Fulcrum in a cluster

For SilverServer full-text search functionality to work in clustered environment, Fulcrum needs to be configured for the network. You must configure Fulcrum SearchServer so that one of the servers in the cluster acts as the Fulcrum server and all the others are clients. All full-text indexing and searches will be directed to the single SearchServer engine.

Setting up Fulcrum on Windows NT

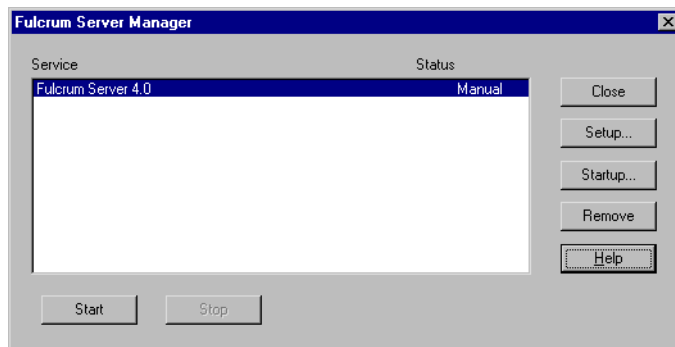
This section includes instructions for configuring the server and the client on Windows NT.

Configuring the server

➤ **To configure the server:**

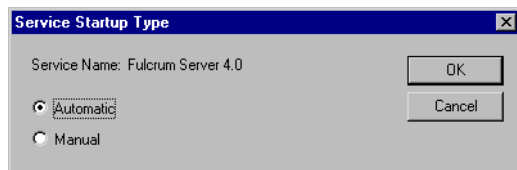
1. Start the Fulcrum Server Manager (found in the Start menu under Programs>SilverStream *version* or **ftsvcadm.exe** in the *SilverStreamInstallDir\Fulcrum\bin* directory).

The following panel displays.



2. Click the **Startup** button.

The following panel displays.



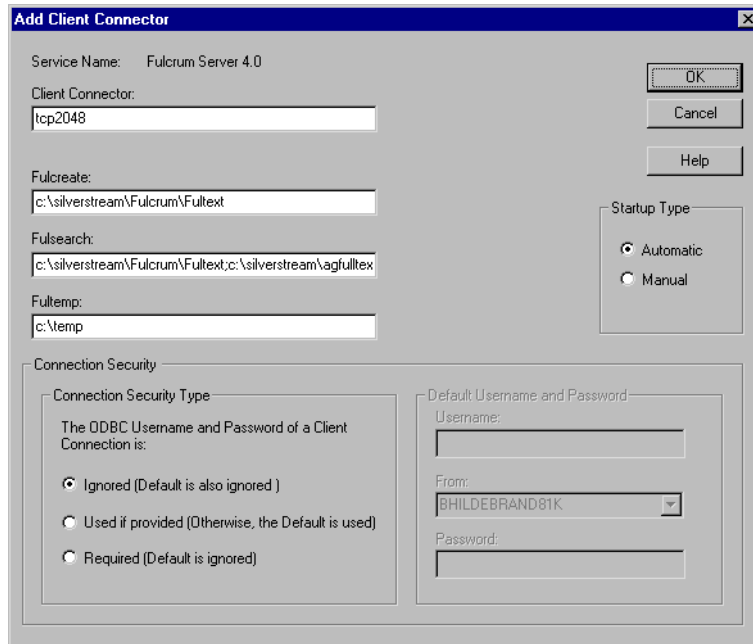
3. Select **Automatic**, then click **OK**.

4. Click the **Setup** button.

The Fulcrum Server Service Setup panel displays.

- Click the **Add** button.

The following panel displays.



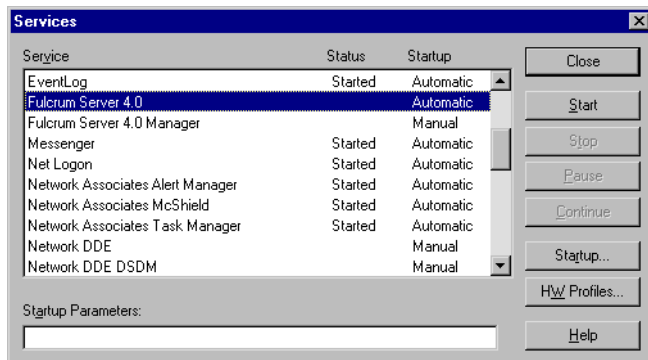
- Fill in the options as described below, then click **OK**.

The description for usage in the following table assumes that the installation directory for SilverStream is c:\silverstream.

Item	Description	Usage
Client Connector	The network connector that allows the server to listen to clients.	Network connector types are: <ul style="list-style-type: none"> TCP/IP connector: syntax is <i>tcpportnumber</i>. This type is recommended for SilverStream. Named Pipes connector: syntax is <i>nmp -pipename</i>.
Fulcreate	The directory where the search tables will be created.	c:\silverstream\Fulcrum\Fultext

Item	Description	Usage
Fulsearch	The directory or directories where the tables will be searched. You must include the directory specified in Fulcreate.	c:\silverstream\Fulcrum\Fultext;c:\silverstream\agfulltext
Fultemp	The location for the temporary files during indexing and searching operations.	c:\temp
Startup type	Startup mode.	Automatic

7. From the Windows NT Start menu, select **Settings>Control Panel>Services**. Verify that the Fulcrum Server 4.0 startup option is **Automatic**.



Configuring the client

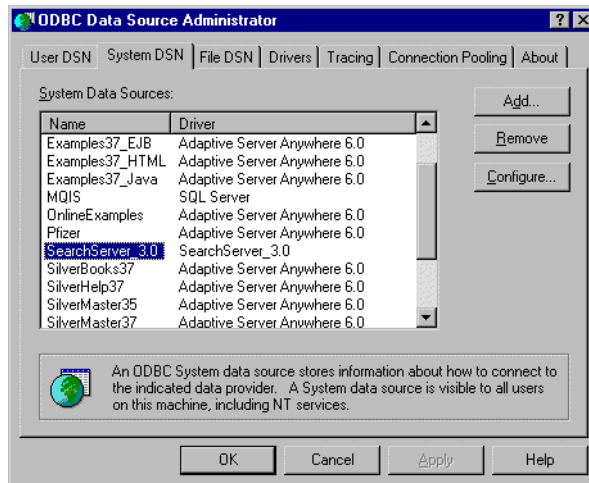
After configuring the server, configure all other machines in the cluster as clients, as follows.

➤ To configure the client:

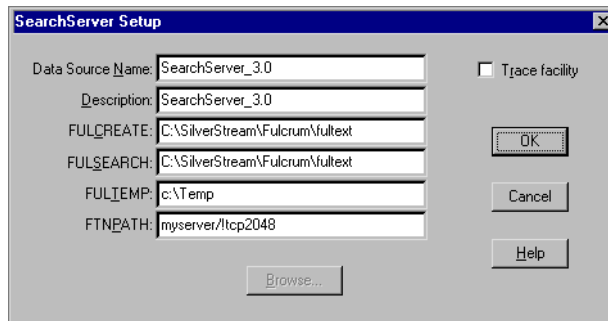
1. From the Windows NT Start menu, select **Settings>Control Panel>ODBC Data Sources**.

The ODBC Data Source Administrator dialog displays.

2. Select the **System DSN** tab.



3. Select the **SearchServer_3.0** entry, then click **Configure**.



4. Change the FTNPATH parameter for the SearchServer_3.0 data source.

For example, if the remote server name is **myserver** and the connectorname is **tcp** and the port parameter for **tcp** is **2048**, you would specify FTNPATH as follows:

```
myserver/!tcp2048
```

The following table describes the syntax for this parameter.

Syntax element	Description
<i>servername</i>	The remote server where SearchServer was configured as a server (see “Configuring the server” on page 360).
!	Tell the server to automatically close any idle connections.
<i>connectorname</i>	The network connector name (case-sensitive). This name should match the Client Connector parameter defined in the server configuration described in “Configuring the server” on page 360.
<i>port</i>	The port associated with the <i>connectorname</i> parameter.

Setting up Fulcrum on UNIX

This section includes instructions for configuring the server and client on UNIX.

➤ To configure the server on UNIX:

1. Log in as **root**.
2. Add an entry in the `/etc/services` file to define **fultext** service and the port number for all Full Text Search Clients. The port number must be greater than or equal to 2048. For example:

```
fultext 2048/tcp # Fulcrum Server
```

3. Add an entry into the `inetd.conf` file for Fulcrum SearchServer program (`ftserver`) to be invoked automatically. For example (the following is a single line entry):

```
fultext stream tcp nowait root /home/silverstream/bin/agftserver
agftserver -uSomeUser
```

- **/home/silverstream** is the SilverStream installation directory.
- **-uSomeUser** sets the default user name for the process to **someUser**. If this option is not given, the process will have **root** as the user.

4. For the above changes to take effect, you must alert the **inetd** process. Typically this is done with the following command:

```
kill -HUP process-id-of-inetd
```

NOTE To find the process ID of inetd, you can run the following command:

```
ps -ef | grep inetd
```

➤ **To configure the client on UNIX:**

- Open the **.agprofile** file in the SilverStream installation directory and add the following line to it:

```
FTNPATH=myserver/tcp2048 ; export FTNPATH
```



For information about FTNPATH, see “Configuring the client” on page 364.

13 Using the Server Administration API

You can configure and manage your SilverStream server using the SilverStream Management Console (SMC), or you can create your own management console (or parts of one) using the Server Administration API. Applications written with this API can be deployed as Java applications running on the server or as Java clients, such as SilverJ2EEClient or an externally developed Java client. The SMC (which uses a Java client) is built with this API.

This chapter describes the Server Administration API:

- Introduction to the Administration API
- How the Administration API is organized
- How server objects are organized
- Getting started with the Administration API
- Common tasks
- Sample code

Before you begin

If you are unfamiliar with the architecture of the SilverStream server, or do not have a working knowledge of SilverStream databases, security, and load balancing, you can find more information about them in this book. For instance:

- Chapter 4, “Data Source Configuration”
- Chapter 9, “Setting Up Security”
- Chapter 12, “Administering a Cluster”

API javadoc

Reference documentation for the Server Administration API is part of the Core API javadoc included in the server’s Core Help.

Use of the API

Though not deprecated in this version of the SilverStream eXtend Application Server, the Server Administration API will likely be replaced in Version 5 of the SilverStream server by the **J2EE Management API**, which will result from JSR-77 and be part of J2EE 1.4. The J2EE Management API will provide a standard model for managing J2EE environments. For more information about the J2EE Management API, see <http://jcp.org/jsr/detail/77.jsp>.

Introduction to the Administration API

The Administration API makes any server-registered object available for configuration or management. A **registered object** is any resource that resides on and is known to the server. Examples of server-registered objects include deployed J2EE archives, SilverStream users and groups, connection pools, load balancing clusters, and classic SilverStream pages, forms, and views.

You can use the Administration API to write applications that let end users, system administrators, or batch processes do a variety of tasks. For example, you can use the API to add connection pools, delete registered objects from the server, or allow end users to change their passwords.

Administering the application server involves:

Task	Description
Managing resources	Managing application resources means managing the components that comprise your application and can include setting security and permissions on those objects. NOTE You manage security in a J2EE application in the application itself, its deployment descriptor, and the runtime deployment information—not by setting security on individual elements on the server.
Managing server performance and configuration	Managing the server's configuration and performance means monitoring the server's runtime environment and changing that environment to improve performance and maintain integrity. The performance and configuration objects include threads, sessions, clusters, and statistics.

The API provides method calls to perform both types of tasks. You can use the API to gain access to objects that represent all of the following:

- Application components, including J2EE archives, tables, and (in classic SilverStream applications) pages, forms, and business objects
- SilverStream, NT, NIS+, and LDAP security objects (including users and groups)
- Certificates
- Load balancing objects
- Database connection objects

The Administration API also provides access to server cache and port-related settings through the `AgiAdmServer` interface.

Client-side versus server-side access to API

The Administration API is available to Java clients and server-based applications. There are some differences in the functionality that is available to you depending on whether your administration console is client-based or server-based. For this reason, it is important to know how you will deploy the application before you get started.

The following functions are restricted to client-side implementation of the Administration API:

- License management
- Certificate management
- Add/remove database
- Synchronize database schema
- Publish
- Logon/logoff
- Enumerate database platforms that are available

About J2EE applications

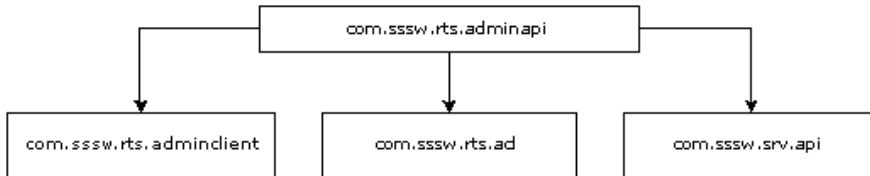
Even though J2EE applications are thought of as running on the server, from the server's point of view they are clients—so you use the client-side implementation of the Administration API with deployed J2EE applications.

How the Administration API is organized

You can easily determine which features or objects can be managed based on the package in which the interface or class that represents that object resides. The Administration API is available through these packages:

Package	Contents
com.sssw.rts.acl	This package contains objects dealing with Access Control Lists (ACLs), principals, and the AgoPermission objects that are used for getting and setting permissions. It is available to both the client- and server-side implementation.
com.sssw.rts.adminapi	This package contains most of the Administration API. The interfaces and objects are implemented by both the adminclient and srv.api packages.
com.sssw.rts.adminclient	This package contains the administration objects that are programmatically available on the client. It includes AgAdmin, the client-side object that provides factory methods for Administration API objects.
com.sssw.srv.api	This package contains the administration objects that are available on the server (that is, in a classic SilverStream page or business object).

The diagram below illustrates how these packages are organized. Note that the `com.sssw.rts.adminapi` package is available to both the client and the server.



How server objects are organized

Before you can work with server-related objects, you should understand more about how the Administration API represents objects and how the server organizes them.

The SilverStream server contains two types of objects: objects that can contain other objects (**containers**) and objects that cannot (**elements**). The server organizes these objects in a hierarchy within each SilverStream database.

A single SilverStream server (represented by the `AgiAdmServer` object) can contain multiple databases. Every SilverStream database (represented by `AgiAdmDatabase` objects) contains multiple directories (represented by `AgiAdmDirectory` objects). Servers, databases, and directories are all containers.

Because an `AgiAdmDirectory` object is a container, it can contain other directory objects as well as single elements. The types of elements that a directory can contain depends on the directory. For example:

Directory	Contents
Security	Directories for each kind of user (<code>SILVER_SECURITY</code> , <code>NT_SECURITY</code> , and so on), each of which ultimately contains users (that is, objects of type <code>AgiAdmUser</code>)
Pages	Pages (classic SilverStream pages, which are objects of type <code>AgiAdmDesignElement.PAGE</code>)

The object hierarchy

Here is how the objects are structured on the SilverStream server:

```
AgiAdmServer.SERVER
  AgiAdmDatabase.DATABASE
    AgiAdmDirectory.APP_OBJECTS
      AgiAdmDirectory.PACKAGE
        AgiAdmDesignElement.APP_OBJECT
    AgiAdmDirectory.PACKAGE
      AgiAdmDirectory.PACKAGE
        AgiAdmDesignElement.APP_OBJECT
    AgiAdmDirectory.FORMS
      AgiAdmDesignElement.FORM
    AgiAdmDirectory.TABLES
      AgiAdmDesignElement.TABLE
    AgiAdmDirectory.VIEWS
      AgiAdmDesignElement.VIEW
    AgiAdmDirectory.MEDIA
      AgiAdmDirectory.GENERAL
        AgiAdmDesignElement.GENERAL_ELEMENT
      AgiAdmDirectory.IMAGES
        AgiAdmDesignElement.IMAGE
      AgiAdmDirectory.JARS
        AgiAdmDeployedObject.DEPLOYED_OBJECT
        AgiAdmDesignElement.JAR
      AgiAdmDirectory.SOUNDS
        AgiAdmDesignElement.SOUND
    AgiAdmDirectory.PAGES
      AgiAdmDesignElement.PAGE
    AgiAdmDirectory.DIRECTORY
      AgiAdmDirectory.DIRECTORY
        AgiAdmDesignElement.SERVLET
    AgiAdmDirectory.SECURITY (see next section)
```

The security objects

As shown in the preceding hierarchy, there is a Security directory on the server. It has its own hierarchy of objects that are related to the security realms that SilverStream supports. Here is the Security hierarchy:

```
AgiAdmDirectory.SECURITY
  AgiAdmDirectory.SILVER_SECURITY
    AgiAdmDirectory.SILVERGROUPS
      AgiAdmGroup.SILVERGROUP
        AgiAdmUserReference.USER_REFERENCE
  AgiAdmDirectory.SILVERUSERS
    AgiAdmUser.SILVERUSER
  AgiAdmDirectory.NT_SECURITY
    AgiAdmDirectory.DOMAIN
      AgiAdmDirectory.NTGROUPS
        AgiAdmGroup.NTGROUP
          AgiAdmUserReference.USER_REFERENCE
      AgiAdmDirectory.NTUSERS
        AgiAdmUser.NTUSER
  AgiAdmDirectory.LDAP_SECURITY
    AgiAdmDirectory.LDAP_SERVER
      AgiAdmDirectory.LDAPUSERS
        AgiAdmUser.LDAPUSER
      AgiAdmDirectory.LDAPGROUPS
        AgiAdmGroup.LDAPGROUP
          AgiAdmUserReference.USER_REFERENCE
    AgiAdmDirectory.NISPLUS_SECURITY
      AgiAdmDirectory.NISPLUS_SERVER
        AgiAdmDirectory.NISPLUSUSERS
          AgiAdmUser.NISPLUSUSER
        AgiAdmDirectory.NISPLUSGROUPS
          AgiAdmGroup.NISPLUSGROUP
            AgiAdmUserReference.USER_REFERENCE
    AgiAdmDirectory.CERTIFICATE_SECURITY
      AgiAdmUser.CERTIFICATEUSER
      AgiAdmGroup.WORLD
```

The load balancing object hierarchy

The load balancing objects are not contained within the database/directory hierarchy as are the other server containers and elements. However, they do follow the same container/element paradigm.

```
AgiAdmLBClusterEnv.CLUSTER_ENV
AgiAdmCluster.CLUSTER
  AgiAdmClusterServer.CLUSTER_SERVER
AgiAdmLBContainer.CACHE_MGR_GROUP
  AgiAdmLBElement.CACHE_MGR
AgiAdmLBContainer.LOAD_MGR_GROUP
  AgiAdmLBElement.LOAD_MGR
AgiAdmLBContainer.DISP_GROUP
  AgiAdmLBElement.DISPATCHER
```

More about containers and elements


Elements

All server objects are elements. A database is an element, a load-balancing cluster is an element, a SilverStream user is an element, and so on.

All server objects (except the load-balancing objects) implement the `AgiAdmElement` interface, which means you can perform similar functions on all objects. For example, you can get the object's name and its type, and you can get and set permissions for it.

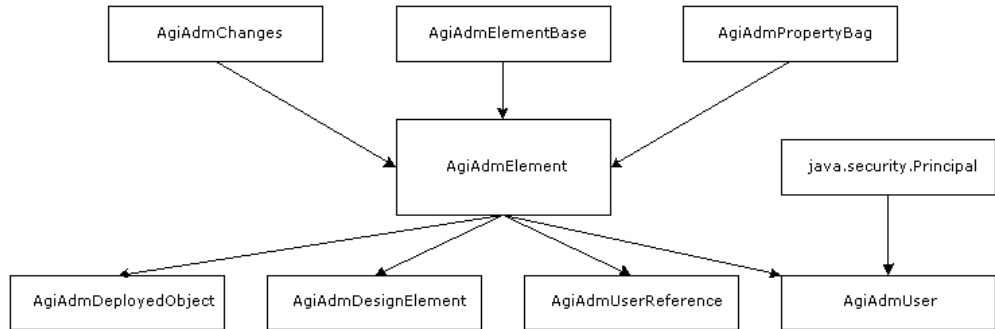
In terms of application components, deployed J2EE archives implement `AgiAdmDeployedObject`. Components in classic SilverStream applications, such as pages and forms, implement `AgiAdmDesignElement`.

If you have a server object, you can obtain any other object by calling `AgiAdmServer.getElement()`.

 For information on obtaining a server object, see “Getting started with the Administration API” on page 376.

Element hierarchy

This diagram illustrates the hierarchy for the interface objects that represent elements on the server.



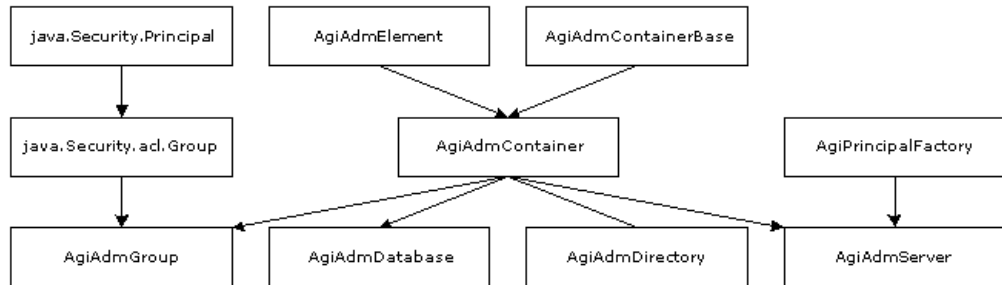
Elements and containers

Some elements are also containers. A container is an element that can contain (or can parent) other elements. The server elements that are also containers include servers, databases, directories, groups, and load-balancing containers. Containers implement the `AgiAdmContainer` interface.

You can perform a specific set of functions on container elements, such as enumerating the children of the container, obtaining a specific child, and adding children.

Container hierarchy

This diagram illustrates the container hierarchy.



Getting started with the Administration API

Your first programming task, before you have access to any server objects, is to obtain a reference to a server object (AgiAdmServer for client-based applications or AgiServer for server-based applications). Once you have the server object, you can obtain and manipulate any of its elements.

Application type	Task
Client-based applications	Obtain the server using the AgAdmin.getServer() method
J2EE applications	Obtain the server using the AgAdmin.getServer() method
Applications containing classic SilverStream pages and business objects	Obtain the server from an event object

You do not need a server object if you intend to work with the publish manager (AgiAdmPublishMgr) or the expression manager (AgiAdmExpressionMgr).

Obtaining a server object from a client application

You can obtain a server reference by calling the `AgAdmin.getServer()` method with three variants. You need the following imports:

```
import com.sssw.rts.adminapi.*;
import com.sssw.rts.adminclient.*;
```

Variant 1

One variant lets you specify the server name and its port:

```
AgiAdmServer getServer(String serverName, int port)
    throws AgoUnrecoverablSystemException
```

where *serverName* is the name of the server and *port* is the port to use (specify -1 to use the default port).

For example:

```
AgiAdmServer srv = AgAdmin.getServer("myserver", 90);
```

Variant 2

The second variant also lets you specify either the HTTP or HTTPS protocol.

```
AgiAdmServer getServer(String protocol, String serverName, int port)
    throws AgoUnrecoverableSystemException
```

For example:

```
AgiAdmServer srv = AgAdmin.getServer("https", "myserver", 88);
```

Variant 3

The third variant lets you specify the server, its port, and the protocol as one string:

```
AgiAdmServer getServer(String server)
    throws AgoUnrecoverableSystemException
```

For example:

```
AgiAdmServer srv = AgAdmin.getServer("https://myserver:8080");
```

Using `getServer`

The `getServer()` method returns an object of type `AgiAdminServer`. Once you have this object, you can get its properties, enumerate its objects (containers and/or elements), and manipulate the elements.

Obtaining a server object from a J2EE application

Even though J2EE applications are thought of as running on the server, from the server's point of view they are clients. So you use `AgiAdmServer` to get the server instance.

From a servlet

You can get a `SilverStream` server object in a servlet in two ways:

- From the **request object**:

```
private AgiAdmServer getServer(HttpServletRequest req)
    throws AgoUnrecoverableSystemException
{
    try
    {
        AgoHttpRequestEvent reqEvent = (AgoHttpRequestEvent) req;
        return (AgiAdmServer) reqEvent.getServer();
    } catch(Exception e) {

        e.printStackTrace();
        throw new AgoUnrecoverableSystemException(e.toString());
    }
}
```

- From the **servlet context**:

```
public void init( ServletConfig config )
    throws ServletException
{
    ...
    // Get the server
    AgoServletContext servletContext =
        (AgoServletContext) config.getServletContext();
    AgiAdmServer srv = servletContext.getServer();
    ...
}
```

From an EJB

You can get a server object from the EJB's session context object:

```
protected SessionContext m_context;  
private AgiAdmServer m_server;  
m_server = (( AgiEJBContext ) m_context).getServer();
```

Obtaining a server object from a classic SilverStream page or business object

An initialized instance of `AgiServer` is passed to server-side objects using the `getServer()` method on the event object passed to the page or business object.

From a page


Code the following:

```
AgoHttpRequestEvent evt = (AgoHttpRequestEvent) getCurrentRequest();  
AgiServer srv = evt.getServer();
```

From a business object

Code the following:

```
AgiServer srv = evt.getServer();
```

 For more information on obtaining objects (such as an `AgiServer`) from an event object, see the triggered object basics chapter in the *Programmer's Guide* of the SilverStream server's Classic Development Help (Classic Edition only).

Getting a server's properties

Once you have the `AgiAdmServer` or the `AgiServer` object, you can access any of the server's properties. This example shows how you can get two server properties using the client-side API (`AgAdmin`).

First make sure you have the proper imports:

```
import com.sssw.rts.adminapi.*;  
import com.sssw.rts.adminclient.*;
```

The following code gets two properties—the name of the SilverMaster database and the DSA port—then displays the values in text fields on a form.

```
try {
    AgiAdmServer srv = AgAdmin.getServer("localhost",80);

    // Get the name of the SilverMaster database
    String sSMaster =
        (String) srv.getProperty(srv.PROP_SILVERMASTER);

    // Get the DSA port used by the server
    Integer iDSAport =
        (Integer) srv.getProperty(srv.PROP_HTTPS_PORT_DSA_RT);

    // Display the values in text fields
    Field1.setText(sSMaster);
    Field2.setText(iDSAport.toString());
}

catch (Exception e) {
    System.out.println(e);
}
```

You can use the same technique on the server, except you would obtain the server object using the server-side API. Instead of importing **com.sssw.rts.adminclient.***, you would import **com.sssw.srv.api.***.

Working with server elements

Once you have the `AgiAdmServer` or the `AgiServer` object, you can access any of its elements. There are several ways that you can obtain and manipulate those elements:

- You can **enumerate all of the elements and containers** that reside on the server and use the enumeration to obtain the specific element(s) that you might want to work with. This is useful when you want to work with groups of objects.
- You can **get a specific element using the `AgiAdmServer.getElement()` method**. This option is a shortcut to obtaining a specific element and is useful when you want only a single element.
- You can **use the parent-child relationship** of the containers and elements with the appropriate method.

Enumerating database elements

This example illustrates how to obtain a server object, enumerate the databases that reside on the server, and enumerate the elements within each database. It shows how to obtain the server object using the client-side API (AgAdmin). You can use the same technique on the server, except you would obtain the server object using the server-side API.

First make sure you have the proper imports:

```
import com.sssw.rts.adminapi.*;
import com.sssw.rts.adminclient.*;
```

Next get the server and enumerate the database elements:

```
try {
    // Get a server and enumerate the databases within it
    AgiAdmServer server = AgAdmin.getServer("dg.myHome.com", 90);
    Enumeration databases =
        server.getChildren(AgiAdmContainer.GET_CHILDREN_SORTED);

    // For each database in the enumeration, get to a database
    while(databases.hasMoreElements()) {
        AgiAdmDatabase db =
            (AgiAdmDatabase)databases.nextElement();

        // Enumerate the 'well-known' directories within the database
        Enumeration directories =
            db.getChildren(AgiAdmContainer.GET_CHILDREN_SORTED);

        // For each directory enumerate the design elements
        while(directories.hasMoreElements()) {
            AgiAdmDirectory dir =
                (AgiAdmDirectory)directories.nextElement();
            Enumeration desElements =
                dir.getChildren(AgiAdmContainer.GET_CHILDREN_SORTED);
            // For each design element, you can do something with it...
            while(desElements.hasMoreElements())
            {
                AgiAdmDesignElement desEl =
                    (AgiAdmDesignElement)desElements.nextElement();
                // Perform an action on the design element here
            }
        }
    }
}
catch (Exception e) {
    System.out.println(e);
}
```

Notes about the code:

1. Obtain a reference to the server dg.myHome.com at port 90.
2. Enumerate the application databases that reside on the server.
3. For each database, enumerate its directories.
4. After you have the enumeration of the objects, you can manipulate the design element, such as setting permissions for it or deleting it.

Obtaining elements using getElement()

The `getElement()` method obtains an object with the specified name and type. It provides a shortcut for instantiating new server objects. This is the declaration:

```
AgiAdmElementBase getElement(String name, String type,  
                             Hashtable info) throws AgoUnrecoverableSystemException,  
                             AgoSecurityException
```

Parameters

The *name* parameter specifies the name of the element (for example, a database table named **Customers**, a page named **HomePage**, a form named **myForm**, and so on).

The *type* parameter specifies the object type of the element that you want to obtain, including the constant. (For example, to obtain an element whose type is form, you specify **AgiAdmDesignElement.FORM**.)

There are types for each kind of known element, including:

- Deployed J2EE archives (AgiAdmDeployedObject)
- Classic design elements such as SilverStream pages and forms (AgiAdmDesignElement)
- Directory elements (AgiAdmDirectory)
- Group elements (AgiAdmGroup)
- User elements (AgiAdmUser)
- Load balancing container or elements (including: AgiAdmLBClusterEnv, AgiAdmLBElement, and AgiAdmLBContainer).

You use a constant to specify its type.

The *info* parameter is a hashtable that contains additional pieces of information that are necessary to identify the object you want to obtain via `getElement()`. You use the `put()` method on a hashtable to insert additional information into the *info* parameter.

The *info* parameter can be one of these constants:

Constant	When required
INFO_PORT_NUMBER	The port number is required when you are obtaining an element of type AgiAdmServer.SERVER
INFO_DATABASE_NAME	The database name is required when you are obtaining an element of type AgiAdmDirectory or AgiAdmDesignElement
INFO_GROUP_NAME	The group name is required when you are obtaining an element type of AgiAdmUserReference

This example illustrates how to obtain a server element of type AgiAdmDatabase where the database name is Montreal and the element type is AgiAdmDatabase.DATABASE. The *info* parameter is specified as **null**, because the AgiAdmDatabase.DATABASE does not require it.

```

AgiAdmServer server = AgAdmin.getServer("myserver", 80);
AgiAdmDatabase myDB = (AgiAdmDatabase) server.getElement("Montreal",
    AgiAdmDatabase.DATABASE, null);

```

More about the info parameter The following table lists additional information about item identifiers that you can use with the info parameter and the elements to which they are related.

Information item identifier	Elements used by
AgiAdmServer.INFO_DATABASE_NAME	AgiAdmDesignElement, AgiAdmDirectory
AgiAdmServer.INFO_GROUP_NAME	AgiAdmUserReference
AgiAdmServer.INFO_PORT_NUMBER	AgiAdmServer.SERVER
AgiAdmElement.PROP_CERTIFICATE	AgiAdmUser.CERTIFICATEUSER
AgiAdmElement.PROP_DOMAIN	AgiAdmUser.NTUSER, AgiAdmGroup.NTGROUP, AgiAdmDirectory.NTUSERS, AgiAdmDirectory.NTGROUPS
AgiAdmElement.PROP_FULL_NAME	AgiAdmUser.SILVERUSER

Information item identifier	Elements used by
AgiAdmElement.PROP_PARENT_URL	AgiAgmDesignElement.APP_OBJECT, AgiAdmDirectory.PACKAGE, AgiAdmDirectory.DIRECTORY, AgiAgmDesignElement.SERVLET
AgiAdmElement.PROP_PASSWORD	AgiAdmUser.SILVERGROUP
AgiAdmElement.PROP_QUAL_NAME	AgiAdmUser, AgiAdmGroup
AgiAdmElement.PROP_LDAP_SERVER	AgiAdmDirectory.LDAPGROUPS, AgiAdmDirectory.LDAPUSERS, AgiAdmUser.LDAPUSER, AgiAdmGroup.LDAPGROUP
AgiAdmElement.PROP_NISPLUS_SERVER	AgiAdmDirectory.NISPLUSGROUPS, AgiAdmDirectory.NISPLUSUSERS, AgiAdmUser.NISPLUSUSER, AgiAdmUser.NISPLUSGROUP

This example shows how you can put the port number value into a hashtable and pass it to the `AgiAdmServer.getElement()` method on the *info* parameter:

```
Hashtable info = new Hashtable();
info.put(AgiAdmServer.INFO_PORT_NUMBER, new Integer(92));
AgiAdmServer anotherServer=
(AgiAdmServer) myServer.getElement("redhook",AgiAdmServer.SERVER,
info);
```

This example shows how you can use a hashtable to populate the *info* parameter with data for the `INFO_GROUP_NAME`:

```
Hashtable info=new Hashtable();
info.put(AgiAdmServer.INFO_GROUP_NAME, "Administrators");
AgiAdmUserReference uref=
(AgiAdmUserReference) server.getElement("admin1",
AgiAdmUserReference.USER_REFERENCE, info);
```


Obtaining child elements

Many objects on the server are parents or children of other elements (or containers). This container relationship is useful for obtaining a reference to an object.

As you saw earlier in the chapter, the server is the parent of all objects on the server. Any SilverStream database is a child of the server. You can use the `AgiAdmContainer.getChild()` method to obtain a single child of a particular container, or the `AgiAdmContainer.getChildren()` to enumerate all of the children.

The `getChild()` method has this declaration:

```
AgiAdmElementBase getChild(String name, String type,
                           Hashtable info) throws AgoUnrecoverableSystemException,
                           AgoSecurityException
```

where *name* is the name of the child, *type* is the object type of the child, and *info* includes any additional properties describing the child.

This example shows how to obtain a reference to the server, obtain a database named `myDatabase`, and obtain the directory of FORM objects:

```
// get a reference to AgiAdmServer
AgiAdmServer server = ...

// get database (a child of server)
AgiAdmDatabase database = (AgiAdmDatabase)server.getChild(
    "myDatabase", AgiAdmDatabase.DATABASE, null);

// get the FORMS directory (a child of database)
Hashtable info = new Hashtable();
info.put(AgiAdmServer.INFO_DATABASE_NAME, "myDatabase");
AgiAdmDirectory formsDir = (AgiAdmDirectory)database.getChild(
    (AgiAdmDirectory.FORMS, AgiAdmDirectory.FORMS, info);
```

You can call `getChild()` on any container-type objects.

Common tasks

Now that you have seen the packages that contain the Administration API and the ways that objects are referenced on the server, you might be wondering how to start writing applications that use these classes, or wondering where to start.

The following table lists some common tasks and the interfaces you can use to implement them.

Task	Interfaces to use
Load balancing configuration	AgiAdmLBClusterEnv, AgiAdmLBContainer, AgiAdmLBElement
Publish	AgiAdmPublishMgr
Set security on objects	AgiAdmExpressionMgr (and the interfaces that let you obtain the object, for example, AgiAdmDesignElement)
Manage deployed J2EE archives	AgiAdmDeployedObject
Work with classic design elements such as SilverStream pages	AgiAdmDesignElement
Manage server sessions	AgiAdmSession
Manage mail accounts	AgiAdmMailAccount
Manage certificates	AgiAdmCertificate
Manage SilverStream licenses	AgiAdmLicense
Manage threads	AgiAdmThreads
Manage statistics	AgiAdmStatistics, AgiAdmStatSet

Sample code

This section provides the following examples of using the Administration API to manage your server:

- Enumerating security providers, users, and groups
- Identifying users
- Customizing the logging class

Enumerating security providers, users, and groups

This sample enumerates security providers as well as the users and groups defined by providers. You'll learn about:

- Enumerating security providers
- Enumerating provider servers
- Enumerating groups
- Enumerating users in a group

Enumerating security providers

The following code loops through the objects known to the current server looking for security directories and loads the results into a tree control. (For a description of how to get the current server, see “Getting started with the Administration API” on page 376).

```
public void loadUsersAndGroups()
{
/**
 * Method:
 *   loadUsersAndGroups()
 * Description:
 *   Populate a tree control specified with users and groups from the
 *   server specified.
 * Parameters:
 *   None.
 * Returns:
 *   Nothing.
 */

myTreeControl tcObjects;
myTreeControlNode nodeChild;
Boolean value = new Boolean(false);
String type, securityName;
Image treeImage=null;
```

```
// Clear the tree control
tcObjects.removeAll();

// Get the security providers for the server.
try
{
    // Get the security provider directory object.
    AgiAdmDirectory dir =
        (AgiAdmDirectory) m_server.getElement(AgiAdmDirectory.SECURITY,
        AgiAdmDirectory.SECURITY, null);

    // Get the children of the directory.
    Enumeration children =
        dir.getChildren(AgiAdmContainer.GET_CHILDREN_SORTED);

    if (children != null)
    {
        while(children.hasMoreElements())
        {
            // Get element from the directory.
            // If the element is a AgiAdmDirectory add it to the tree control.
            AgiAdmElement element = (AgiAdmElement)children.nextElement();

            if (element instanceof AgiAdmDirectory)
            {
                AgiAdmDirectory child = (AgiAdmDirectory)element;

                // Get the server name.
                securityName = "";
                type = child.getName();

                if (type.equals(AgiAdmDirectory.SILVER_SECURITY))
                {
                    // Add the SilverStream security.
                    securityName = "Silver Security";
                    treeImage = getSilverSecurityImage();
                }
                else if (type.equals(AgiAdmDirectory.NT_SECURITY))
                {
                    // Add the NT security.
                    securityName = "NT Security";
                    treeImage = getNTImage();
                }
                else if (type.equals(AgiAdmDirectory.LDAP_SECURITY))
                {
                    // Add the LDAP security.
                    securityName = "LDAP Security";
                    treeImage = getLDAPImage();
                }
            }
        }
    }
}
```

```
else if (type.equals(AgiAdmDirectory.NISPLUS_SECURITY))
{
    // Add the NIS+ security.
    securityName = "NIS+ Security";
    treeImage = getNISImage();
}

Hashtable info = loadUserData(type, true);
info.put("Data", element);

// Add the Security node.
nodeChild = tcObjects.add(null, myTreeControl.NEXT,
    securityName, info, treeImage);

if (type.equals(AgiAdmDirectory.SILVER_SECURITY))
{
    // Get the children of the directory.
    Enumeration groupuser =
        child.getChildren(AgiAdmContainer.GET_CHILDREN_SORTED);

    if (groupuser != null)
    {
        while (groupuser.hasMoreElements())
        {
            // Get element from the directory. If the element is
            // a AgiAdmGroup add it to the tree control.
            AgiAdmElement subdir =
                (AgiAdmElement)groupuser.nextElement();
            String dirType = subdir.getType();

            if (dirType.equals(AgiAdmDirectory.SILVERGROUPS))
            {
                // Add the SilverGroups
                info = loadUserData(AgiAdmDirectory.SILVERGROUPS, true);
                info.put("Data", subdir);
                m_agGroups = tcObjects.add(nodeChild, myTreeControl.CHILD,
                    "Groups", info, getGroupsImage());
                tcObjects.add(m_agGroups, myTreeControl.CHILD,
                    "AgEmptyGroup");
            }

            if (dirType.equals(AgiAdmDirectory.SILVERUSERS))
            {
                // Add the SilverUsers
                info = loadUserData(AgiAdmDirectory.SILVERUSERS, true);
                info.put("Data", subdir);
                m_agUsers = tcObjects.add(nodeChild, myTreeControl.CHILD,
                    "Users", info, getUsersImage());
            }
        }
    }
}
```



```
* Parameters:
*   String type
* Returns:
*   boolean
*/
myTreeControl tcObjects;
// Get the selected node and delete any children.
myTreeControlNode nodeDomains = tcObjects.getSelectedNode();

if (nodeDomains != null) removeChildren(nodeDomains);

// Get the servers for the specified type.
try
{
    // Get the directory object for the servers requested.
    AgiAdmDirectory dir = (AgiAdmDirectory) m_server.getElement(type,
        type, null);

    // Get the children of the directory.
    Enumeration children =
        dir.getChildren(AgiAdmContainer.GET_CHILDREN_SORTED);

    if (children != null)
    {
        while(children.hasMoreElements())
        {
            // Get element from the directory. If the element is a
            // AgiAdmDirectory, add it to the tree control.
            AgiAdmElement element = (AgiAdmElement) children.nextElement();

            if (element instanceof AgiAdmDirectory)
            {
                AgiAdmDirectory child = (AgiAdmDirectory)element;

                // Get the server name.
                String serverName = child.getName();

                Hashtable info = new Hashtable();
                String server, group, user;

                if (type.equals(AgiAdmDirectory.NT_SECURITY)) // NT Server.
                {
                    server = AgiAdmDirectory.DOMAIN;
                    group = AgiAdmDirectory.NTGROUPS;
                    user = AgiAdmDirectory.NTUSERS;
                }
                else if (type.equals(AgiAdmDirectory.LDAP_SECURITY)) // LDAP Server
                {
```

```
server = AgiAdmDirectory.LDAP_SERVER;
group = AgiAdmDirectory.LDAPGROUPS;
user = AgiAdmDirectory.LDAPUSERS;
}
else
{
server = AgiAdmDirectory.NISPLUS_SERVER; // NIS+ Server
group = AgiAdmDirectory.NISPLUSGROUPS;
user = AgiAdmDirectory.NISPLUSUSERS;
}

// Load up the hashtable with the server info.
info.put("Type", server);
info.put("Retrieve", new Boolean(true));
info.put("Data", element);

// Add the node
myTreeControlNode nodeDomain =
    tcObjects.add(nodeDomains, myTreeControl.CHILD, serverName,
        info, m_imgServer);

myTreeControlNode nodeChild;

// Get the children of the directory.
Enumeration groupuser =
    child.getChildren(AgiAdmContainer.GET_CHILDREN_SORTED);

if (groupuser != null)
{

while(groupuser.hasMoreElements())
{

// Get element from the directory. If the element is a
// AgiAdmGroup, add it to the tree control.
AgiAdmElement subdir = (AgiAdmElement)groupuser.nextElement();
String dirType = subdir.getType();

info = new Hashtable();
info.put("Retrieve", new Boolean(true));
info.put("Directory", new Boolean(false));
info.put("Data", subdir);

// Add the group node.
if (dirType.indexOf("Groups") >= 0)
{
info.put("Type", group);
nodeChild =
    tcObjects.add(nodeDomain, myTreeControl.CHILD, "Groups",
        info, m_imgGroups);
}
```



```

        tcObjects.add(nodeChild,myTreeControl.CHILD,serverName +
            "EmptyGroup");
    }

    // Add the user node.
    if (dirType.indexOf("Users") >= 0)
    {
        info.put("Type", user);
        nodeChild =
            tcObjects.add(nodeDomain,myTreeControl.CHILD,"Users",
                info,m_imgUsers);
        tcObjects.add(nodeChild,myTreeControl.CHILD,serverName +
            "EmptyUser");
    }
    }
    }
    }
    }
}

// Reset the retrieve of the user data to false.
Hashtable userData = (Hashtable) nodeDomains.getUserData();
userData.put(RETRIEVE, new Boolean(false));

// Set the user data so we don't to retrieve next time around.
if (nodeDomains != null) nodeDomains.setUserData(userData);

// Select the node.
if (nodeDomains != null) tcObjects.setSelectedNode(nodeDomains);

return true;
}
catch (Exception e)
{
    myDialog.displayError(e);
    return false;
}
}
}

```

Notes about the code

- The code uses `getElement()` to get the security directory type, which is passed in from the parent method. The result is cast to an object of type `AgiAdmDirectory`. Then `getChildren()` is used to enumerate and sort the result in ascending order.

- The code uses the Java method `nextElement()` to loop through the enumeration of `getChildren()`. The result is cast to an object of type `AgiAdmElement`, then checked for elements of type `AgiAdmDirectory`.
- For each element of type `AgiAdmDirectory`, the code uses the API to sort for security type directories passed to the parent method `getServers()`, and sets appropriate values for each item. For example, for type `NT_SECURITY` the server object is obtained from `AgiAdmDirectory.DOMAIN`, and the groups known to that server from `AgiAdmDirectory.NTGROUPS`.

Enumerating groups

The following code takes the directory type and server and enumerates groups known to each security provider. Each group is then loaded into a tree control.

```
public boolean getGroups(String type, String server)
{
    /**
     * Method:
     *   getGroups()
     * Description:
     *   Get a the list of groups for a specific type.
     *
     * Parameters:
     *   String type
     *   String server
     * Returns:
     *   boolean
     */
    myTreeControl tcObjects;
    // Get the selected node.
    myTreeControlNode nodeGroups = tcObjects.getSelectedNode();
    if (nodeGroups != null) removeChildren(nodeGroups);

    //Get the groups for the type that was passed in.
    try
    {
        Hashtable info = new Hashtable();
        String childType;

        if (type.equals(AgiAdmDirectory.NTGROUPS)) //NT groups
        {
            childType = AgiAdmGroup.NTGROUPE;
            info.put(AgiAdmElement.PROP_DOMAIN, server);
        }
        else if (type.equals(AgiAdmDirectory.LDAPGROUPS)) //LDAP groups
        {
            childType = AgiAdmGroup.LDAPGROUP;
        }
    }
}
```

```
        info.put(AgiAdmElement.PROP_LDAP_SERVER, server);
    }
else if (type.equals(AgiAdmDirectory.NISPLUSGROUPS)) //NIS+ groups
{
    childType = AgiAdmGroup.NISPLUSGROUP;
    info.put(AgiAdmElement.PROP_NISPLUS_SERVER, server);
}
else
{
    childType = AgiAdmGroup.SILVERGROUP;
    info = null; //SilverStream groups
}

// Get the directory object for groups.
AgiAdmDirectory dir = (AgiAdmDirectory) m_server.getElement(type,
    type, info);

// Get the children of the directory.
Enumeration children =
    dir.getChildren(AgiAdmContainer.GET_CHILDREN_SORTED);

if (children != null)
{
    while(children.hasMoreElements())
    {
        // Get element from the directory. If the element is a AgiAdmGroup,
        // add it to the tree control.
        AgiAdmElement element = (AgiAdmElement)children.nextElement();

        if (element instanceof AgiAdmGroup)
        {
            // Add the node to the tree.
            AgiAdmGroup child = (AgiAdmGroup)element;
            String group = child.getName();

            // Load the user data hashtable.
            Hashtable userInfo = loadUserData(childType, true);

            // Put the AgiAdmUser object in the user data.
            userInfo.put("Data", child);

            myTreeNode nodeChild =
                tcObjects.add(nodeGroups, myTreeControl.CHILD, group,
                    userInfo, null);

            // Add a dummy node for group users.
            tcObjects.add(nodeChild, myTreeControl.CHILD, "GroupUser");
        }
    }
}
```

```
    }

    // Reset the retrieve of the user data to false.
    info = (Hashtable) nodeGroups.getUserData();
    info.put(RETRIEVE, new Boolean(false));

    // Set the user data for groups not to retrieve next time around.
    if (nodeGroups != null) nodeGroups.setUserData(info);

    // Select the groups node.
    if (nodeGroups != null) tcObjects.setSelectedNode(nodeGroups);

    return true;
}
catch (Exception e)
{
    myDialog.displayError(e);
    return false;
}
}
```

Notes about the code

- Using a hashtable, the code checks the type parameter passed to the parent method and checks for groups specific to each type of security directory. It then sets appropriate values. For example, for Windows NT security, the type is defined in NTGROUP, and the server in PROP_DOMAIN. Since the server is already known for SilverStream groups, the server value is null.

Enumerating users in a group

The following code takes the security directory type, server, and group and enumerates the users known to each group. It loads the result into a tree control.

```
public boolean getGroupUsers(String type, String server, String group)
{
    /**
     * Method:
     *   getGroupUsers()
     * Description:
     *   Get a the list of users for a group.
     *
     * Parameters:
     *   String type
     *   String server
     *   String group
     * Returns:
```

```

*    boolean
*/
myTreeControl tcObjects;
// Get the selected node and delete any children.
myTreeControlNode nodeGroupUsers = tcObjects.getSelectedNode();
if (nodeGroupUsers != null) removeChildren(nodeGroupUsers);

try
{
    Hashtable info = new Hashtable();
    String groupUser;
    if (type.equals(AgiAdmGroup.NTGROUP)) //NT groups
    {
        groupUser = NTGROUPUSER;
        info.put(AgiAdmElement.PROP_DOMAIN, server);
    }
    else if (type.equals(AgiAdmGroup.LDAPGROUP)) //LDAP groups
    {
        groupUser = LDAPGROUPUSER;
        info.put(AgiAdmElement.PROP_LDAP_SERVER, server);
    }
    else if (type.equals(AgiAdmGroup.NISPLUSGROUP)) //NIS+ groups
    {
        groupUser = NISPLUSGROUPUSER;
        info.put(AgiAdmElement.PROP_NISPLUS_SERVER, server);
    }
    else
    {
        groupUser = SILVERGROUPUSER;
        info = null; //SilverStream groups.
    }
    // Get the SilverGroup object.
    AgiAdmGroup groupUsers = (AgiAdmGroup)
        m_server.getElement(group, type, info);

    // Get the children of the directory.
    Enumeration children =
    groupUsers.getChildren(AgiAdmContainer.GET_CHILDREN_SORTED);

    if (children != null)
    {
        while(children.hasMoreElements())
        {
            // Get element from the group object.  If the element is a
            // AgiAdmUserReference, add it to the tree control.
            AgiAdmElement element = (AgiAdmElement)children.nextElement();

            if (element instanceof AgiAdmUserReference)
            {

```

```
        AgiAdmUserReference child = (AgiAdmUserReference)element;
        info = loadUserData(groupUser, false);
        info.put("Data", element);
        tcObjects.add(nodeGroupUsers, myTreeControl.CHILD,child.getName(),
            info, null);
    }
}

// Reset the retrieve of the user data to false.
if (nodeGroupUsers != null)
{
    info = (Hashtable) nodeGroupUsers.getUserData();
    info.put(RETRIEVE, new Boolean(false));

    // Set the user data for SilverGroups not to retrieve next time around.
    nodeGroupUsers.setUserData(info);

    // Select the SilverGroups node.
    tcObjects.setSelectedNode(nodeGroupUsers);
}

return true;
}
catch (Exception e)
{
    myDialog.displayError(e);
    return false;
}
}
```

Notes about the code

- Using a hashtable, the code checks the group type passed to the parent method, checks for users belonging to each type, and sets appropriate values. For example, for LDAP groups the user is defined in LDAPUSER and the server is in PROP_LDAP_SERVER.
- The code uses getElement() to get the group object passed to the parent method. Then it uses getChildren() to enumerate and sort (in ascending order) the child objects of the group.
- The code uses the Java method nextElement() to loop through the enumeration of getChildren(). The result is cast to an object of type AgiAdmElement, then checked for elements of type AgiAdmUserReference, where group users are defined in the API. The result is loaded into the tree control.

Identifying users

This sample identifies a user logged on to a SilverStream server. You'll learn about:

- Getting the name of the current user
- Determining if the current user belongs to a specified group

Getting the name of the current user

The following code uses the `getUser()` method to determine the login identity of the current user. This method attempts to match the user login name to a user defined in SilverStream users and groups. If it does not find a match, it returns **Anonymous**. The code uses `setText()` to set the results of `getUser()` to the text box `lblUser`.

```
lblUser.setText(getUser());
```

Determining if the current user belongs to a specified group

This line of code specifies the group to which you want to compare the user to determine membership. It uses `setText()` to set the field `fldUserInGroup` to the group name **Administrators**. The group specified must already be defined as a SilverStream group.

```
fldUserInGroup.setText("Administrators");
```

This code uses the `userInGroup()` method in a conditional statement to test for the group membership of the current SilverStream user. It then sets the test result to the text box `lblUserInGroup`.

```
String userInGroup;

if (userInGroup(fldUserInGroup.getText()))
    userInGroup = "true";
else
    userInGroup = "false";

lblUserInGroup.setText(userInGroup);
```

Customizing the logging class


This sample defines a custom class to log server activity. You'll learn about:

- Creating a custom logging class
- The sample class
- Using the sample class

Creating a custom logging class

You can have the server log HTTP requests, errors, and trace information (see “Using server logging” on page 112). By default, the SilverStream server uses its own internal class to do the logging. If you want to customize the log output—for example, to specify an extended log file format—you can write your own class, then specify it to the SilverStream server.

To enable customized logging, the SilverStream server provides the `com.sssw.srv.api.AgiLogger` interface. The server calls methods in `AgiLogger` to initialize the logger class; log HTTP requests, errors, and trace events; and shut the logger down. Your customized logging class must implement the `AgiLogger` interface.

 For more information about the interface and its methods, see `AgiLogger` in the SilverStream API documentation.

➤ To customize logging:

1. Write your custom logging class that implements `AgiLogger`. The class must:
 - Define all the methods in `AgiLogger`
 - Provide an empty constructor

The class must live outside the server, since it needs to be available when the server starts (see “Writing the custom class” below for more information).

2. Make sure the `.class` file is on the `AGCLASSPATH`.
3. Tell the server to use the custom logging class by doing one of the following:
 - On the SMC General panel, select User Defined logging and specify the name of the class.

OR

- Specify the name of the class in the `httpd.props` file for the **`http-server.com.sssw.srv.logger`** property.
4. Set the types of logging you want in the SMC (HTTP, Error, and/or Trace).
 5. Specify the names of the files to log the output to.

6. Restart the server.

The SilverStream server uses your custom class to log the information.

Writing the custom class The logging class you write is a class that lives outside the SilverStream server but uses SilverStream classes. When you compile the class, you must include SilverStream\lib\SilverServerAll.jar and SilverStream\lib\servlet.jar explicitly on your CLASSPATH when you compile in order to access the SilverStream and servlet classes. For example, your command line might look like this:

```
javac -classpath d:\silverstream\lib\SilverServerAll.jar;  
d:\silverstream\lib\servlet.jar SimpleLogger.java
```

The sample class

Here is a custom logging class that logs information in the W3C compound log file format (like the W3C common log file format, except that it also logs the Referer and User-Agent headers from each HTTP request):

```
//  
// Copyright (c) 2002, SilverStream Software, Inc., All Rights Reserved  
//  
  
// This is a simple example extended logging class that implements  
// the com.sssw.srv.api.AgiLogger interface. It logs to files in  
// the file system in the W3C "Compound Log File Format", creating  
// a new log file each time.  
  
package myLogger;  
  
import java.util.*;  
import java.io.*;  
import java.text.*;  
  
import com.sssw.shr.http.*;  
import com.sssw.rt.util.AgoApplicationException;  
import com.sssw.srv.api.*;  
  
public class SimpleLogger implements AgiLogger  
{  
    private AgiServer m_server = null;  
    private boolean m_logging = false;  
    private String m_logName = null;  
    private PrintWriter m_logWriter = null;  
    private boolean m_errlogging = false;  
    private String m_errlogName = null;  
    private PrintWriter m_errlogWriter = null;
```

```
private boolean m_tracelogging = false;
private String m_tracelogName = null;
private PrintWriter m_tracelogWriter = null;

/* Empty constructor */
public SimpleLogger()
{
}

/**
 * Log normally a full handled request. The request and reply
 * are in the standard HttpServletRequest and HttpServletResponse
 * format. The logger must not modify the request or reply.
 * @param request The HTTP request
 * @param reply The HTTP response
 * @param nbytes The number of bytes sent back to this client
 * @param duration The time to process this request, in milliseconds
 */
public void log(AgiHttpServletRequest request,
               AgiHttpServletResponse reply, int nbytes, long duration)
{
    if (!m_logging)
        return;

    StringBuffer entry = new StringBuffer(120);
    Date now = null;

    // get date from request header, otherwise use current date
    try {
        now = new Date(reply.getDateHeader("Date"));
    } catch (Exception ex) {
        now = new Date();
    }

    // construct date as string
    SimpleDateFormat myFormat =
        new SimpleDateFormat("yyyy.MM.dd 'at' hh:mm:ss z");
    String dateString = myFormat.format(now);

    String user = request.getRemoteUser();

    entry.append(request.getRemoteAddr());
    entry.append(" - ");
    entry.append((user==null) ? "-" : user);
    entry.append(" [");
    entry.append(dateString);
    entry.append("] ");
    entry.append(request.getMethod());
    entry.append(" ");
    entry.append(request.getRequestURI());
```

```
entry.append(" ");
entry.append(request.getProtocol());
entry.append("\n ");
entry.append(reply.getStatus()); // reply status
entry.append(" ");
entry.append(nbytes); // # of emitted bytes
entry.append(" \n");
entry.append(request.getHeader("Referer")); // Referer
entry.append("\n \n");
entry.append(request.getHeader("User-Agent")); // User Agent
entry.append("\n");

log(entry.toString());
}

/**
 * Log a message not associated with any particular session or
 * request to the error log.
 * @param msg The message to be logged
 */
public void log(String msg)
{
    if (m_logging)
        m_logWriter.println(msg);
}

/**
 * Log an error message on behalf of the specified session to an
 * error log.
 * @param session The session logging the error
 * @param msg The message to be logged
 */
public void errlog(AgiSession session, String msg)
{
    errlog("Error on session " + session.getId());
}

/**
 * Log an error message not associated with any particular session
 * to the error log.
 * @param msg The message to be logged
 */
public void errlog(String msg)
{
    if (m_errlogging)
        m_errlogWriter.println(msg);
}

/**
```

```
    * Log a tracing message to the trace log.
    * @param session The session logging the message
    * @param msg The message to be logged
    */
public void trace(AgiSession session, String msg)
{
    trace("Trace from session " + session.getId());
}

/**
 * Log a tracing message not associated with a particular
 * session to the trace log
 * @param msg The message to be logged
 */
public void trace(String msg)
{
    if (m_tracelogging)
        m_tracelogWriter.println(msg);
}

/**
 * Initialize this logger for the specified server.
 */
public void initialize(AgiServer server)
{
    m_server = server;
}

/**
 * Shut down the logger, closing any resources it may hold.
 */
public void shutdown()
{
    if (m_logWriter != null) {
        m_logWriter.close();
        m_logWriter = null;
        m_logging = false;
    }
    if (m_errlogWriter != null) {
        m_errlogWriter.close();
        m_errlogWriter = null;
        m_errlogging = false;
    }
    if (m_tracelogWriter != null) {
        m_tracelogWriter.close();
        m_tracelogWriter = null;
        m_tracelogging = false;
    }
}
```

```
/**
 * Enable or disable standard HTTP-level logging.  If disabled,
 * the log() methods should not write anything to the log.
 * The logFileName is a parameter passed to the logger (null
 * if disabled).
 */
public void enableLogging(boolean doEnable, String logFileName)
{
    try {
        if (m_logWriter != null) {
            m_logWriter.close();
            m_logWriter = null;
        }
        m_logging = doEnable;
        m_logName = logFileName;
        if (m_logging) {
            m_logWriter = new PrintWriter(new FileWriter(m_logName), true);
        }
    } catch (IOException ex) {
        m_logging = false;
        throw new AgoApplicationException(ex, "Error opening " +
            logFileName);
    }
}

/**
 * Enable or disable error logging.  If disabled,
 * the errlog() methods should not write anything to the log.
 * The logFileName is a parameter passed to the logger (null
 * if disabled).
 */
public void enableErrorLogging(boolean doEnable, String errorLogFileName)
{
    try {
        if (m_errlogWriter != null) {
            m_errlogWriter.close();
            m_errlogWriter = null;
        }
        m_errlogging = doEnable;
        m_errlogName = errorLogFileName;
        if (m_errlogging) {
            m_errlogWriter = new PrintWriter(new FileWriter(m_errlogName),
true);
        }
    } catch (IOException ex) {
        m_errlogging = false;
        throw new AgoApplicationException(ex, "Error opening " +
            errorLogFileName);
    }
}
```

```
    }

    /**
     * Enable or disable trace logging. If disabled,
     * the trace() methods should not write anything to the log.
     * The logFileName is a parameter passed to the logger (null
     * if disabled).
     */
    public void enableTraceLogging(boolean doEnable, String traceLogFileName)
    {
        try {
            if (m_tracelogWriter != null) {
                m_tracelogWriter.close();
                m_tracelogWriter = null;
            }
            m_tracelogging = doEnable;
            m_tracelogName = traceLogFileName;
            if (m_tracelogging) {
                m_tracelogWriter = new PrintWriter(new FileWriter(m_tracelogName),
                    true);
            }
        } catch (IOException ex) {
            m_tracelogging = false;
            throw new AgoApplicationException(ex, "Error opening " +
                traceLogFileName);
        }
    }
}
```

Using the sample class

To use the sample, it was necessary to:

1. Compile it and place it in c:\myClasses\myLogger\SimpleLogger.class.
2. Define AGCLASSPATH as follows:
set AGCLASSPATH=c:\myClasses
3. In the SMC General panel, select **User Defined** under **Server logging** and specify the following class:
myLogger.SimpleLogger
4. Specify the kinds of logging wanted and what files to log the output to.
5. Restart the server.

14 Troubleshooting

This chapter describes some techniques and procedures that you can use for troubleshooting the SilverStream server.

This chapter contains the following sections:

- Using error logging
- Low-level debugging
- Setting JDBC/ODBC tracing
- Using the server's command shell
- Using the Watcher
- Common problems starting the SilverStream server
- Using SilverMonitor
- Using the SilverMasterInit program
- Handling a stack overflow
- Miscellaneous issues

Using error logging

SilverStream recommends that you turn on error logging at all times when running the server. Error logging is a lightweight process that prints detailed information about error messages either to the AgErrorLog table in the SilverMaster or to a file you designate. You can activate logging using the SilverStream Management Console (SMC). For more information, see “Using server logging” on page 112.

When logging to the SilverMaster, you can view the log in the SMC by selecting the **Monitor** icon from the toolbar and then selecting **Logs**.



For more information, see “Displaying logs” on page 151.

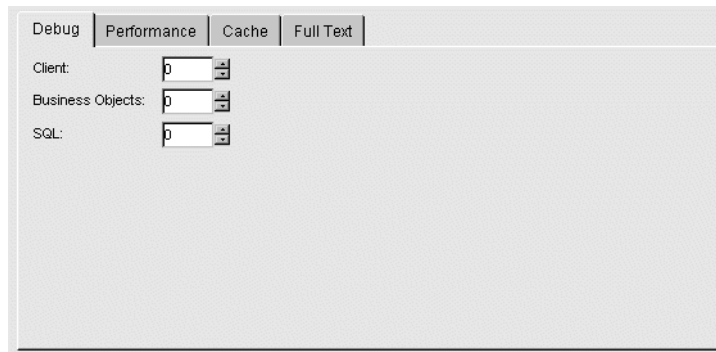
Low-level debugging

The Debug options in the SMC enable the printing of server debug messages to the server console. Options include debugging client requests and database SQL statements. You should activate debugging options only for application debugging purposes, as this activity can significantly inhibit server performance.

NOTE If you are running the server as a service in Windows NT, the output is printed to the error log instead of the console window.

➤ To print debugging messages:

1. Start the SMC.
2. Select the **Configuration** icon from the toolbar.
3. Select **Advanced**.
4. Select the **Debug** tab.



5. Change the value for the type of activity you are debugging.
The number you enter indicates the level of detail you want displayed. The value 0 means that messages are not printed. You have the following debugging options:

Field	Description
Client	<p>If this parameter is set to 1, the server logs information for each client (such as http GETs, PUTs, and POSTs).</p> <p>If this parameter is set to 2 or greater, the server logs the complete request and reply message to the console window.</p> <p>TIP This option is useful for problems with HTTP, servlets, and other client-related issues.</p>

Field	Description
Business objects	<p>If this parameter is set to 1, the server sets the jBroker ORBDebug property to true. For more information about ORBDebug, see the ORB Initialization section of the jBroker tutorial.</p> <p>If this parameter is set to 1 or higher, the server logs information about the execution of each running business object as follows:</p> <ul style="list-style-type: none"> • If set to 1 or 2, provides minimal or maximal information about methods called in the public SilverStream API. • If set to 3 or 4, provides minimal or maximal information about all methods called, even methods not in the public SilverStream API. Set this parameter to 3 or 4 if requested by SilverStream Technical Support. • If set to 5, echoes all output to ServletOutputStreams, so you can see what is being output from the business object. • If set to 6, includes stack traces of various calls, so you can see where calls are being made.
SQL	<p>If this parameter is set to 1, the server logs each SQL statement executed against the database for client data.</p> <p>If this parameter is set to 2 or greater, the server logs additional information that the SilverStream Technical Support group can use to track down server problems.</p> <p>TIP This option is useful for debugging database-related problems.</p>

Setting JDBC/ODBC tracing

If you experience persistent problems with a database connection, you can turn on JDBC or ODBC tracing. As a rule, you should use JDBC tracing. Use ODBC tracing only for databases accessed through ODBC.

➤ To set JDBC tracing:

1. If necessary, create a log file for storing trace data.
2. Shut down the server (see “Shutting down the SilverStream server” on page 99).

3. Open the **httpd.props** file located in **SilverStream\resources**.
4. Add an **http-server.Jdbc.DriverManager.LogFile** entry to the props file and point it to the log file. For example, if the log file is `d:\test\jdbc.log`, create this line in your `httpd.props` file:

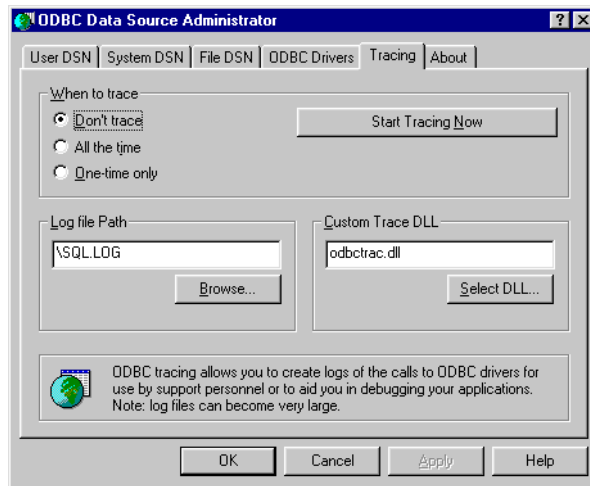
```
http-server.Jdbc.DriverManager.LogFile=d:\\test\\jdbc.log
```

5. Restart the server.

NOTE Use JDBC tracing for troubleshooting only, as it will slow down the server and use considerable disk space.

➤ **To set ODBC tracing:**

1. From the Start menu, choose **Settings**.
2. Open the ODBC panel to display the ODBC Data Source Administrator.
3. Select the **Tracing** tab.



4. Set **When to Trace** to either **One-time only** or **All the time**.

NOTE If you choose **All the time**, your system will be slower—and the trace output will take up disk space on your system.

Using the server's command shell

You can type commands into the server's console window (that is, the window in which the server was started) to obtain diagnostic information about the state of the server. For example, you can get information on memory usage, threads, sessions, and server system properties, as well as enable tracing for different subsystems.

To learn more once the server has started, go to the console window and enter:

```
help
```

All available commands are listed and the command line is indicated by a **!** character (you can change the character using the **prompt** command).

For help on a particular command, enter:

```
help command
```

You can disable the command shell by entering this line in the `httpd.props` file and restarting the server:

```
http-server.com.ssw.srv.commandshell=false
```

Using the Watcher

The Watcher tool helps you understand the state of the server in cases when the server becomes unresponsive. Once activated, the Watcher logs the state of the server once a minute.

You might find the Watcher valuable when faced with problems that are hard to debug.

➤ To use the Watcher:

1. Add the following property to the **httpd.props** file:

```
http-server.com.ssw.srv.httpdwatcher
```
2. Set the value of the property to the pathname of a watcher configuration file. For example:

```
http-server.com.ssw.srv.httpdwatcher=c:\\temp\\watchconfig.txt
```

(Remember to escape backslashes in the **httpd.props** file.)

What happens

If this property is set when the server is started, the server will create a **watcher thread**. The watcher thread sleeps, waking up once a minute to check for the existence of the watcher configuration file supplied as the value of the **httpdwatcher** property.

If the watcher configuration file does not exist As long as the file doesn't exist, the Watcher does nothing and just goes back to sleep. Under these circumstances, the Watcher has minimal impact on server performance. And even if the server hangs, in most cases the Watcher will not hang.

If the watcher configuration file exists When the Watcher discovers that the watcher configuration file exists, it reads the configuration file and uses it to control its further actions.

About the watcher configuration file The watcher configuration file is an ASCII text file that must include:

- A **flags value** that tells the Watcher what information to print
The flags value is a bit-coded integer, in which the bits are defined as follows:
 - Bit 0 (== 0x1) dumps threads info
 - Bit 1 (== 0x2) dumps session info
 - Bit 2 (== 0x4) dumps database connection info
 - Bit 3 (==0x8) dumps fetch-ahead buffer
 - Bit 4 (==0x16) dumps thread event log
- An (optional) **output file name** for the Watcher output (if not supplied, the output goes to the server console)

A typical watcher configuration file looks like this:

```
7
c:\temp\watchout.txt
```

This tells the Watcher to dump information once every minute about threads, sessions, and database connections to the specified output file.

When the watcher configuration file is removed, the server stops logging the output. That means you can turn the logging on and off by creating and removing the watcher configuration file.

Common problems starting the SilverStream server

This section describes some reasons the SilverStream server might fail to start, and how you can address the problem. For more information on troubleshooting server problems, see “Using SilverMasterInit to recreate or refresh SilverMaster” on page 426.

NOTE Server failure is often related to the specific database you are using. For database-specific information, see the *Installation Guide*.

System resource problems

There are two causes of server failure that are due to inadequate system resources:


Cause	Description	What to do
Insufficient disk space	The operating system needs more space to write files to disk	Create more disk space by eliminating or moving files, then try starting the server again
Insufficient memory	This might be a temporary problem or it might indicate insufficient resources	Shut down other programs, expand your swap file, or add more memory to the server machine

Business object generating errors

If an error message indicates that a server listener type of SilverStream business object is preventing the server from starting, use the **-noserverlisteners** server startup option. From a DOS prompt, enter the following:

```
SilverStream\bin\SilverServer.exe -noserverlisteners
```

This command starts the server and allows a SilverStream developer to access the object in the SilverStream Designer.

 For more information about startup options and about stopping and starting the server, see Chapter 5, “Running the Server”.

Database not synchronized

If an error message indicates that a database is not synchronized properly, the server might fail to start. This error might occur if you use a tool outside SilverStream to modify database schema. There are two server startup options you can use to address this problem. Before using these options, be sure that no other users are accessing the database at the same time.

- From a DOS prompt, force-start the server using the **-nodbcheck** option:

```
SilverStream\bin\SilverServer.exe -nodbcheck
```

This prevents the server from checking database integrity.

- From a DOS prompt, start the server using the **-noexitondbcheck** option:

```
SilverStream\bin\SilverServer.exe -noexitondbcheck
```

This allows the server to start and prints errors to the server console even if the database integrity check fails.

If you see errors related to database consistency, go to the SMC and execute the Synchronize database schema option, then restart the server.



For more information, see “Synchronizing the database schema” on page 68.

Using SilverMonitor

SilverMonitor is a background process running on the server that monitors the server status and attempts to restart the server if it terminates abnormally. By default, this process is activated when the server is started. SilverMonitor starts with default parameters, which you can modify. You can also run the server without SilverMonitor.

There are two ways you can modify the parameters:

- Specify parameter(s) as server startup options on the DOS command line. For information about server startup options, see “Using startup options” on page 92.
- Specify machine default values by editing the registry (Windows NT only).

Order of precedence

The order of precedence is as follows:

- Any values in the registry override the default values.
- Any values manually entered as startup options override registry values.

Summary of parameters

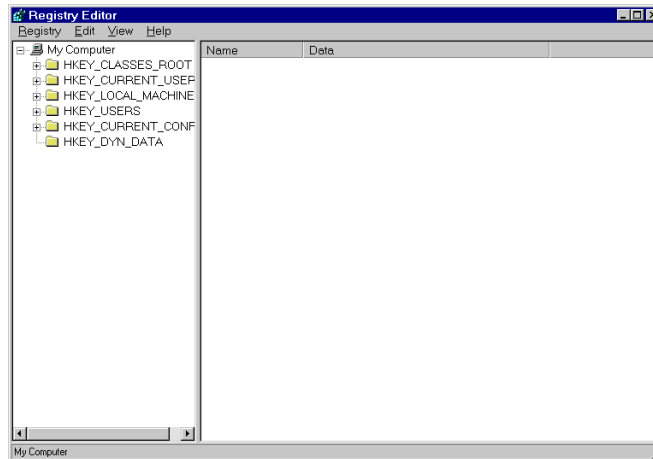
The following is a summary of SilverMonitor parameters.

Startup option	Registry option (NT)	Description
Default	/X	Command in the registry to start SilverMonitor.
-retry <i>number</i>	/C_ <i>number</i>	Number of restart tries. The default is 3.
-minspan <i>number</i>	/M_ <i>minutes</i>	Minute span for restart tries. The default is 10.
—	/D	Debug information about the SilverMonitor process.
-nomonitor	—	Run the server without SilverMonitor.

➤ To modify SilverMonitor parameters in the NT Registry:

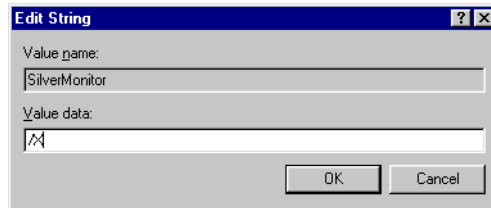
1. From the Start menu, choose **Run**.
2. Type **regedit**.

The Registry Editor displays.



3. Navigate the tree as follows:
HKEY_LOCAL_MACHINE>SOFTWARE>SilverStream Software Inc.>eXtend>AppServer>version number.
4. Double-click the **SilverMonitor** process.
5. When the following dialog displays, type one or more of the options described above. Separate each option with a space.

The following shows the option for starting SilverMonitor.



6. Click **OK**.

What happens

SilverMonitor writes to the NT EventLog when it restarts the server. It also writes to a SilverMonitor.log file in the directory where it is run (usually the SilverStream bin directory). This log file gets an entry every time the monitor starts.

When you restart the SilverMonitor, the log file is emptied and restarted.

Using the SilverMasterInit program

The SilverStream server relies on the SilverMaster database for overall system management. **SilverMasterInit** is a command-line program that performs several types of processes on the SilverMaster database. SilverMasterInit can:

- Recreate or refresh tables and properties used by the SilverMaster database
- Generate logs
- Display debug information
- Regain access to locked resources

This section contains the following topics:

- Command-line options
- Using SilverMasterInit to recreate or refresh SilverMaster
- Regaining access to SilverMaster

Command-line options

The table below describes how and when to run each of the SilverMasterInit command-line options. To see a list of options, type the following at the command prompt:

```
SilverStreamInstallDir\bin\SilverMasterInit -?
```

Administration accounts

There are two administration accounts (**database** and **server**). Both accounts are defined during installation. The server administration account restricts who can log in and administer the SilverStream server. You define the server administration account using SilverMasterInit. After a default installation, the server administrator user account is part of the predefined Administrators group and has the Locksmith privilege.

The SilverStream server uses the database administration account when connecting to the SilverMaster database. The only time you need to specify the SilverMaster database account is when you are running SilverMasterInit.

Entering options

You must enter the database user account name and password for all command-line options. In addition, you need to specify Full or Refresh mode for all SilverMasterInit options **except** those noted in the following table. When you specify a Full mode database initialization, three options (-A, -n, and -W) require that you also define the server administration account name and password on the command line.

You specify parameter(s) as SilverMasterInit startup options on the command line.

SilverMasterInit startup option	Description	Use
-?	Displays usage for SilverMasterInit	Use to check option usage.

SilverMasterInit startup option	Description	Use
<code>+cp:a <i>path</i></code>	Appends specified path to the class path.	<p>This option makes additional Java classes available to SilverStream server applications by appending the specified path to the class path.</p> <p>NOTE You should use the AGCLASSPATH environment variable to extend Java classes.</p> <p>Example:</p> <pre>SilverMasterInit [-f or -r] +cp:a <i>path</i> -U <i>dbusername</i> -P <i>dbpassword</i></pre>
<code>+cp:p <i>path</i></code>	Prepends specified path to the class path.	<p>Don't use this debugging option without first contacting SilverStream Technical Support. Instead, use AGCLASSPATH to make additional Java classes available to SilverStream server applications. See "Setting the AGCLASSPATH variable" on page 138.</p>

SilverMasterInit startup option	Description	Use
-A <i>adminname</i>	Specifies the server administrator user name used to log in to and administer the SilverStream server.	<p>This option lets you define a server administrative account name (and password) when you are creating a new SilverMaster database catalog.</p> <p>The server user account you specify will be part of the Administrators group and have full Locksmith privilege. Use this account to administer the server. See “About your administrator account” on page 129.</p> <p>When running a Full mode database initialization you must specify the server administration account name and password.</p> <p>Example:</p> <pre>SilverMasterInit -f -U dbusername -P dbpassword -A adminusername -W adminpassword</pre>
-a	Causes the SilverStream server to require users to authenticate themselves.	<p>Set this parameter if you accidentally restrict Read access to your SilverMaster database, which includes the login resource. This option also provides a quick way to set authentication without running the SMC. See “Using server authentication to access the login resource” on page 429.</p> <p>You don't need to specify Refresh or Full mode when running the -a option.</p> <p>Example:</p> <pre>SilverMasterInit -a -U dbusername -P dbpassword</pre>

SilverMasterInit startup option	Description	Use
-b	Displays boot environment settings.	<p>Run to see the initial SilverMaster environment properties used by Full mode or Refresh mode.</p> <p>Example:</p> <pre>SilverMasterInit [-f or -r] -b -U dbusername -P dbpassword</pre>
-c	Checks that BLOBs are inserted into the database correctly.	<p>Run to verify that these objects are properly stored.</p> <p>Example:</p> <pre>SilverMasterInit [-f or -r] -c -U dbusername -P dbpassword</pre>
-D <i>database</i>	Deletes all Ag tables from the specified SilverMaster database.	<p>Deletes all existing SilverStream tables (including users, groups, and licensing data) from the specified SilverMaster database. Use to remove a SilverStream database from the server.</p> <p>Unlike Full mode, this option deletes SilverStream data but does not replace it with initial properties.</p> <p>Example:</p> <pre>SilverMasterInit -U dbusername -P dbpassword -D Agdb</pre>

SilverMasterInit startup option	Description	Use
<i>-e error log file</i>	Writes errors to the specified file, which is created as necessary.	<p>If no errors are found, a log file is not created.</p> <p>If you don't specify a path, the error log file is stored in the directory from which you ran SilverMasterInit. The default file name is sminit.log.</p> <p>Example:</p> <pre>SilverMasterInit [-f or -r] -U dbusername -P dbpassword -e c:\silverstream\bin\logs\sminit .log</pre>
-f	Runs in Full mode to create a new SilverMaster database.	<p>Creates new SilverMaster system data and resources. This option deletes existing users, groups, and licensing data.</p> <p>NOTE By default, the server will be restricted when you run Full mode. To install the server unrestricted (for a development environment), run SilverMasterInit in Full mode with -n.</p> <p>When running a Full mode database initialization you must specify the server administration account name and password.</p> <p>Example:</p> <pre>SilverMasterInit -f -U dbusername -P dbpassword -A adminusername -W adminpassword</pre>

SilverMasterInit startup option	Description	Use
<i>-L jdbc log file</i>	Writes JDBC debugging information to the specified log file.	<p>If you don't specify a log file name, this option is ignored.</p> <p>If you don't specify a path, the JDBC log file is stored in the SilverStream\bin directory.</p> <p>Example:</p> <pre>SilverMasterInit [-f or -r] -L c:\SilverStream\logs\jdbclogfile.log</pre>
<i>-l locksmith account</i>	Specifies a user or group account to grant Locksmith privilege to.	<p>Use this option if you accidentally delete all accounts that have Locksmith privilege. See "Regaining access to SilverMaster" on page 428.</p> <p>You don't need to specify Refresh or Full mode when running the -l option.</p> <p>Example:</p> <pre>SilverMasterInit -l -U dbusername -P dbpassword</pre>


SilverMasterInit startup option	Description	Use
-n	Unrestricts access to the SilverStream server.	<p>Use when you do not want to lock down access to SilverStream data nor require user authentication.</p> <p>This option means any user can perform administrative operations and browse directory listings until you lock down access by setting permissions. See “Default server and object security” on page 288.</p> <p>You need to specify Full mode and the server administration user name and password when running the -n option.</p> <p>Example:</p> <pre>SilverMasterInit -n -f -U dbusername -P dbpassword -A adminusername -W adminpassword</pre>
-O <i>table space</i>	Creates all Ag tables in the specified Oracle table space for SilverMaster.	<p>Use this option when creating a SilverMaster database to use with Oracle. More space (than the default) must be allocated for SilverMaster table objects because of the way an Oracle database stores data.</p> <p>Example:</p> <pre>SilverMasterInit [-f or -r] -U dbusername -P dbpassword -O tablespacename</pre>

SilverMasterInit startup option	Description	Use
-P <i>dbpassword</i>	Specifies the database password used by the SilverStream server to access SilverMaster.	<p>The database administration password and associated user account are stored encrypted in the SilverStream registry during server installation. The server will use the specified account name and password at startup to access the SilverMaster database.</p> <p>Example:</p> <pre>SilverMasterInit [-f or -r] -U dbusername -P dbpassword</pre>
-p <i>properties file</i>	<p>Reads startup properties from the specified file.</p> <p>Defaults to httpd.props in the SilverStream resources directory.</p>	<p>Use to specify SilverMaster startup property file name and location other than the default.</p> <p>After you set the property file option with SilverMasterInit, you need to start the SilverStream server from the command line with the -p option to use the new property file.</p> <p>Example:</p> <pre>SilverMasterInit [-f or -r] -p c:\SilverStream\ resources\httpd.newprops -U dbusername -P dbpassword</pre>
-r	Runs in Refresh mode to update SilverMaster resources.	<p>This process skips some of the database installation steps used by Full mode. Use this option to refresh SilverMaster system data and resources when you don't want to delete existing users, groups, and licensing data.</p> <p>Example:</p> <pre>SilverMasterInit -r -U dbusername -P dbpassword</pre>

SilverMasterInit startup option	Description	Use
<i>-W adminpassword</i>	Specifies the server administrator account password used to log in to and administer the SilverStream server.	<p>Use the server administrator user and account password to administer the server.</p> <p>When running a Full mode database initialization you must specify the server administration account name and password.</p> <p>Example:</p> <pre>SilverMasterInit -f -U dbusername -P dbpassword -A adminusername -W adminpassword</pre>
<i>-U dbusername</i>	Specifies the database user account used by the SilverStream server to access SilverMaster.	<p>The database administration user account and associated password are stored encrypted in the SilverStream Registry. The server will use the specified account name and password at startup to access the SilverMaster database.</p> <p>Example:</p> <pre>SilverMasterInit [-f or -r] -U dbusername -P dbpassword</pre>
<i>-v</i>	Generates verbose output as SilverMasterInit runs.	<p>If the process fails, run this option to identify where the failure occurred.</p> <p>Example:</p> <pre>SilverMasterInit [-f or -r] -v - U dbusername -P dbpassword</pre>
<i>-x</i>	Displays SilverMaster initialization properties and then exits without starting SilverMasterInit.	<p>Run to view local server startup properties. This option does not change or refresh properties. Use this debugging option to check for misdirected initialization settings.</p> <p>Example:</p> <pre>SilverMasterInit -x</pre>

Using SilverMasterInit to recreate or refresh SilverMaster

The SilverMaster database, which is created during installation, can also be recreated or updated using SilverMasterInit. The SilverMaster database keeps track of all of the application databases used by the SilverStream server and also holds the SilverStream system tables, including those containing group, user, and licensing information. There is one SilverMaster catalog for each SilverStream server or SilverStream cluster.

 For more information about the SilverMaster database, see “Configuring the SilverMaster database” on page 58.

If your SilverMaster database is damaged, you can run SilverMasterInit. If you cannot start the SilverStream server, try one of the following procedures if nothing else has worked:

- Refreshing the SilverMaster database with SilverMasterInit
- Creating a new SilverMaster database with SilverMasterInit

CAUTION *Connection problems may be due to a corrupt driver connection, damaged application database, or network problems. If you have questions about what is causing your server problem, call SilverStream Technical Support before running SilverMasterInit. Running SilverMasterInit in Full mode will delete the contents of all of your existing SilverStream system tables and replace them with initialized data. Do not run in Full mode if you want to preserve existing SilverStream system tables, including those that contain group, user, and license data.*

Refreshing the SilverMaster database with SilverMasterInit

You can run SilverMasterInit in Refresh mode to upgrade or access SilverMaster properties. The refresh process skips some of the database installation steps used by Full mode. Run SilverMasterInit in Refresh mode to refresh SilverMaster system data and resources without deleting existing users, groups, and licensing data.

NOTE As part of the SilverStream server installation process, SilverMasterInit upgrades resources. You typically upgrade the SilverStream server by running the installation program.

➤ **To run SilverMasterInit in Refresh mode:**

1. Shut down the SilverStream server and Designer.
2. From **SilverStream\bin** directory, enter:

```
SilverMasterInit -r options
```

The following message displays: **Creating Resources will take a few minutes; please wait.**
3. Restart the SilverStream server once SilverMasterInit completes without errors.

Creating a new SilverMaster database with SilverMasterInit

SilverMasterInit can often fix problems caused by someone removing or renaming a file or table that SilverMaster relies on. If you cannot start the SilverStream server or connect to the SilverMaster database, you may need to run SilverMasterInit.

While SilverMasterInit can reset corrupted SilverMaster properties, this program **cannot** repair a corrupted Registry key, configuration files, sample databases, or files associated with databases. To address these types of problems, run the installation program.

To avoid deleting all your database tables, try running SilverMasterInit in Refresh mode (before running it in Full mode) to see if that resolves the server problem.

If you run SilverMasterInit in Full mode to regenerate new SilverMaster properties, you will have to rerun the license install, recreate any SilverStream users and groups, and manually add your SilverStream application databases. Running SilverMasterInit will not alter your application databases unless you have stored application objects in SilverMaster (this is not common practice and is not recommended).

➤ **To run SilverMasterInit in Full mode:**

1. Shut down the SilverStream server and Designer.
2. From **SilverStream\bin** directory, enter:

```
SilverMasterInit -f options
```

The following message displays: **Creating Resources will take a few minutes; please wait.**
3. Record any errors from this command.

4. Once SilverMasterInit completes without errors, rerun your license install using the installation program.

NOTE Running SilverMasterInit in Full mode deletes the table data that stores license information.



For information on adding licenses, see “Managing licenses” on page 136.

5. Start the SilverStream server.
6. Start the Designer and add your application databases again.
7. Recreate any users and groups.

NOTE If you do **not** want to lock down access to SilverStream data and require user authentication, you can run the **-n** option in Full mode.

Regaining access to SilverMaster

You can use SilverMasterInit to regain access to locked resources. The SilverMaster database is where SilverStream stores all system resources and links to other databases. By default, any user with the SilverStream Locksmith privilege has Read access permission to the SilverMaster. If all users are accidentally denied access to the SilverMaster database, no one will be able to access the SilverStream server, through either the Designer or the SMC.

See the following sections if you suspect that Read access to SilverMaster has been restricted:

- Using the Locksmith option to access locked resources
- Using server authentication to access the login resource

Using the Locksmith option to access locked resources

By default, the administrator and any other user with Locksmith privilege can get and set data access permissions for any resource in any database, read all SilverMaster resources, and grant Locksmith privilege to users and groups. The only user that can grant the Locksmith privilege is someone who is already a Locksmith. If all accounts with the Locksmith privilege get deleted, use the SilverMasterInit Locksmith option to grant this privilege to a user to regain access to resources.

NOTE By default, after a new installation or after you run SilverMasterInit in Full mode, an administrator account is automatically created that has Locksmith privilege.

Once a user with the Locksmith privilege can access SilverMaster, that user can unlock resources and reset access privileges.



For more information, see “Using the Locksmith privilege” on page 134.

➤ **To reset Locksmith privilege:**

1. Shut down the SilverStream server and Designer.
2. From **SilverStream\bin** directory, enter:

```
SilverMasterInit -l -U dbusername -P dbpassword
```

The Locksmith can now use the SMC to unlock resources.

Using server authentication to access the login resource

You can set server authentication from either the SMC or the SilverMasterInit command line. You should set server authentication if you accidentally restrict Read access to your SilverMaster database. If users cannot access SilverMaster, run the SilverMasterInit server authentication option to allow users to authenticate themselves when they initially connect to the server. When a user logs in to the SilverStream server from the Designer or the SMC, a request for the login resource is issued. Users cannot access the login dialog if their access to SilverMaster is restricted, because they have no Read access to the database. This is because the SilverMaster has the **login** resource.

You do not need to specify Full or Refresh mode when you run the server authentication option. When you restart the server after setting server authentication, your first attempt to access the server will bring up the credentials dialog and you can log in.

➤ **To set server authentication:**

1. Shut down the SilverStream server and Designer.
2. From **SilverStream\bin** directory, enter:

```
SilverMasterInit -a -U dbusername -P dbpassword
```
3. Restart the SilverStream server.
Users will now be prompted to log in.

Handling a stack overflow

In some obscure situations it is possible to exceed the limits of the stack, in which case the Java Virtual Machine (JVM) throws a `java.lang.StackOverflowError`.

About stacks

On a Windows system in the Java environment, there are at least two program stacks (and possibly more, depending on the JVM implementation), any one of which can overflow and cause a `StackOverflowError` to be thrown:

- There is always a hardware stack, which is used by native code in the JVM itself and by native code that is compiled from Java bytecodes by the Just In Time compiler (JIT).
- There is always a Java bytecode stack, which is used for temporary storage of method call arguments and local variables for Java methods. This is a **soft** stack that is created and managed by the JVM.

Each thread created in the JVM has its own hardware and Java stacks.

What to do if you get a stack overflow

It is possible to alter the size of each of the stacks if it is determined that the default stack size is too small. However, the most common cause of stack overflow errors is a programming error where a method is called recursively a number of times. If this is the case, increasing the size of the stack will not fix the stack overflow problem. Before attempting to increase the stack size, verify that the code does not contain any errors of this nature. Assuming the stack overflow is not caused by an infinite recursion error, it should be possible to fix the stack overflow by increasing the stack size. To determine which stack overflowed is largely a matter of trial and error.

The size of the hardware stack is determined by the operating system using a value stored in the header of the executable. The SilverStream executables (`SilverServer.exe`, `SilverDesigner.exe`, and `SilverJRunner.exe`) all specify a default stack size of 256K.

Changing the stack size

In order to change the stack size, you must modify the executable header using Microsoft's EDITBIN utility. For example, to change the default stack size for `SilverServer.exe` to 512K, use the following command line:

```
EDITBIN /STACK:0x80000 SilverServer.exe
```


Of course, you should make a backup before modifying any executable.

Changing the Java stack size

If increasing the size of the hardware stack doesn't work, it is possible that the Java stack is the problem. In the JDK documentation, there are two command-line options affecting the stack size:

- `-ss` sets the maximum native stack size
- `-oss` sets the maximum Java stack size

The defaults were 128K and 400K respectively. Although these options are no longer documented in JDK 1.2 (Java 2), they appear to have been carried forward as nonstandard (`-X`) switches. To set these options for a SilverStream executable, use `+X` instead of `-X` (the SilverStream executables interpret `+` options as options to be passed to the JVM).

 For more information about SilverStream startup options, see “Using startup options” on page 92.

Example

For example, to set both the native and Java stacks for the SilverStream server to a maximum of 512K, use the following command line:

```
SilverServer +Xss512k +Xoss512k
```

NOTE Increasing any of the default stack size values will increase the amount of virtual memory allocated per thread. Virtual memory is a finite resource, albeit a large one (in a 32-bit operating system such as Windows NT, processes can address up to 2G of virtual memory). Increasing the per-thread virtual memory requirement will reduce the number of threads that can be created. It is important to realize that this could reduce the number of simultaneously connected users that the server is able to support (since the server uses one thread per connected client).

Miscellaneous issues

This section describes some issues that you may need to address.

Browser issues

This section contains browser issues that you may encounter.

Internet Explorer 5 error reporting

By default, Internet Explorer 5 returns its own HTML error page for common HTTP error messages. You need to turn off this processing to get the SilverStream error page.

➤ **To turn off Internet Explorer 5's default error reporting:**

1. Launch Internet Explorer 5.
2. Select **Tools>Internet Options**.
3. Select the **Advanced** tab.
4. Deselect **Show friendly http error messages**.
5. Click **OK**.

Server appears to be hung

If the SilverStream server seems to be hung or in a loop, you can generate a listing for each thread with a stack trace. Doing this does not stop the server.

In Windows NT

In the window where you started the server, press **Ctrl+Break**. The SilverStream server lists each thread with a stack trace.

In UNIX

Determine the process that the SilverStream server is running under:

```
ps -all | grep Silver
```

Issue the following command:

```
kill -3 SilverServer_process_ID
```

The SilverStream server lists each thread with a stack trace in the window where the server was started from.

Socket exceptions

You may receive a Socket Exception message in your NT application log. Typically, this is not a problem: it usually indicates that a client has unilaterally closed a socket. Browsers such as Internet Explorer frequently do this when the connection has been idle for a while, and it will show up as a Socket Exception in the server's console when running with debugging.

You can usually ignore such warnings; they simply reflect a normal situation.

Part III Appendixes

This part describes some miscellaneous topics concerning administering the SilverStream eXtend Application Server

- Appendix A, “The httpd.props File”
- Appendix B, “The SilverStream SNMP Agent”
- Appendix C, “SilverStream System Tables and URLs”

A The httpd.props File

This appendix describes the httpd.props file and has these sections:

- About the httpd.props file
- Server properties

About the httpd.props file

The SilverStream server maintains most of its properties in the internal AgProperties system table (for more information, see Appendix C, “SilverStream System Tables and URLs”). You set these properties using the SMC or the Administration API (you don’t access AgProperties directly).

However, some properties are needed at server startup time and are therefore stored externally so they are available when the server starts. SilverStream stores these properties in the **httpd.props** configuration file, which is located in the **silverstream\resources** directory on your system. You can change these settings either by modifying the httpd.props file or by using the SMC, which updates the httpd.props file when the change is saved.

NOTE Whenever possible, it is best to make changes in the SMC rather than in the httpd.props file.


IMPORTANT Always stop the server before editing the file. You can use the **Stop** button in the SMC. For more information, see “Shutting down the SilverStream server” on page 99.





Server properties




The following properties appear in the default httpd.props file and are listed in alphabetical order. (There are additional httpd.props properties that are optional and rarely used. They do not appear in the following table, but are described in the appropriate sections of the documentation.) All properties are case-sensitive.


NOTE All property names begin with **http-server**.


Property/panel in SMC where settable	Description/default
com.sssw.db.dbplatforms NOTE Not settable in SMC or API.	Database platforms configuration file location Default: \\resources\\platforms.dbl in the SilverStream installation directory NOTE Do not change this value
com.sssw.orb.orbkey SMC: Configuration/General	ORB to use Default: ObjectEra_Jbroker
com.sssw.orb.orbplatforms NOTE Not settable in SMC or API	ORB platforms configuration file Default: \\resources\\orbs.dbl in the SilverStream installation directory NOTE Do not change this value.
com.sssw.srv.agent.debug SMC: Configuration/Advanced/Debug	Prints debug messages for business objects to the server console. The number assigned (1-6) represents the level of verbosity for the messages. Enter 0 to disable. Default: 0
com.sssw.srv.autoupgrade.25 NOTE Not settable in SMC or API	Whether to update databases from Version 2.5 of the SilverStream server when starting the server Default: False

Property/panel in SMC where settable	Description/default
com.sssw.srv.client.debug SMC: Configuration/Advanced/Debug	Prints debug messages for client connections to the server console. The number assigned (1-5) represents the level of verbosity for the messages. Enter 0 to disable. Default: 0
com.sssw.srv.ContentCache.Disk.Directory NOTE Not settable in SMC or API	Location of content cache Default: \\temp\\ContentCache in the SilverStream installation directory
com.sssw.srv.enable.fulcrum NOTE Not settable in SMC or API	Whether SilverStream tries to start Fulcrum SearchServer when starting up. If true , SilverStream tries to start Fulcrum; if false , it does not. The SilverStream installation program sets this to true if Fulcrum is present when SilverStream is installed; otherwise, it sets the property to false. You should reset the value accordingly if you install or uninstall Fulcrum after SilverStream has been installed.
<ul style="list-style-type: none"> • com.sssw.srv.http.listen_admin • com.sssw.srv.http.listen_des • com.sssw.srv.http.listen_rt SMC: Configuration/General	Whether the server listens on any or all of the HTTP administration, design, or runtime ports. Default is true, meaning that the server listens on the HTTP port. False means that the server will not listen on the HTTP port.  See “Setting up separate ports” on page 106. Default: true
com.sssw.srv.http.webmaster SMC: Configuration/General	SilverMaster data source name Default: Set at installation

Property/panel in SMC where settable	Description/default
<ul style="list-style-type: none"> • com.sssw.srv.https.listen_dsa_admin • com.sssw.srv.https.listen_dsa_des • com.sssw.srv.https.listen_dsa_rt SMC: Security/Server Security	Whether the server listens on any or all of the HTTPS DSA administration, design, or runtime ports.  See “Setting up separate ports” on page 106. Default: false
<ul style="list-style-type: none"> • com.sssw.srv.https.listen_rsa_admin • com.sssw.srv.https.listen_rsa_des • com.sssw.srv.https.listen_rsa_rt SMC: Security/Server Security	Whether the server listens on any or all of the administration, design, or runtime HTTPS RSA ports.  See “Setting up separate ports” on page 106. Default: false
<ul style="list-style-type: none"> • com.sssw.srv.https.port_dsa_admin • com.sssw.srv.https.port_dsa_des • com.sssw.srv.https.port_dsa_rt SMC: Security/Server Security	The HTTPS DSA administration, design, and/or runtime port. At initialization time, the server will bind its accepting socket to the host it runs on, and to the provided port. The server will use a DSA/Diffie-Hellman certificate and encryption algorithms for SSL on this port.  See “Setting up separate ports” on page 106. Default: 444
<ul style="list-style-type: none"> • com.sssw.srv.https.port_rsa_admin • com.sssw.srv.https.port_rsa_des • com.sssw.srv.https.port_rsa_rt SMC: Security/Server Security	The HTTPS RSA administration, design, and/or runtime port. At initialization time, the server will bind its accepting socket to the host it runs on, and to the provided port. The server will use an RSA certificate and encryption algorithms for SSL on this port.  See “Setting up separate ports” on page 106. Default: 443

Property/panel in SMC where settable	Description/default
<p>com.sssw.srv.international.UrlEncoding</p> <p>NOTE Not settable in SMC</p>	<p>URL-encoding and decoding scheme</p> <p> See “Specifying character set encoding” on page 122.</p> <p>Default: utf-8</p>
<p>com.sssw.srv.jms.debug</p> <p>NOTE Not settable in SMC or API</p>	<p>Prints JMS-related debug messages to the server console. For basic debugging, specify 1. For deeper debugging, specify a number greater than 1. Specify 0 to disable.</p> <p> See “Running the JMS (jBroker MQ) server” on page 123.</p> <p>Default: 0</p>
<p>com.sssw.srv.jmsServerLaunch</p> <p>NOTE Not settable in SMC or API</p>	<p>Whether the SilverStream server tries to start the JMS (jBroker MQ) server when starting up. To automatically start the JMS server, specify true. Otherwise, specify false.</p> <p>When you install the SilverStream server, the installation program asks if you want to configure jBroker MQ, then it sets this property according to your response.</p> <p> See “Running the JMS (jBroker MQ) server” on page 123.</p> <p>Default: false (if jmsServerLaunch property is removed)</p>
<p>com.sssw.srv.loader.debug</p> <p>NOTE Not settable in SMC</p>	<p>Whether to turn on classloader-related debugging messages, which display in the Server console. Values range from 1 to 5; 5 gives the most detail. Enter 0 to disable.</p> <p>Default: 0</p>
<p>com.sssw.srv.logger</p> <p>SMC: Configuration/ General</p>	<p>The logging class</p> <p>Default: com.sssw.srv.http.AgLogger</p>

Property/panel in SMC where settable	Description/default
com.sssw.srv.logger.logging SMC: Configuration/ General	Whether to log every standard http client request to the server Default: false
com.sssw.srv.logger.errlogging SMC: Configuration/General	Whether to turn on error logging Default: true
com.sssw.srv.logger.errorlogname SMC: Configuration/General	The name of the error log file (if you are logging to a file) Default: errlog
com.sssw.srv.logger.logname SMC: Configuration/General	The name of the http log file (if you are logging to a file) Default: log
com.sssw.srv.logger.tracelogging SMC: Configuration/General	Whether to turn on trace logging Default: false
com.sssw.srv.logger.tracelogname SMC: Configuration/General	The name of the trace file (if you are logging to a file) Default: traces
com.sssw.srv.logger SMC: Configuration/General	Java class to do the logging Default: com.sssw.srv.http.AgLogger
com.sssw.srv.nameServicePort SMC: Configuration/General	The port of the name service that the server is using. Default: 54890
<ul style="list-style-type: none"> • com.sssw.srv.port_admin • com.sssw.srv.port_des • com.sssw.srv.port_rt SMC: Configuration/General	Server http administration, design, and/or runtime port.  See “Setting up separate ports” on page 106. Default: 80

Property/panel in SMC where settable	Description/default
com.sssw.srv.server NOTE Not settable in SMC or API.	SilverStream server protocol version Default: SilverStream server/4.0
com.sssw.srv.sminit NOTE Not settable in SMC or API	Location of internal sminit.props file, used by SilverMasterInit (do not edit this file). Default: <i>SilverStreamInstallDir</i> \\resources\\sminit.props
com.sssw.srv.sql.debug SMC: Configuration/Advanced/Debug	Prints debug messages for SQL database calls to the server console. The number assigned (1-5) represents the level of verbosity for the messages. Enter 0 to disable. Default: 0
com.sssw.srv.SupportNTLocalGroups NOTE Not settable in SMC (settable as PROP_SUPPORT_NT_LOCAL_GROUPS in AgiAdmServer and AgiAdmCluster)	Whether the SilverStream server uses NT local groups for authentication. Default: true  See “Speeding NT authentication” on page 212.
com.sssw.srv.system.out.log.allowed	Whether the SilverStream server when run as an NT service will log output to System.out and System.err. Default: False.
com.sssw.srv.system.out.log.file	The file to which the SilverStream server when run as an NT service will log output to System.out and System.err (if the property system.out.log.allowed is True). Default: <i>SilverStreamInstallDir</i> \\temp\\SilverServerSysOutput.txt
Jdbc.LDSKey SMC: Configuration/Databases	JDBC driver for SilverMaster Default: Set at installation.

Property/panel in SMC where settable	Description/default
Jdbc.URL SMC: Configuration/Databases	The database URL. The driver uses it to connect to the SilverMaster database. The URL is driver-specific. See your driver documentation for more information. Default: Set at installation.
Jdbc.URL.Attributes SMC: Configuration/Databases	Any extra attributes to set for the JDBC driver. The syntax is driver-specific. See your driver documentation for more information. Default: Set at installation.

B The SilverStream SNMP Agent

This appendix describes how you can set up and test SNMP to monitor the SilverStream server and has these sections:

- About SNMP
- SNMP implementation overview
- Setting up SNMP for the SilverStream server

About SNMP


Simple Network Management Protocol (SNMP) is a protocol used to remotely manage and control nodes on a TCP/IP network. Using SNMP, one workstation running management software can monitor information being collected by routers, servers, and other workstations on the system. This information is used to determine the performance integrity of the network.

NOTE The SilverStream SNMP implementation currently runs on the Windows NT and Windows 2000 platforms only, and does not allow the SNMP service to control the SilverStream server.

SNMP implementation overview

The SilverStream server implements SNMP using the following components:

Component	Description
snmp_options.props	<p>File that defines the following settings, which are used by the AgSNMPGetStats servlet:</p> <ul style="list-style-type: none"> • StatisticsUpdateInterval—number of seconds to wait before updating the statistics file. Default is 120. • WriteStatisticsEnabled—whether to write server statistics and the update interval to AgSNMP.props; 0 for false, 1 for true. Default is 1. • StatisticsDebug—whether to send debugging messages to the server console; 0 for false, 1 for true. Default is 0. <p>The file is located in Resources in the SilverStream installation directory.</p>
AgSNMPGetStats servlet	<p>A load_on_startup servlet that must be deployed to the SilverMaster database. It is responsible for updating server statistics in the AgSNMP.props file.</p>
AgSNMP.props	<p>File that the AgSNMPGetStats servlet writes the server statistics to at the interval specified in snmp_options.props. (WriteStatisticsEnabled must be set to 1 in snmp_options.props for the statistics to be written.)</p> <p>The file is located in Resources in the SilverStream installation directory.</p>
SNMP extension agent (AgSNMP40.dll)	<p>Implements the Windows NT SNMP Application Program Interface (API). The SNMP extension agent reads the SilverStream server statistical information from AgSNMP.props.</p>

 For a list of statistics and object identifiers, see “Setting up access from your SNMP Management node” on page 453.

How the SilverStream components work

A SilverStream `load_on_startup` servlet (`AgSNMPGetStats`) updates the `AgSNMP.props` file according to a scheduled interval. When the servlet is loaded, the `init()` method is fired. This method:

1. Gets registry information from the server to determine the SilverStream installation path.
2. Reads the `snmp_options.props` file in the SilverStream Resources directory in the SilverStream installation directory. This file contains settings such as whether to print debugging messages to the server console or write out statistics as well as the interval to update statistics to the `AgSNMP.props` file.
3. Starts a timer task that runs every minute to check whether the file update interval in the `snmp_options.props` file has changed.
4. Starts a timer task that runs on the specified file update interval to build the statistics data and write it to the `AgSNMP.props` file in the SilverStream Resources directory.

On an SNMP GET request, if the update interval has elapsed, the SilverStream extension agent updates the MIB data by accessing the registry to get the SilverStream path and reading the `AgSNMP.props` file in the SilverStream Resources directory. Otherwise, it returns the value that was stored the last time the file was read. If the timestamp in the file is not updated within the given interval, the Server Responding status is set to false, indicating a possible problem with the SilverStream server.

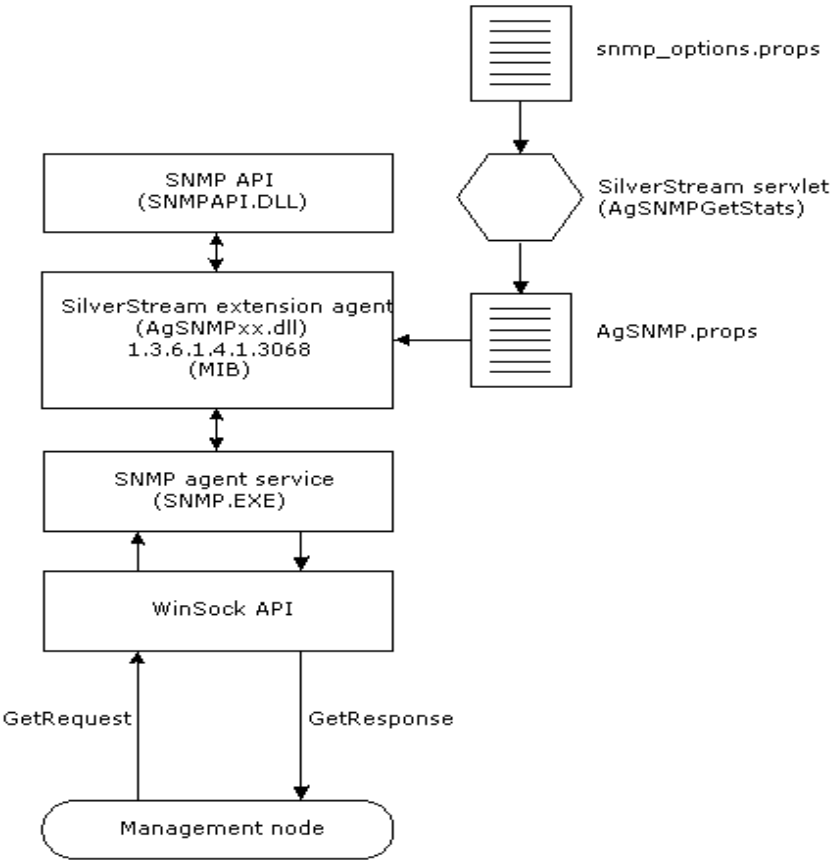
NOTE The SilverStream extension agent does not use a timer to determine whether the update interval has changed. If the interval is decreased by a substantial amount, it can give a false Server Responding status. In the event that the interval value is decreased, the SNMP service should be stopped and restarted.

Process flow and terminology

The following terms (shown in the following figure) are used in an SNMP-enabled architecture:

Term	Description
Management node	The workstation or server running one or more network management processes. These processes are usually software applications that gather information from the managed nodes, or SNMP agents. Examples of management node software include Unicenter TNG from Computer Associates, OpenView from Hewlett-Packard, and Tivoli from IBM.
Managed Information Bases (MIBs)	The hierarchical map of all managed objects and how they are accessed.
Managed objects (MIB objects or variables)	The collection of objects that describe the SNMP managed node to the management node. This data is defined with a specific set of attributes that are manipulated using the standard SNMP operations Get, GetNext, and Set.
Object identifier (OID)	A unique identifier for an MIB variable.
SNMP agent	Software or firmware that runs as one or more processes on a managed server. The SNMP agent provides management services by collecting and returning management information requested by the management node. An SNMP agent can be read-only, or it may allow the management node to control or alter the node it is managing. An SNMP agent may also generate traps, which are unrequested notifications of events.
Extension agent (Subagent)	A DLL that implements a set of registered managed objects defined in an MIB module and communicates with the SNMP service using the SNMP API.

The following figure shows how the SilverStream components work within the SNMP framework.



Setting up SNMP for the SilverStream server

Setting up SNMP as an NT service involves the following basic steps:

1. Installing the SNMP software as a service
2. Installing the SilverStream server
3. Deploying the AgSNMPGetStats servlet
4. Testing the SNMP program

Installing SNMP as a service

If you are installing the SNMP software service on a machine that is currently running the SilverStream server, you need to first stop the SilverStream server.

➤ **To install the SNMP service:**

1. Log on to NT with administrator privileges.
2. Select the **Network** control panel.
3. Select the **Services** tab.
4. Click **Add**.
5. Select **SNMP Service** (you may need the NT install CD).
6. Complete the **Agent** tab:
Contact: **Name of person or name of machine**
Location: **Any descriptive term**
7. Complete the **Traps** tab (optional):
Community name: **public (optional)**
Trap destinations: **IP address of the machine (optional)**
8. After installing the **SNMP Service**, the following items need to be reinstalled:
 - The latest NT Service Pack
 - Internet Explorer
9. If the SilverStream server has already been installed on this machine, make sure the necessary SilverStream registry settings are still defined, as listed in the next section.

Installing the SilverStream server

If you haven't already installed the SilverStream server, use the SilverStream installation program to install SilverStream on the machine to be managed. If you choose to reinstall the SilverStream server, you must first stop the SNMP service. Otherwise, the install program will not be able to write over the AgSNMP40.dll file.

The SilverStream installation program takes care of the required registry key entries and places the agent in the correct location. The required registry entries are as follows:

Key:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SNMP\Parameters\ExtensionAgents
```

Name:

```
AgversionSNMP
```

Value:

```
Software\SilverStream Software  
Inc.\eXtend\AppServer\version\SNMP\ExtensionAgents\AgSNMPAgent\CurrentVersion
```

Key:

```
HKEY_LOCAL_MACHINE\SOFTWARE\SilverStream Software  
Inc.\eXtend\AppServer\version\SNMP\ExtensionAgents\AgSNMPAgent\CurrentVersion
```

Name:

```
Pathname
```

Value:

```
SilverStream Install Directory\bin\AgSNMPversion.dll
```

Deploying the AgSNMPGetStats servlet

A number of files are provided to build and deploy the AgSNMPGetStats servlet. They are located in the SilverStream `servertools\snmp` directory:

File	Filename
A readily deployable EAR file	SilverGetStats.ear
A batch file to deploy the EAR to the SilverMaster	deploySilverGetStats.bat
A batch file to undeploy the EAR from the SilverMaster	deleteSilverGetStats.bat
A deployment plan	SilverGetStats_depl_plan.xml)
eXtend Workbench project files to build the WAR and EAR files	SilverGetStatsWar.spf and SilverGetStatsEar.spf
Source files to build the WAR and EAR files	—

To deploy the EAR, run the `deploySilverGetStats.bat` file, passing the server name and SilverMaster datasource name.

You can also rebuild the EAR and deploy it using the provided Workbench projects.

Testing the program

The SilverStream server provides a tool that allows you to test the SNMP installation from a DOS prompt.

➤ **To test the SNMP extension agent:**

1. Stop and restart the **SNMP Service**.
2. Open a DOS window.
3. Change to the SilverStream bin directory.
4. Run the batch file **SilverSNMPGetStats.bat**, passing the server name and SNMP parameter name. To get a list of available parameters, type:

```
SilverSNMPGetStats -?
```

Setting up access from your SNMP Management node

The SilverStream Object Identifier is composed of an enterprise ID and the OID identifying each Managed Object (MIB). You will need the following information to set up access to the SilverStream server OIDs from your SNMP Management node:

Item	Description
Private Enterprise ID	A unique number assigned to a company. The SilverStream private enterprise ID is 3068.
Object Identifier (OID)	A unique identifier that references a managed object. An OID is the location of a managed object within a MIB namespace. The OID of a MIB object is also referred to as the object's identity or registration. The SilverStream host name OID is 1.3.6.1.4.1.3068.1.7.6.1.0.

The following are the SilverStream OIDs. When accessing MIB data using SNMPTool.exe, you must precede the OID with a period.

Statistic description	OID	Data type
Host Name OID	1.3.6.1.4.1.3068.1.7.6.1.0	OCTET STRING
Server Revision OID	1.3.6.1.4.1.3068.1.7.6.2.0	OCTET STRING
Server Start Time OID	1.3.6.1.4.1.3068.1.7.6.3.0	OCTET STRING
Data Time Snapshot OID	1.3.6.1.4.1.3068.1.7.6.4.0	OCTET STRING
Maximum Requested URL OID	1.3.6.1.4.1.3068.1.7.6.5.0	OCTET STRING
Minimum Requested URL OID	1.3.6.1.4.1.3068.1.7.6.6.0	OCTET STRING
Server Load OID	1.3.6.1.4.1.3068.1.7.6.7.0	INTEGER*
Free Thread Count OID	1.3.6.1.4.1.3068.1.7.6.8.0	INTEGER*
Idle Thread Count OID	1.3.6.1.4.1.3068.1.7.6.9.0	INTEGER*
Total Thread Count OID	1.3.6.1.4.1.3068.1.7.6.10.0	INTEGER*
Hit Count OID	1.3.6.1.4.1.3068.1.7.6.11.0	INTEGER*

Statistic description	OID	Data type
Mean Request Time OID	1.3.6.1.4.1.3068.1.7.6.12.0	INTEGER*
Max Request Time OID	1.3.6.1.4.1.3068.1.7.6.13.0	INTEGER*
Min Request Time OID	1.3.6.1.4.1.3068.1.7.6.14.0	INTEGER*
Emitted Bytes OID	1.3.6.1.4.1.3068.1.7.6.15.0	INTEGER*
Free Memory OID	1.3.6.1.4.1.3068.1.7.6.16.0	Counter*
Total Memory OID	1.3.6.1.4.1.3068.1.7.6.17.0	Counter*
Garbage Collection Count OID	1.3.6.1.4.1.3068.1.7.6.18.0	Counter*
Idle Sessions OID	1.3.6.1.4.1.3068.1.7.6.19.0	INTEGER*
Total Sessions OID	1.3.6.1.4.1.3068.1.7.6.20.0	INTEGER*
Server Responding OID	1.3.6.1.4.1.3068.1.7.6.21.0	OCTET STRING

*Before eXtend Application Server Version 4, these statistics were all strings.

C SilverStream System Tables and URLs

The SilverStream server stores application and system data in a database called the SilverMaster (the SilverMaster is described in Chapter 4, “Data Source Configuration”) and in application databases. This appendix provides a listing of system tables and database URLs:

- SilverStream internal system tables
- SilverStream database URLs

Where application and system data is stored

NOTE The items listed are reserved for SilverStream’s use. This listing is provided for informational purposes only.

SilverStream internal system tables

The following are the SilverStream internal tables. The information in these tables controls major server functions such as application data storage and management, security, and server clustering for load balancing. Some tables are used in each database connected to the server. The remainder of the tables exist in the SilverMaster only.

Tables in all databases The following tables exist in or in conjunction with each database connected to the SilverStream server (including SilverMaster).

Database system table	Purpose
AgAccessRights	Contains security access information on resources.
AgAgents	Contains business object information.
AgContents	Contains the contents of application resources, such as pages and forms.

Database system table	Purpose
AgInfo	Contains a catalog of known databases and their system information.
AgResources	Manages details of all objects stored in SilverStream databases.

Tables only in SilverMaster The following tables exist in the SilverMaster database only.

SilverMaster table	Description
AgCacheMgr	Contains Cache Manager information.
AgCacheMgrGroup	Contains Cache Manager group information.
AgCertificates	Contains certificate information.
AgCluster	Contains SilverStream cluster information. Used with load balancing software.
AgClusterEnv	Contains SilverStream cluster information. Used with load balancing software.
AgDispatcherMgr	Contains Dispatcher information. Used with SilverStream clusters.
AgDispatcherMgrGrp	Contains Dispatcher group information. Used with SilverStream clusters.
AgErrorLog	Contains SilverStream error log information.
AgGroupMembers	Contains a list of members of SilverStream groups in UUID format.
AgLoadMgr	Contains Load Manager information. Used with SilverStream clusters.
AgLoadMgrGroup	Contains Load Manager group information. Used with SilverStream clusters.
AgLog	Contains SilverStream log information.
AgObjects	Contains SilverStream object information.

SilverMaster table	Description
AgPasswords	Contains SilverStream password information.
AgProperties	Contains SilverStream server properties
AgPrincipals	Contains a list of SilverStream users and groups in UUID format.
AgServer	Contains SilverStream server information
AgSessBeans	Used for J2EE failover support and stateful session bean passivation. Contains HTTP session state and stateful session beans (local and remote).
AgSessSubjects	Used for J2EE fail over support. Contains login state (via encrypted Subject info) for HTTP session state if session is logged in.
AgTraceLog	Contains trace log information.
AgUserLicense	Contains SilverStream user license information.

SilverStream database URLs

The following table lists the directory structure for SilverStream applications that contain SilverStream tables and objects. By default, these items are located in **/database/SilverStream/**. This listing is provided for informational purposes only.

Database URL	Description
Administration	Server administration settings.
Classes	Compiled Java class subdirectories and files.
ClusterAdmin	Administrative functions (server clusters only).
Downloads	Temporary communication endpoints.
ErrorLogs	Error messages.
Login	Forces user login access.

Database URL	Description
Logout	Logs user out of session.
Meta/...	Corresponds to... <ul style="list-style-type: none"> • Agents • Certificates • Entities • Forms • Licenses • Reports • ServerCertificate • Tables • UUIDTranslator • Views • Webbases
Objectstore/	Uploaded media files.
Pages	HTML pages (active presentation & static).
Resources	Directory of system files.
Security	Endpoint for security administration.
Sessions	Displays current session information, including session number, user, host, and state.
SilverJRunnerInstall	SilverJRunner installation.
Statistics	Displays server statistics.
Timestamps	Displays time stamped events (internal only).
VersionCheck	Displays SilverStream version number.

Index

A

- A startup option 419
 - a startup option 96
 - a startup option 419
 - access
 - administration operations 288
 - restricted and unrestricted 288, 293
 - to Locksmith privilege 422, 428
 - to login resource 428
 - to SilverMaster 428
 - access control
 - about 271
 - changing defaults 276
 - changing user access 276
 - defined 203
 - INLIST operator 284
 - object security defaults 288
 - permission types 272
 - row-level security 283
 - using advanced expressions 280
 - using simple list 279, 280
 - UUID support 281
 - account
 - administration 129, 417
 - database 129, 417
 - administration
 - account 419, 425
 - permissions 128
 - Administration API
 - about 367
 - client-side capabilities 369
 - container hierarchy 376
 - containers and elements 374
 - creating a custom logging class 400
 - enumerating database elements 381
 - enumerating security providers, users, and groups 387
 - identifying users 399
 - info parameter 382
 - interfaces 371
 - obtaining elements 382
 - packages 370
 - server objects 371
 - uses 368
 - using 376
 - administration port 106, 108
 - administration procedures
 - quick reference 7
 - administration resource 272, 295
 - Administrators group 128, 288
 - advanced expressions
 - INLIST operator 284
 - row-level security 283
 - using for access control 280
 - UUID support 281
 - AGCLASSPATH 94, 138, 340, 418
 - AgDigitalIDStep1 executable 240
 - AgDigitalIDStep2 executable 240
 - AgErrorLog table 407
 - AgiAdmContainer objects 375
 - AgiAdmDatabase objects 371
 - AgiAdmDirectory 387
 - AgiAdmDirectory objects 371
 - AgiAdmElement 387
 - AgiAdmElement objects 374
 - AgiAdmGroup 387
 - AgiAdmServer objects 369, 371
 - AgiLogger 400
 - agrootca.jar 264
 - AgWSIUser utility 191
 - algorithms 202
 - Allow users to modify own account, field in the SMC 267
 - anonymous access 214
 - API Server Administration
 - see Administration API
 - append path startup option 418
 - application database
 - restricting access 298
 - Apply to this directory and all its descendants, button in SMC 278
 - Apply to this directory only, button in SMC 278
-

- authentication
 - about 293
 - Allow users to modify own account, field in the SMC 267
 - certificate 201
 - defined 202
 - DSA encryption 202
 - enabling 266
 - require login for access 278
 - require user authentication 267, 419, 429
 - RSA encryption 202
 - user names and passwords 221
- authorization 271

- B**
- b startup option 420
- BLOBs
 - stored 420
- browsers
 - connecting through SSL to server 204
- buffering reply size
 - setting 303
- Buffering reply size, field in SMC 305
- business objects
 - debugging 409
 - in a server cluster 330
 - troubleshooting 413

- C**
- c startup option 420
- cache
 - setting parameters 312
- Cache Manager
 - adding to a cluster profile 344
 - cluster component defined 329
 - function described 330
 - setting failover parameters 354
 - starting in UNIX 338
 - starting in Windows NT 338
- certificate jar file 264
- certificates
 - about 226
 - CA 228
 - client 228
 - enabling 266
 - installing in a server cluster 358
 - installing on server 248
 - server 201, 228, 250
 - SSL 265
 - trusted 265
- character set
 - international 122
- cipher suites 253
- classic development environment
 - independent 29
 - shared 31
- +classic startup option 93
- client connections
 - load levels 308
 - MAXFREE parameter 309
 - MAXTHREADS parameter 309
 - sessions and threads 306
 - setting parameters 310
 - states 307
- client requests
 - reuse connection 305
 - setting timeout for 304
- clients
 - HTML 17, 40
 - J2EE 17, 40
 - Java 17, 40
- clusters
 - adding servers in UNIX 341
 - adding servers in Windows NT 340
 - adding servers to existing cluster profile 350
 - administering 348
 - Cache Manager 330
 - certificate install 358
 - components overview 328
 - configuration options 330
 - configuration requirements 329
 - creating cluster profile 342
 - defined 327
 - Dispatcher function 333
 - distribution mapping 333

- failover 336
- Load Manager 332
- removing server from existing cluster profile 350
- restarting servers to activate 347
- setup overview 337
- starting the software components 338
- command shell, server 411
- command-line options
 - SilverMasterInit 417
- compiler
 - switches 93
- com.sssw.rts.acl 370
- com.sssw.rts.adminapi 370
- com.sssw.rts.adminclient 370
- com.sssw.srv.api 370
- configurations
 - network 34
 - running more than one server on a host 119
 - server 27
- connection pool connections
 - about 314
 - setting default min/max number 315
 - setting min/max number for a connection pool 315
- connections
 - managing 314, 319
 - reusing client 305
- cookies 48
- +cp startup option
 - a path option 418
 - p path option 418
 - SilverStream server 94
- creating a new SilverMaster 427

D

- data
 - integrity 203
 - privacy 203
- data encryption 202
- database accounts
 - creating 54
- database connections
 - about 314, 319
 - prefetch buffers 325
 - session timeout parameter 304

- setting default min/max number 315, 320
 - setting min/max number for a specific
 - database 315, 320
- database logging 113
- database system tables 455
- database URL, SilverStream server, default 140
- database URLs, SilverStream 457
- databases
 - access to 55
 - configuring 66
 - creating accounts for SilverStream 54
 - JDBC access 55
 - ODBC access 55
 - setting min/max number of connections 315, 320
 - SilverMaster 58
 - using only a subset of tables 62
- +debug startup option 94
- debugging
 - business objects 409
 - client 408
 - command-line option for 96
 - JBroker ORB 409
 - JDBC tracing 409
 - ODBC tracing 409
 - options 408
 - SQL 409
 - using server command shell 411
- Default Execute
 - permission type defined 274
- Delete idle connections, button in SMC 68
- deployed objects
 - about 139
 - disabling 142
 - enabling 142
 - managing 139
 - shutting down 142
 - undeploying 142
- design port 106
- Designer, starting 429
- Developers group 128, 289
- development environment
 - security checklist 299
- development server
 - restricting access 299

- directory level
 - permissions 285, 287
- directory listing, restricting 294
- disable HTML directory listing, field in SMC 267
- Disable indexing before full-text search, property 163
- disk cache 312
- Dispatcher, SilverStream
 - adding to a cluster profile 345
 - cluster component defined 329
 - DSA port 347
 - function described 333
 - RSA port 347
 - starting in UNIX 339
 - starting in Windows NT 339
- distribution map
 - defined 333
 - editing 353
- +Djava.compiler 94
- driver sets 62
- DSA certificates, creating and installing 240
- DSA encryption 202
- DSA port
 - enabling and changing 251
 - in cluster dispatcher 347

E

- e startup option 421
- EARs
 - security roles 299
- EJBs (Enterprise Java Beans)
 - maintaining 140
 - security roles 299
- e-mail
 - assigning the port 161
 - assigning the protocol 161
 - assigning the server 161
 - format 161
 - IMAP 161
 - polling for 159
 - Pop3 161
 - setting up 159
- Enable HTTP port, check box in SMC 252, 346
- Enable HTTP port, field in SMC 111
- Enable partial indexing at startup, property 163

- Enable RMI Server, field in SMC 117
- enCommerce 192
- encryption
 - algorithms 202
 - defined 202
 - specifying using cipher suites 253
- error logging
 - about 407
 - setting 114
 - SilverMasterInit 422
- error page, HTML
 - overriding Internet Explorer 5 default page 432
- executable objects 285
- Execute
 - permission type defined 274
- Execute permission 285, 287
- expressions, security 271
- Extension Agent
 - SNMP term defined 448

F

- f startup option 421
- failover
 - component recovery 336
 - defined 328
 - function described 336
 - persistent failure 337
 - setting Cache Manager parameters 354
 - setting Load Manager parameters 355
 - SilverMonitor in clustering 336
- file logging 113
- firewalls
 - defined 31
 - separate ports 107
 - server configuration 33
- forms content
 - encoding 122
- free memory 158
- Fulcrum full-text search
 - disabling indexing at startup 163
 - installing in a server cluster in NT 360
 - installing in a server cluster in UNIX 365
 - setting properties in SMC 161

Full mode

SilverMasterInit 427

full-text search

see Fulcrum full-text search 161

G

garbage collection count 158

getAccess

configuration 195

integration 192

security 196

global certificates 229

global group 279

Global Secure Site IDs 229

graphing server statistics 146

groups

accessing 220

adding users 132

creating 132

default permissions for 288

defined 132

determining whether current user in group 399

enumerating 387

predefined 128

H

host name

specifying for server startup 99

-host startup option 96

hosts

running more than one server on 119

HTML

disabling directory listing 267

HTML directory 294

HTTP

about 44

communications 46

logging 114

request time statistics 157

resources 45

response chains 47

security validation 202

URLs 44

HTTP port

disabling 252, 346

in server cluster 346

setting 111

httpd.props file 107, 437

HTTPS

about 200

creating connection to server 204

I

IIOP SSL port 118

IIS server integration 165

IMAP server 159

Indexer interval property in SMC 162

info administration parameter 382

InitialContext

viewing 142

INLIST operator 284

Internet Explorer 5

HTML error reporting 432

IP address

specifying for server startup 99

iPlanet server integration 165

ISO 8859-1 122

J

J2EE

application clients 140

applications, typical permissions 287

archives, setting Modify permissions 274

clients 17, 40

deployed objects 139

security roles 299

transactions 144

Java clients

connecting through SSL to server 204

Java security 271

Java Virtual Machine (JVM)

memory statistics 158

JavaMail 161

- jBroker MQ
 - configuring 123
 - debugging 123
 - in clusters 123
 - running 123
 - jBroker ORB
 - debugging 409
 - JDBC
 - access 55
 - debugging 422
 - defined 55
 - tracing 409
 - JMS servers
 - configuring 123
 - debugging 123
 - in clusters 123
 - running 123
 - JNDI 209
 - tree, viewing 142
 - jsessionId parameter 49
 - JSP pages
 - listed in the Deployed Objects panel 140
 - JSP session 49
 - JTS log file 146
 - jvmversion startup option 96
- L**
- L startup option 422
 - l startup option 422
 - language mapping 122
 - languages, multibyte 122
 - LDAP
 - about 213
 - login format 223
 - setting up security 214
 - SSL connection 214
 - system credentials 214
 - Version 2 214
 - Version 3 214
 - licenses
 - accessing information 137
 - adding (NT) 137
 - adding (UNIX) 137
 - deleting 137
 - relicensing 428
 - load balancing 327
 - Load Manager
 - adding to a cluster profile 344
 - defined 328
 - distribution mapping 333
 - function described 332
 - setting failover parameters 355
 - starting in UNIX 339
 - starting in Windows NT 339
 - local group 279
 - locking down
 - application or server 289, 290
 - locking down an application or server 290
 - locking resources 297
 - Locksmith privilege
 - defined 134
 - regaining 422, 428
 - logging
 - creating custom class 400
 - custom class 114
 - database 113
 - displaying in the SMC 151
 - error 114, 407, 421, 422
 - file 113, 421, 422
 - HTTP 114
 - JDBC 422
 - server state 411
 - trace 115
 - turning on 112
 - types defined 112
 - logging in 25
 - requiring 300
 - logging out 26
 - login
 - formats 221
 - resource 419, 429
 - startup option 419, 429
- M**
- Managed Information Bases
 - SNMP term defined 448

Managed Objects

SNMP term defined 448

Management Node

SNMP term defined 448

MAXFREE parameter 309

Maximum result row count property in SMC 162

MAXTHREADS parameter 309

memory cache 312

MIB variables

SNMP term defined 448

-minspan startup option 96

Modify

permission type defined 274

Modify server configuration

permission type defined 273

Monitor options in SMC 146

multibyte encodings 122

multihoming, support for 99

multiple servers on one host 119

N

-n startup option 423

network configurations, samples 34

NIS+

security login format 223

setting up security 220

-nodbcheck startup option 96

-noexitondbcheck startup option 96

-nomonitor startup option 97

-noserverlisteners startup option 97

-nostartagents startup option 97

O

-O startup option 423

Object Identifier

SNMP term defined 448

object security, defaults 288

ODBC

tracing 409

OID

SNMP term defined 448

Oracle

table space allocation 423

ORB settings 116

P

-P startup option 424

-p startup option 97, 424

passwords

creating and editing 131

SilverMaster 424

Path parameters 50

Performance Monitor 146

permissions 298

changing 276

database object 273

defined 272

for J2EE archives 274

restricting 295

types 272

typical for classic applications 285

typical for J2EE applications 287

persistent failure

failover function 337

plug-ins

configuring SilverStream as 159

POP3 server 159

ports

administration 106

configuration, simplified 265

configuration, SSL 265

design 106

e-mail 161

IIOP SSL 118

RSA 244, 251, 265

runtime 106

separate 21, 106

support for session-level failover 118

types 107, 109

prefetch buffers

defined 325

setting 303

prepend path startup option 418

- production environments
 - configurations 28, 34
 - security checklist 292
 - production server 292
 - +profile startup option 95
 - prohibiting access by Anonymous user 293
 - properties file
 - SilverMaster 424
 - protocol, e-mail 161
 - proxy servers
 - about 31
 - response chains 47
 - server configuration 33
- ## Q
- Query parameters 50
- ## R
- r startup option 424, 426
 - RARs
 - viewing deployed 142
 - Read
 - access to SilverMaster database 428
 - permission type defined 274
 - Read Directory Listing
 - permission type defined 273
 - Read Server Configuration
 - permission type defined 273
 - Read Users and Groups
 - permission type defined 273
 - Refresh mode
 - SilverMasterInit 426
 - regaining SilverMaster access 428
 - registered objects
 - defined 368
 - reinitializing SilverMaster 426
 - remote objects 116
 - remote server, administering 135
 - Require user authentication
 - button in SMC 278
 - SilverMaster access startup option 429
 - field in SMC 267
 - SilverMasterInit access startup option 419
 - requiring user authentication 293
 - resources 45
 - defined 45
 - recreated 421
 - refreshed 424
 - response chains, HTTP 47
 - restarting servers in a cluster 347
 - restarting the server 100
 - restricting directory listing 294
 - retry startup option 97
 - RMI
 - about 13, 106, 116, 252
 - name services 117
 - server 117
 - settings 116
 - viewing JNDI tree 142
 - RMI Port
 - with Load Manager 346
 - robots
 - robots.txt 301
 - specifying access 301
 - row-level security
 - advanced expressions 283
 - RSA
 - certificates, creating and installing 240
 - encryption 202
 - RSA port
 - enabling and changing 251
 - in cluster dispatcher 347
 - simplified 244, 251, 265
 - running SilverStream server as a service 101
 - runtime port 106
- ## S
- search engines, specifying access 301
 - secure
 - application 290
 - connections, establishing to a SilverStream server 204
 - Secure Site IDs 229
 - secure_application_sample.xml 289

- secure_cluster_sample.xml 291
- secure_server_sample.xml 289
- security
 - advanced expressions for access control 280
 - checklist 292
 - database access 200
 - expressions 220, 280
 - functions 203
 - getAccess 196
 - levels 292
 - locking down an application or server 289
 - Locksmith privilege 134
 - provider systems 208
 - SilverStream server 193
 - simple list for access control 279, 280
 - specifying access to robots 301
 - Windows NT 211
 - WSI module 193
- security provider systems 221
 - about 208
 - accessing 209
 - LDAP, defined 208
 - NIS+ security 208
 - SilverStream 208
 - updating lists of users and groups 210, 268
 - Windows NT 208
- security providers
 - enumerating 387
- Security resource timeout, field in SMC 210, 268
- security roles
 - mapping 299
- Select
 - permission type defined 275
- separate ports 106
 - firewall 107
 - for runtime, design, and administration 293
- server
 - adding to existing cluster profile 350
 - authentication 428
 - charting statistics 146
 - configuration setting 296
 - custom logging class 400
 - defaults 288
 - encoding 122
 - integration with IIS 165
 - integration with iPlanet 165
 - list of properties 438
 - logging the state 411
 - removing from existing cluster profile 350
 - running more than one on a host 119
 - specifying user that starts in UNIX 111
 - troubleshooting starting 413
 - viewing statistics 152
- server administration
 - e-mail 159
 - remote server administration 135
 - restarting 100
 - setting the listener port 111
 - starting in Windows NT 92
- Server Administration API
 - see Administration API
- server certificates 228
 - verified 264
- server clusters
 - adding servers in UNIX 341
 - adding servers in Windows NT 340
 - administering 348
 - Cache Manager 330
 - certificate install 358
 - components overview 328
 - configuration options 330
 - configuration requirements 329
 - creating cluster profile 342
 - defined 327
 - Dispatcher function 333
 - distribution mapping 333
 - failover 336
 - Load Manager 332
 - restarting servers to activate 347
 - setup overview 337
 - starting the software components 338
- server configuration
 - about 27
 - classic development environment 29
 - multiple independent classic development environments 29
 - production environment 28
 - running more than one server on a host 119
 - shared classic development environment 31

- server content cache
 - setting parameters 312
- server relative load weight, specifying 353
- server statistics
 - sessions 153
 - summaries 157
 - threads 154
 - viewing 152
- server URL, SilverStream server, default 140
- service
 - running server as 101
- servlets 140
- session 49
- URLs 286
- session ID 48
- session management
 - defined 48
- session state 48
- session timeout
 - setting 303
- session tracking 48
- session-level failover
 - ports 118
- sessions
 - defined 306
 - statistics 153
 - timeout parameter 304
- Set Permissions 273, 274, 285
- SetSecurity command, SilverCmd
 - sample input files 289
- setting up ports 106
- ShiftJIS 122
- Silver Security 127
- SilverCmd
 - about 286
 - locking the server down 291
- SilverJ2EEClient 40
- SilverJunction
 - using 159
- SilverMaster
 - access 297
 - account 424, 425
 - boot environment 420
 - configuring in a cluster profile 343
 - creating 427
 - database access 297
 - database system tables 455
 - debugging 425
 - defined 58
 - functions of 58
 - moving 59
 - properties 425
 - properties file 424
 - recreating 426
 - refreshing 426
 - server cluster component 328
 - specifying in SMC 112
 - standard system tables 456
 - system data and resources, new 421
 - table space 423
 - troubleshooting 416, 433
- SilverMasterInit
 - about 416
 - Full mode 417, 421
 - options 417
 - Refresh mode 417, 424, 426
 - starting 427
 - startup parameters 417
- SilverMonitor
 - about 414
 - in server cluster 336
 - startup parameters 415
- SilverServiceUtil 101
- SilverStream applications
 - database URLs 457
- SilverStream HTML error page
 - displaying in Internet Explorer 5 432
- SilverStream Management Console (SMC) 367
 - about 20
 - login 24
 - logout 24
 - online help 26
 - panels 1
 - quick reference 1
 - starting 21, 429
 - user interface 23
 - using 20
- SilverStream Performance Monitor 146
- SilverStream resources
 - defined 45

- SilverStream security
 - defined 208
 - login format 222
 - SilverStream server
 - access 289
 - cluster component 328
 - defined 13
 - integrating with Web server 159
 - logging in 419, 429
 - services and daemons 101
 - starting 413, 429
 - simple list
 - using for access control 279, 280
 - SJIS 122
 - SNMP (Simple Network Management Protocol)
 - Extension Agent 448
 - implementation overview 445
 - Managed Information Bases 448
 - Managed Objects 448
 - Management Node 448
 - Object Identifier 448
 - running as a service 450
 - setting up in SilverStream 450
 - SNMP Agent 448
 - socket exceptions
 - reported in NT application log 433
 - SQL debugging 409
 - SSL
 - about 200
 - certificates 264
 - creating connection to server 204
 - encryption 118
 - for remote objects 118
 - remote objects 116
 - using cipher suites 253
 - with LDAP 214
 - stack overflows, handling 429
 - StackOverflowError, handling 429
 - stacks, changing size of 429
 - starting the server 429
 - as a daemon 91
 - as a service 91
 - startup properties 122, 437
 - troubleshooting 413
 - Windows NT 92
 - statistics
 - charting 146
 - viewing 152
 - Subagent
 - SNMP term defined 448
 - Super User 134
 - Synchronize database schema, button in SMC 67
 - synchronizing database schema
 - troubleshooting 414
 - System database properties, button in SMC 68
 - system login credentials 214
 - system tables (SilverStream)
 - database 455
 - in SilverMaster 456
- ## T
- tables
 - using a subset of 62
 - threads
 - about 306
 - listing 432
 - statistics 154
 - three-tiered communications 14
 - advantages of 15
 - timeout on server request
 - setting 303
 - Timeout on server request, field in SMC 304
 - total memory 158
 - trace logging 115
 - trace startup option 97
 - tracking sessions 48
 - transactions
 - JTS log file 146
 - log file max size 145
 - recovering 144
 - resource recovery 146
 - timeout 146
 - troubleshooting
 - browser issues 432
 - low-level debugging 408
 - server hang 432
 - SilverMaster access 428
 - starting the server 413
 - using server command shell 411

U

- U startup option 425
- UNIX
 - starting server as a daemon 91
 - user account that starts server 111
- unlocking resources 422, 428
- URLs
 - about 44
 - default database 140
 - default server 140
 - encoding 122
 - rewriting 49
 - SilverStream database 457
- user login formats 221
- user-defined logging 114
- user-defined logging class 400
- Username for server, field in SMC 111
- users
 - accessing 220
 - adding 132
 - adding to groups 132
 - adding to the registry 130
 - changing passwords 131
 - editing user properties 131
 - enumerating 387
 - getting group membership of current user 399
 - getting name of current user 399
 - identifying using Administration API 399
- utf-8 122
- UUID support 281

V

- v startup option 425
- verbose output 425
- +verbose startup option 95
- VM, specifying at startup 98

W

- W startup option 425
- warn on cookies, browser option 48
- WARs
 - security roles 299

- Watcher 411
- web crawlers, specifying access 301
- Web server integration
 - see WSI
- Windows NT
 - login format 222
 - security 279
 - security provider system 208
 - setting up NT security 211
 - starting server as a service 91
 - starting the server 92
- Write
 - permission type defined 274
- WSI (Web server integration) modules
 - about 165
 - AgWSIUser 191
 - authentication 190
 - connection pooling 190
 - enabling 172, 182
 - examples 166
 - for IIS 172
 - for iPlanet 172
 - for Solaris 172, 184
 - for Windows 184
 - in cluster 171
 - installing 172
 - NTLM authentication 191
 - security 190
 - URL forwarding 177
- WSI configuration file 174

X

- x startup option 425
- XML
 - command files 286
- +Xms startup option 95
- +Xmx startup option 95