# CA SiteMinder® Secure Proxy Server

## Administration Guide

### Release 12.51

# CA Technologies Product References

This document references the following CA Technologies products:

- CA SiteMinder® for Secure Proxy Server
- CA SiteMinder®

# Contact CA Technologies

**Contact CA Support**

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

**Providing Feedback About Product Documentation**

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

# Documentation Changes

Updates have been made to the following areas since the last release of this documentation: Upgrade of SPS, Set Log Files and Command-line Help to Another Language, General Server Settings in the server.conf File, Proxy and Redirect Settings in the server.conf File, Proxy Rules, SSL Configuration, and Troubleshooting.

Affected topics include the following:

- Installation Prerequisites: Updated topic to cover the requirement to install Legacy Software Development package on the RHEL 5.5 machine.
- Upgrade SPS: Updated topic to cover the necessity to take a back up of configured filters and customized session schemes before an upgrade.
- Additional Tasks for Upgrades: Updated topic to cover the requirement to enable SSL on Linux after an upgrade.
- New Language Settings Introduced: You can set the log files and command-line help to other languages.
- Ignoring the SSL Exceptions: New topic to cover the new parameter in the server.conf file.
- Proxy Service Configuration: Updated topic to cover the new parameter in the server.conf file for managing proxy service.
- nete:rule and nete:result: Updated topic to cover the usage of tags in the proxy rules.
- Configure SSL for SPS: New topic to cover the changes to the SSL configuration in SPS.
- Resource Request Fails: Updated topic to cover the different types of log files that can be used for troubleshooting the issue.

**More information:**

Prerequisites (see page 29)
Upgrade SPS (see page 33)
Additional Tasks for Upgrades (see page 36)
Configure SSL for CA SiteMinder SPS (see page 206)
Ignoring the SSL Exceptions (see page 95)
Proxy Service Configuration (see page 103)

# Contents


## Chapter 1: SiteMinder Secure Proxy Server Overview 13

Introduction to the SPS ..... 13
- Proxy Server Architecture ..... 13
- Traditional Reverse Proxy Server Architecture ..... 14
- SPS Architecture ..... 14
- SPS Components ..... 16

SPS Product Features ..... 18
SPS Product Limitations ..... 19
SPS in an Enterprise ..... 20
- SPS as a Centralized Access Control Filter ..... 21
- SPS Support for Cookieless Sessions ..... 22

SPS Support for Extranet Access Control ..... 24

## Chapter 2: Install, Upgrade, and Configure the SPS 27

Install, Upgrade, and Configure SPS ..... 27
- Prerequisites ..... 29
- Installation Worksheet ..... 29
- Install SPS ..... 31
- Install Multiple Instances of SPS ..... 32
- Upgrade SPS ..... 33
- Configure SPS ..... 34
- Additional Tasks for Upgrades ..... 36
- Protect the Administrative User Interface ..... 37
- Launch the Administrative User Interface ..... 38
- Silent Installation and Configuration ..... 39
- Uninstall SPS ..... 39
- Start SPS in Single or Multi Process Mode ..... 40
- Modify the Default Location of the SiteMinder Forms ..... 41

How to Set Log Files, and Command-line Help to Another Language ..... 42
- Determine the IANA Code for Your Language ..... 43
- Environment Variables ..... 44

## Chapter 3: FIPS-140 Support 49

FIPS Support Overview ..... 49
Configuration Process for FIPS ONLY Mode ..... 50
Migration to FIPS MIGRATE Mode ..... 50

Migration to FIPS ONLY Mode ..... 51

## Chapter 4: Using the SPS with Federation Security Services 53

Federation Security Services Introduction ..... 53
SPS Use Cases in a SiteMinder Federated Environment ..... 53
Use Case 1: Single Sign-on Based on Account Linking ..... 54
Use Case 2: Single Sign-on Based on User Attribute Profiles ..... 55
Use Case 3: Single Sign-on with No Local User Account ..... 56
Use Case 4: Extended Networks ..... 57
SPS Roles in a SiteMinder Federated Environment ..... 58
Solutions for SPS Use Cases ..... 58
Solution 1: SSO Based on Account Linking ..... 58
Solution 2: SSO Using User Attribute Profiles ..... 61
Solution 3: SSO with No Local User Account ..... 63
Solution 4: SSO in an Extended Network ..... 65
Cookieless Federation ..... 67
Enable Cookieless Federation at the Consuming Side ..... 69
SPS as a Web Agent Replacement ..... 69
Prerequisites for Using the SPS as a Web Agent Replacement ..... 70
Configuring the SPS as a Web Agent Replacement for Federation ..... 71
SPS as a Federation Gateway ..... 71
Prerequisites for Using the Federation Gateway ..... 73
Configuring the SPS Federation Gateway ..... 73
Limitations of the SPS Federation Gateway ..... 74

## Chapter 5: Security Zones on SPS 75

Overview Single Sign-on Security Zones ..... 75
Security Zones Benefits ..... 76
Security Zone Basic Use Case ..... 77
Parameters for Security Zones ..... 77
Configure the SPS Security Zones ..... 79

## Chapter 6: Configuring the Apache Web Server 81

Apache Web Server Configuration File ..... 81

## Chapter 7: Configuring the SPS Server Settings 83

SPS server.conf File Overview ..... 83
Modifying the server.conf File ..... 84
General Server Settings in the server.conf File ..... 84

HTTP Connection Parameters ....84
Tomcat Tuning Parameters in the server.conf File ....85
Resolve Differences in Cookie Specification for Different Version of Tomcat ....86
Parsing the Equal Sign in a Cookie ....87
Federation Settings in the server.conf File ....87
HttpClient Logging ....88
SSL Settings in the server.conf File ....90
Setting for Special Characters within the Cookie ....94
Select Character Set for Code Headers ....94
Caching POST Data ....95
Ignoring the SSL Exceptions ....95
Custom Error Pages Parameters ....96
Enable Custom Error Messages ....97
Default Custom Error Pages ....98
Session Store Settings in the server.conf File ....101
Service Dispatcher Settings in the server.conf File ....102
Proxy and Redirect Settings in the server.conf File ....103
Proxy Service Configuration ....103
Connection Pooling Recommendations ....107
Redirect Service Configuration ....108
Connection-oriented Connection Pools Configuration ....109
Session Scheme Settings in the server.conf File ....110
Establishing a User Session ....111
Default Session Scheme ....111
SSL ID Session Scheme ....113
IP Address Session Scheme ....114
Mini-cookies Session Scheme ....114
Simple URL Rewriting Session Scheme ....115
Wireless Device ID Session Scheme ....118
Uses for Each Session Scheme ....119
Multiple Session Schemes for Virtual Hosts ....120
Deleting Attribute Cookies for Cookieless Federation ....121
User Agent Settings in the Server.conf ....121
Nokia User Agent Settings ....122
Virtual Host Settings in the server.conf File ....123
Setting Virtual Host Cookie Path and Domain to the Correct URI ....124
Preserve the HOST Header File ....125
Handling Large Files Using Data Blocks ....125
Session Scheme Mapping for the Default Virtual Host ....127
Web Agent Settings for the Default Virtual Host ....128
Handling Redirects by Destination Servers ....129
Virtual Host Names Configuration ....130

Override Default Values for Virtual Hosts ..... 131

## Chapter 8: Configuring the SPS Log Settings 133

SPS logger.properties File Overview ..... 133
Modifying the logger.properties File ..... 133
Logging Settings ..... 134
SvrConsoleAppender Settings ..... 134
SvrFileAppender Settings ..... 135
Log Settings ..... 136
Log Rolling Settings ..... 137
Modify ServerPath in WebAgent.conf for Logging ..... 138

## Chapter 9: Configuring Proxy Rules 141

Proxy Rules Overview ..... 141
Planning Routes for Incoming Requests ..... 141
Proxy Rules Terminology ..... 143
Establish a Proxy Rules Configuration File ..... 144
Proxy Rules DTD ..... 145
nete:proxyrules ..... 145
nete:case ..... 147
nete:cond ..... 149
nete:default ..... 151
nete:forward ..... 152
nete:redirect ..... 153
nete:local ..... 154
nete:xprcond ..... 154
How nete:xprcond Elements Works ..... 156
Regular Expression Syntax ..... 156
Regular Expression Examples in nete:rule and nete:result ..... 159
Header Values in Forwards, Redirects, and Results Filters ..... 160
Dynamic Header Value in a nete:forward ..... 161
Dynamic Header Value in a nete:redirect ..... 161
Dynamic Header Value in a nete:result ..... 161
Response Handling ..... 162
Modify Proxy Rules ..... 162
Sample Proxy Rules Configuration Files ..... 162
Proxy Rules Example—Routing Requests by Virtual Host ..... 163
Proxy Rules Example—Routing Requests by Header Value ..... 164
Proxy Rules Example—Routing Requests by Device Type ..... 165
Proxy Rules Example—Routing Requests with URIs ..... 165
Proxy Rules Example—Routing Requests by File Extension ..... 166

Proxy Rules Example—Routing Requests with Nested Conditions ..... 166
Proxy Rules Example—Using Regular Expression in Proxy Rules ..... 167
Proxy Rules Example—Routing Requests by Cookie Existence ..... 168
Proxy Rules Example—Routing Requests by Cookie Value ..... 168

## Chapter 10: Deploying the SPS 169

SPS Deployment in an Enterprise ..... 169
Sticky-Bit Load Balancing ..... 170
Proxying to Trusted Sites vs. Non-Trusted Sites ..... 171
Configuring Virtual Hosts for the SPS ..... 171
Implementing Session Scheme Mappings for Multiple Virtual Hosts ..... 172

## Chapter 11: Integrating the SPS with SiteMinder 177

How the SPS Interacts with SiteMinder ..... 177
Authentication Scheme Considerations ..... 178
Proxy-Specific WebAgent.conf Settings ..... 179
Avoiding Policy Conflicts with Destination Server Web Agents ..... 180
Configuring SiteMinder Rules that Redirect Users ..... 182
SPS and SharePoint Resources ..... 183
SPS and ERP Resources ..... 183
Password Services for SPS ..... 184
Configure a Password Policy for SPS ..... 185
Verify Password Services for SPS ..... 186
Configuring Managed Self Registration for the SPS ..... 186
Install a Web Agent for MSR ..... 187
Configure the Policy Server User Interface for MSR ..... 188
Proxy Rules for an MSR Request ..... 188
Firewall Considerations ..... 189
Keep Alive and Connection Pooling ..... 189
HTTP Header Configuration for Sun Java Web Servers ..... 189
HTTP Header for SiteMinder Processing with SPS ..... 190
Handling Encoded URLs ..... 190

## Chapter 12: Configure SPS to Support the SessionLinker 191

Configure the SPS to Support the SessionLinker ..... 191
How the SessionLinker Works ..... 192
Enable the SessionLinker ..... 194
Create the NPS_Session_Linker ACO ..... 195

## Appendix A: Working with Cookies 197

Troubleshooting ..... 200

## Chapter 13: SSL and the Secure Proxy Server 201

Keys and Server Certificates Management ..... 201
Generate a Private RSA Key ..... 202
RSA Key Decryption ..... 203
RSA Key Encryption ..... 203
Modify the Passphrase for an RSA Key ..... 203
Create Certificate Signing Request ..... 204
Create a Self-Signed Certificate ..... 204
Obtain Certificate Signed by a CA ..... 205
Install a Signed Certificate ..... 205
Configure SSL for CA SiteMinder SPS ..... 206
Enable SSL for Virtual Hosts ..... 207

## Chapter 14: Configure SPS to Support Integrated Windows Authentication 209

Configure SPS to Support Integrated Windows Authentication ..... 209
Windows Authentication Schemes ..... 211
Kerberos Authentication Schemes ..... 214

## Chapter 15: Data Monitoring Using CA Wily Introscope 231

Data Monitoring Using CA Wily Introscope ..... 231
Enable Data Monitoring ..... 233
Monitor Web Agents with the OneView Monitor ..... 233

## Chapter 16: Operating System Tuning 235

Tune the Shared Memory Segments ..... 235
How to Tune the Solaris 10 Resource Controls ..... 237

## Chapter 17: SPS APIs 239

Session Scheme API ..... 239
Overview of Session Scheme API Processing ..... 239
Implement a Custom Session Scheme ..... 243
Configure Rewritable Session Schemes ..... 244
Use an IP Address Session Scheme ..... 245
Session Storage API ..... 246
Filter API Overview ..... 246

How SPS Processes Custom Filters ..... 247
Associate Custom Filters to Proxy Rules ..... 247
Filter API Class File ..... 247
ProxyFilter Interface ..... 248
BaseProxyFilter Abstract Implementation ..... 248
ProxyFilterConfig Interface ..... 250
ProxyResponse Interface ..... 251
ProxyFilterException Class ..... 252
ProxyRequest Interface ..... 253
Implement a Filter ..... 255
Filter API Example ..... 256
Using a Filter to Rewrite Absolute Links in a Requested Page ..... 256

## Chapter 18: Troubleshooting 257

Unable to Start Apache on UNIX systems ..... 257
Non-english Input Characters Contain Junk Characters ..... 258
Unable to Log Federation Web Services Errors ..... 258
DNS Caching in the SPS ..... 259
Resource Request Fails ..... 259
Configure spsagent Logs ..... 260
Configure SPSAgentTrace Logs ..... 261
Configure the mod_jk.log File ..... 262
Configure the httpclient.log File ..... 262
No Root Permissions ..... 263
Cannot Start the SPS Server ..... 263
Cannot Access the SPS with a Browser ..... 264
Unknown Server Name ..... 264
Issues Configuring Virtual Hosts ..... 265
Command not found Error Received ..... 265
SPS Not Forwarding Requests ..... 265
SPS and SharePoint ..... 266

## Index 267

# Chapter 1: SiteMinder Secure Proxy Server Overview

This section contains the following topics:

Introduction to the SPS (see page 13)
SPS Product Features (see page 18)
SPS Product Limitations (see page 19)
SPS in an Enterprise (see page 20)
SPS Support for Extranet Access Control (see page 24)

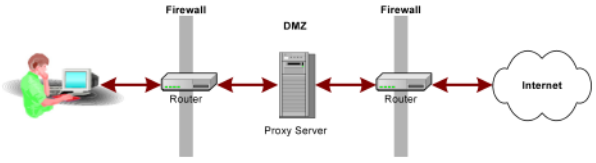
## Introduction to the SPS

The SPS is a stand-alone server that provides a proxy-based solution for access control. The SPS employs a proxy engine that provides a network gateway for the enterprise and supports multiple session schemes that do not rely on traditional cookie-based technology.

### Proxy Server Architecture

A traditional proxy server is located between a firewall and an internal network and provides caching of resources and security for the users on the internal network. Traditional proxy servers act as a proxy on behalf of a group of users for all resources on the Internet.

The following illustration shows a proxy server configuration. The proxy server caches frequently accessed resources so that requests for those resources are handled faster in the Demilitarized Zone (DMZ).



## Traditional Reverse Proxy Server Architecture

A reverse proxy server represents one or more destination servers. A typical use of reverse proxy architecture provides:

- Cached resources
- Security
- SSL acceleration
- Load balancing

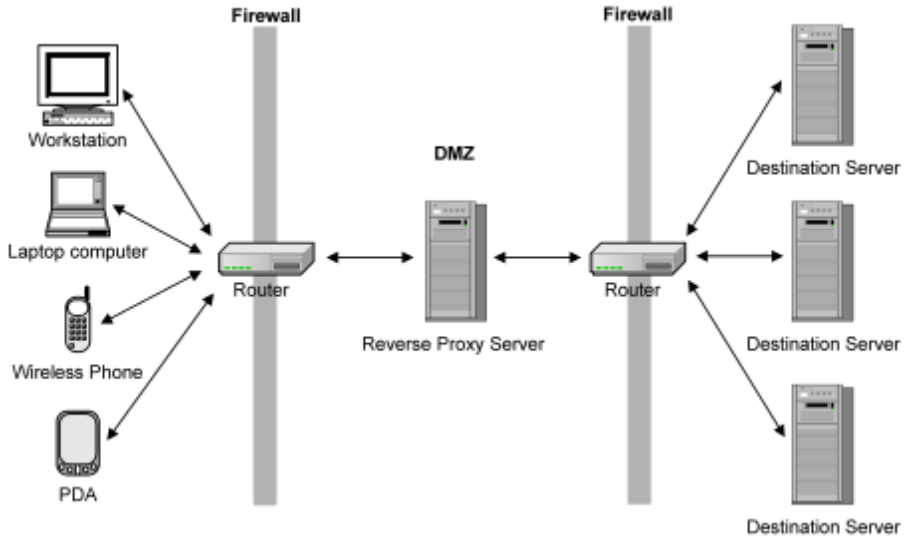
Rather than requesting a resource directly from a destination server, the reverse proxy server caches much of the content from the destination servers, providing ready access for users. This type of proxy server deployment is considered a reverse proxy, because the proxy is transparent to the user and works on behalf of the destination servers in the enterprise. Multiple reverse proxy servers can be used for load balancing and can also provide some SSL acceleration for HTTPS requests. A reverse proxy server also provides an additional layer of security by insulating destination servers that reside behind the DMZ.

## SPS Architecture

The SPS is not a traditional reverse proxy solution, because it does not provide resource caching. The SPS serves as a single gateway for access to enterprise resources, regardless of the method of network access.

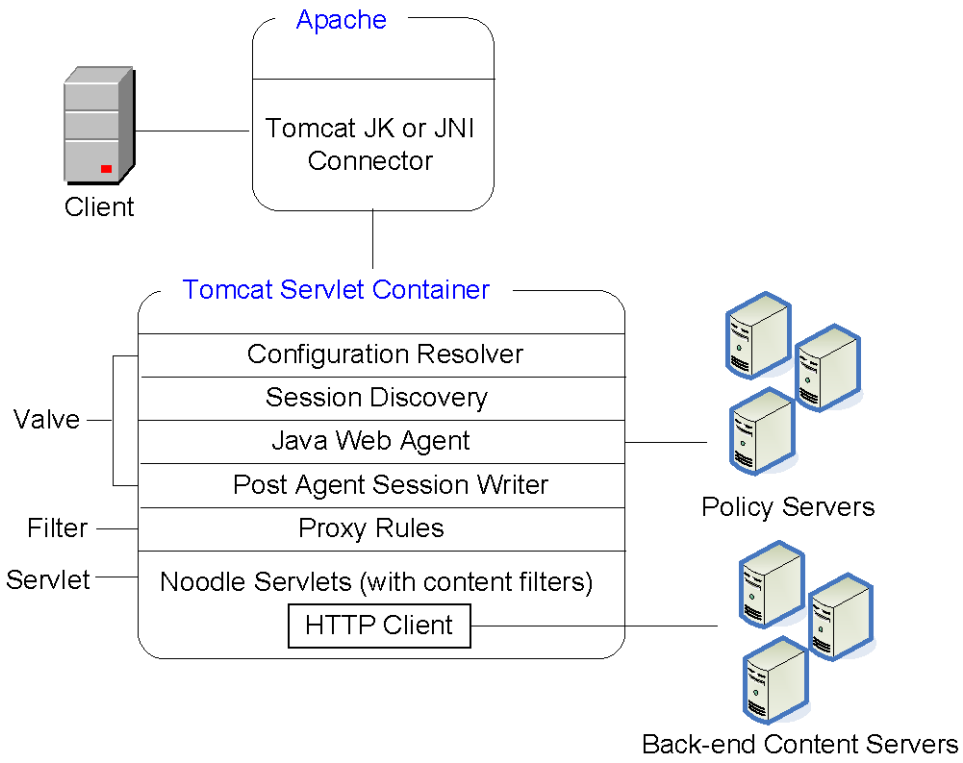
A set of configurable proxy rules determines how the SPS handles a user request. Users can access resources through multiple session schemes based on mapping between user agent types and virtual hosts. Requests can be routed to different destination servers based on the type of device being used to access the network.

The following illustration shows a configuration of the SPS. Users access the SPS using various devices. The SPS determines the session scheme to create based on the access device, and then forwards or redirects requests to the appropriate destination servers. Users are not aware that there is a reverse proxy server in the enterprise.

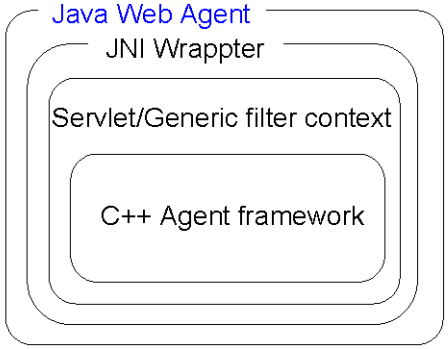


## SPS Components

A stand-alone SPS consists of an HTTP listener (Apache) and a Tomcat servlet container, as shown in the following illustration:




**Java Web Agent Components**



The SPS architecture includes the following components:

**Apache**

SPS uses the open source Apache Web server to act as the HTTP listener for incoming requests. An additional component, mod_jk (1.2.18), acts as the Tomcat connector, which enables communication between the Apache Web server and Tomcat using the Apache JServ protocol (AJP).

**Tomcat**

The Tomcat server provides Tomcat servlet container for the SPS. The Tomcat initialization is customized so it does not allow deployment of any external applications or servlets. The standard Tomcat xml (server.xml) is not used for initialization. The components inside the Tomcat container of the SPS include the following:

**Configuration Resolver ProxyBootstrap**

The configuration resolver proxybootstrap is responsible for reading the SPS configuration from the server.conf file and initializes the SPS.

**Session Discovery**

The session discovery component analyzes the incoming requests for extracting the SPS session information. Depending on the user agent type and the virtual host being used, this component uses the appropriate session scheme for extracting the SPS session information.

If the request uses an existing SPS session, this component uses the SPS session identifier contained in the request to extract the corresponding SiteMinder session from the in-memory session store. The SPS passes the SiteMinder session to the Java Web Agent for session validation. If the request does not contain an existing SPS session, this component passes the request on to the Java Web Agent for user authentication.

**Java Web Agent**

The Java Web Agent, together with the SiteMinder Policy Server, authenticates and authorizes the user.

**Post Agent Session Writer**

The post Agent session writer performs additional processing for cookieless session schemes. After the Java Web Agent authenticates and authorizes the user and creates a SiteMinder session, this component creates a SPS session identifier. This identifier is attached to the SiteMinder session created by the Java Web Agent.

This session identifier is then maintained in the in-memory session store of the SPS. In addition to maintaining the session in the session store, this component transforms the URI. For example, the Post Agent Session Writer manipulates the URI for the simple_url session scheme.

**Proxy Rules Servlet Filter**

The proxy rules servlet filter loads the proxy rules from the proxyrules.xml file. Depending upon the incoming request and the proxy rule, the request is forwarded or redirected to the backend server. If the request is forwarded, an open source component Noodle is used.

Any changes made to the proxy rules do not require a restart for the changes to take effect. The proxyrules are reloaded when there is any change in the proxyrules.xml file.

**Noodle Servlet**

The Noodle servlet forwards requests to the backend servers. Noodle also supports the use of proxy pre-filters which enable the request to be modified before sending the same to the backend server. Similarly support for proxy post-filters is also available which enables modification of the response received from the backend server before sending it back to the user client.

**HTTP Client**

The HTTP client sends requests to the backend server and receives responses from the backend server.

# SPS Product Features

SPS offers the following features:

**Access Control for HTTP and HTTPS Requests**

SPS allows you to control the flow of HTTP and HTTPS requests to and from destination servers using an embedded SiteMinder web agent. In addition, SPS is fully integrated with SiteMinder to manage e-business transactions.

**Single Sign-on**

The embedded web agent in SPS enables single sign-on (SSO) across an enterprise, including SSO with SiteMinder Web agents that can be installed on destination servers within the enterprise.

**Multiple Session Schemes**

A session scheme is a method for maintaining the identity of a user after authentication. Core SiteMinder products use cookies to maintain a session. SPS, however, can maintain sessions based on SSL ID, mini-cookies, device IDs for handheld devices, URL rewriting, IP addresses, and schemes created using the Session Scheme API.

**Session Storage**

SPS is equipped with an in-memory session store. The session store maintains session information. SPS uses a token, such as a mini-cookie or SSL ID, to refer to the session information in the session store. Multiple session schemes and in-memory session storage enable SPS to provide a solution for e-business management beyond computers, wireless devices such as PDAs and wireless phones.

**Cookieless Single Sign-on**

Some enterprises prefer solutions that do not use cookie technology. Because of the session schemes and the session store built into SPS, it offers a solution to enterprises that want an alternative to cookie-based session management.

**Intelligent Proxy Rules**

Proxy rules allow you to configure different paths for fulfilling client requests from SPS based on characteristics such as the requested virtual host or URI string. The proxy engine interprets a set of proxy rules to determine how to handle user requests.

**Centralized Access Control Management**

By providing a single gateway for network resources, SPS separates the corporate network and centralizes access control.

**Enterprise Class Architecture**

SPS is designed to be scalable, manageable, and extensible.

# SPS Product Limitations

The following conditions apply to SPS:

- SPS is not a plug-in to another Web server. SPS is a fully supported, stand-alone server.
- SPS does not support local content. The ability to place content on SPS is not exposed, and SPS does not support proxy rules for providing access to local content.
- SPS does not support having the Web server on the same system as SPS. If the two are set up on the same system, the Web server is accessible from the Internet. Security is not sure with this configuration.
- SPS provides its own session storage. However, the session store has no public APIs for use as a general session server.

- In some enterprises that use SPS, destination servers can also have SiteMinder web agents or application server agents installed. When policies for SPS agent are inconsistent with policies for the agent installed on the destination server, SPS can pass responses back to the invoking client. Because SPS does not provide warnings about inconsistencies in processing such policies, use care when setting up SiteMinder policies in such environments.
- SPS makes a new request to the destination server for every request that it receives. All caching directives are ignored.
- In the simple-url session scheme, SPS does not rewrite URLs embedded in or resulting from JavaScript.
- The simple_url session scheme does not preserve the POST data when posting to a protected resource.

# SPS in an Enterprise

Enterprises that provide access to network resources for employees, customers, and partners face a number of challenges, including:

- Directing requests to appropriate services
- Verifying user identities and establishing entitlements
- Maintaining sessions for authorized users
- Providing centralized access control configuration
- Supporting multiple device types
- Employing flexible and secure architectures

SiteMinder provides solutions to many of these challenges, including authentication and authorization of users, and a complex engine for evaluating user entitlements. SPS further expands the benefits of core Policy Server and Web Agent functionality by providing a reverse proxy solution.

This reverse proxy solution adds the following capabilities:

- Inter-operability with existing SiteMinder Web Agents
- Cookieless single sign-on and sessions storage

- Centralized configuration through proxy rules
- Multiple options for maintaining sessions
- Multiple device support

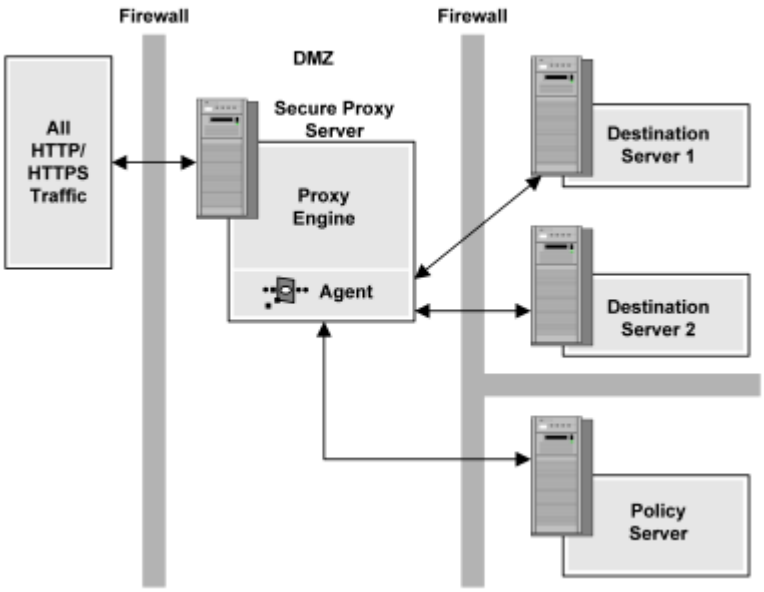
You can deploy SPS in an enterprise to serve the following functions:

- Act as a centralized access control filter
- Support cookieless session schemes
- Support extranet access control

## SPS as a Centralized Access Control Filter

To limit access to destination servers and provide a central entry point to the network, SPS can be placed in front of all destination servers in the enterprise. HTTP or HTTPS requests that come into the enterprise can be filtered through SPS, and forwarded to the appropriate destination server for fulfillment.

The following illustration shows how SPS handles all HTTP and HTTPS requests.



Destination servers that contain content do not require SiteMinder Web Agents. The only network element that resides behind the first firewall is SPS. All users must be authenticated and authorized by SiteMinder residing behind the second firewall. The destination servers provide content after SiteMinder and SPS verify user entitlements.

This deployment provides the following benefits:

- Centralizes configuration through proxy rules

  SPS uses proxy rules defined in XML configuration files to establish how SPS handles requests. Proxy rules can be based on:

  - Host name
  - URI substring
  - HTTP header
  - SiteMinder header
  - Regular Expressions based on URI

  In addition, the conditions for proxy rules can be nested to create rules that incorporate multiple conditions.

- Directs requests to appropriate services

  All HTTP and HTTPS traffic passes through SPS. Based on the proxy rules established for SPS, requests are forwarded to the appropriate destination servers for fulfillment.

- Verifies user identities and establishes entitlements

  SPS uses the built-in web agent to communicate with SiteMinder and perform authentication and authorization of requests.

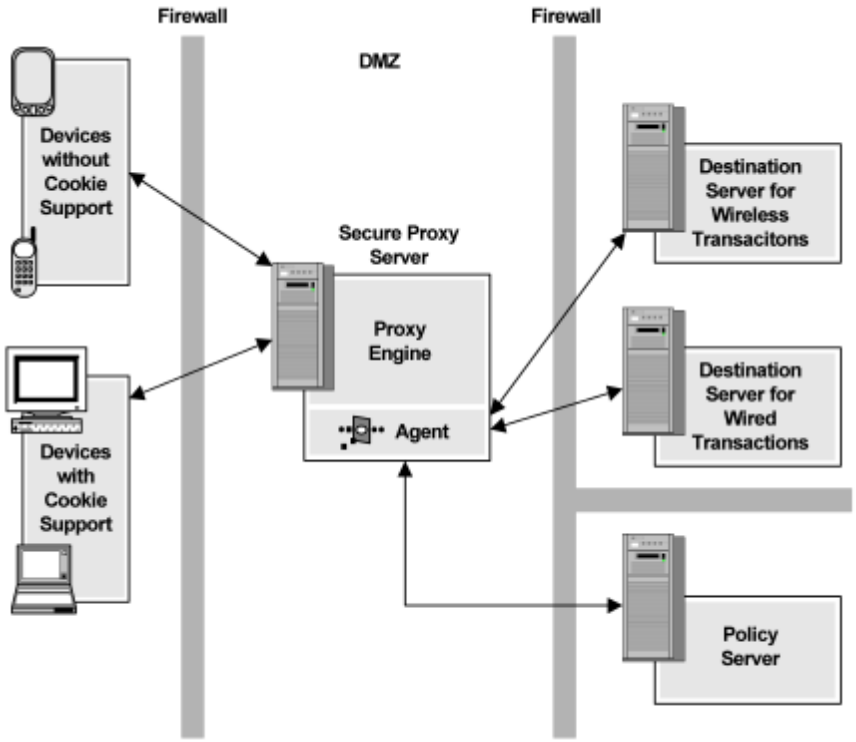
## SPS Support for Cookieless Sessions

Most solutions use cookie technology. However, when accessing resources over HTTP or HTTPS, some enterprises want alternatives for establishing and maintaining a user session and provide single sign-on with a cookieless solution.

SPS provides an in-memory session store and allows the use of any of the following cookieless session schemes:

- Mini-cookies (uses small cookies in place of standard SiteMinder cookies)
- SSL ID
- Device ID
- Simple URL Rewriting
- IP Address

The following illustration shows a deployment in which SPS provides a combination of standard sessions using cookies and sessions without cookies:




The deployment shown in the previous illustration provides the following benefits:

- Supports multiple device types

  Through a set of proxy rules, SPS forwards, or redirects, requests based on the type of device issuing the requests. For example, all initial requests can be directed at SPS, which forwards requests to destination servers based on device types. Browser requests can be redirected to destination servers, and SPS handles wireless requests.

- Maintains sessions for authorized users

  Both standard SiteMinder cookies and cookieless session schemes are employed for maintaining user sessions. Session schemes are assigned based on user agent type for each virtual host. For example, all users accessing the network through web browsers are assigned to a standard cookie session scheme. Users accessing resource through a wireless telephone are assigned to a device ID session scheme.

- Provides cookieless single sign-on and session storage

  Through an in-memory session store and the support of multiple session schemes, SPS provides alternatives to cookie-based sessions. SPS maintains session information in the session store and returns a token. This token is exchanged with all transactions, allowing SPS to match the token to the session information captured in the session store.

- Offers multiple options for maintaining sessions

### Cookieless Session Scheme in a Federation Environment

SPS, with its built-in handling of cookieless session schemes, enables it to be deployed in environments where the user agent, such as a wireless device, does not support traditional SiteMinder cookies.

If you deploy SPS in a SiteMinder federation security services environment, the following process is enforced when a user request is received:

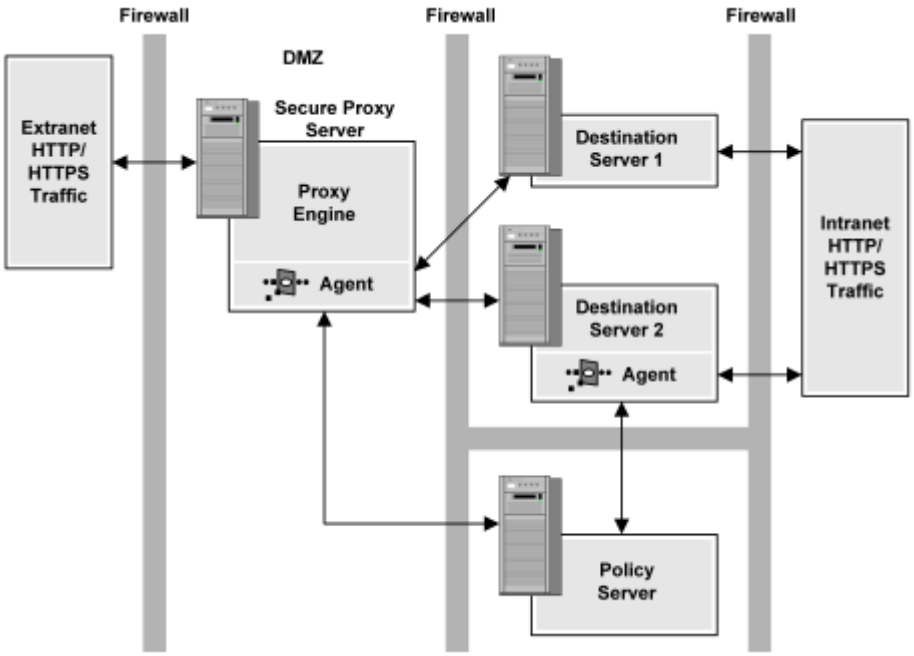
1. SPS receives a request for a federated resource. The request is directed to the Federation Web Services (FWS) application at the site producing assertions.
2. SPS verifies if cookieless federation is enabled for the virtual host requesting the redirect.
3. If a cookieless scheme is being used, SPS removes the session key (SMSESSION cookie) for the current session.
4. SPS sends the user to the link provided by the FWS redirect.

If SPS is using a rewritable session scheme such as simple_url session scheme, SPS rewrites the redirect response to include the session key information in the redirected URL.

## SPS Support for Extranet Access Control

Another deployment of SPS provides access control for external users, but allows direct access to destination servers for internal users. If a destination server provides access to secure applications for individuals within the enterprise, a standard SiteMinder Web Agent can be installed on the destination server to provide access control. Users who are properly authenticated through SPS can use single sign-on.

The following illustration shows an example of an extranet network deployment.




This deployment provides the following benefits:

- Directs requests from extranet sources

  All extranet traffic passes through SPS and is forwarded to the appropriate destination server once users are authenticated and authorized for requested resources.

- Employs flexible architectures

  All information is located behind multiple firewalls to protect from extranet attacks. Information that is appropriate for intranet users does not incur the overhead of agent to SiteMinder communication. SiteMinder can still protect sensitive resources, however.

- Provides interoperability between Web Agents

  SPS and intranet Web Agents use the same Policy Server and provide single sign-on for authorized extranet users on all destination servers.

# Chapter 2: Install, Upgrade, and Configure the SPS

This section contains the following topics:

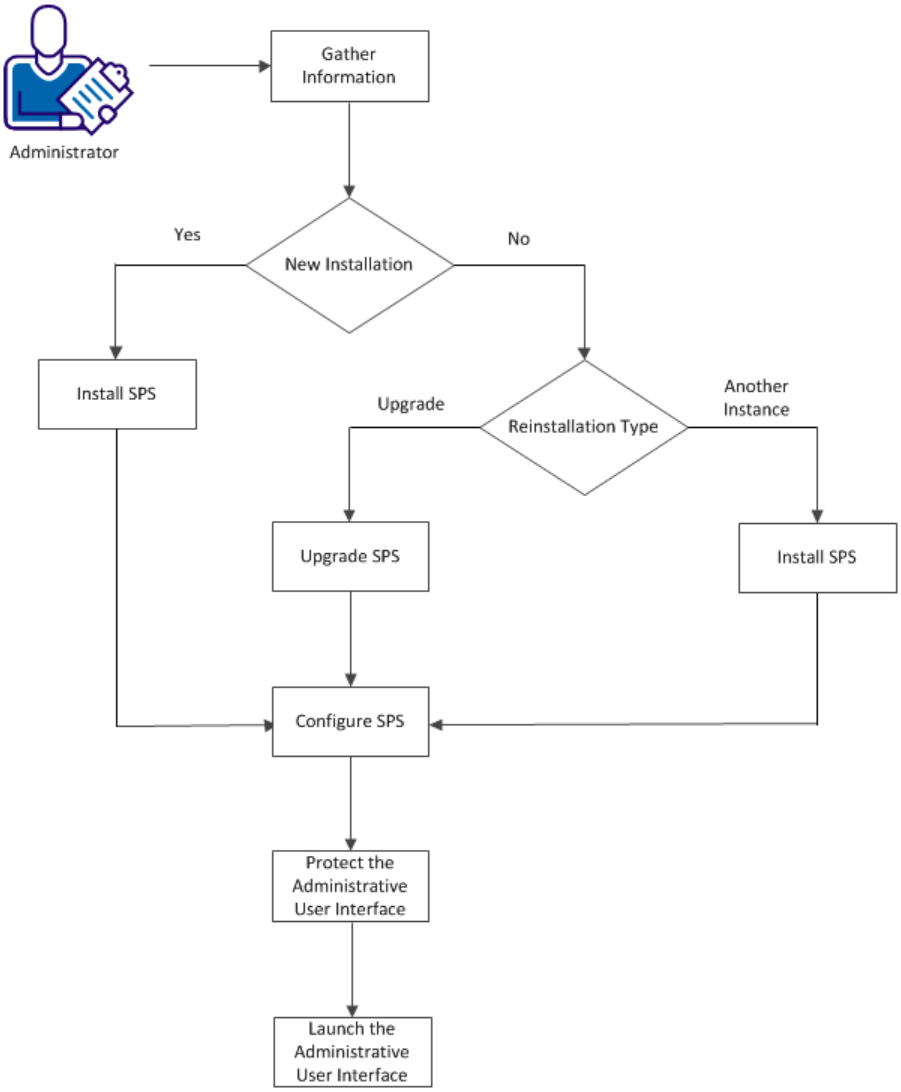
Install, Upgrade, and Configure SPS (see page 27)
How to Set Log Files, and Command-line Help to Another Language (see page 42)

## Install, Upgrade, and Configure SPS

The CA SiteMinder for Secure Proxy Server is a stand-alone server that provides a proxy-based solution for access control. SPS employs a proxy engine that provides a network gateway for the enterprise and supports multiple session schemes that do not rely on traditional cookie-based technology.

The following diagram describes how you can install and configure SPS:

**Install, Upgrade, and Configure SPS**



**More information:**

Installation Worksheet (see page 29)
Install SPS (see page 31)
Install Multiple Instances of SPS (see page 32)
Upgrade SPS (see page 33)
Configure SPS (see page 34)
Protect the Administrative User Interface (see page 37)
Launch the Administrative User Interface (see page 38)

## Prerequisites

Before you install SPS, verify the following prerequisites:

- If you are installing SPS on an RHEL 5 or RHEL 6 64-bit computer, verify that you have installed the following libraries on the computer:
  - libstdc++.so.6
  - ibexpat.so.0
  - libuuid.so.01

  **Note**: These libraries must be 32-bit binaries rather than 64-bit binaries.
- If you are installing SPS on an RHEL 5.5 computer, verify that you have installed the Legacy Software Development package on the computer.

## Installation Worksheet

SPS configuration wizard displays a series of prompts for registering a trusted host. A trusted host is a client computer where one or more SiteMinder Web Agents can be installed. To establish a connection between the trusted host and the Policy Server, register the host with the Policy Server. After registration is complete, the registration tool creates the SmHost.conf file. When this file is created, the client computer becomes a trusted host.

Before you install, upgrade, or configure SPS, verify that you gathered the following information required for host registration, embedded Apache web server and Tomcat server:

| Information Requested | Description/Default |
|---|---|
| SiteMinder administrator name | Name of the administrator that matches the name already defined at the Policy Server.This administrator must have the privileges to create a trusted host. |

| Information Requested | Description/Default |
|---|---|
| SiteMinder administrator password | Password of the SiteMinder administrator who has privileges to register a trusted host. Must match the password used at the Policy Server. |
| Trusted host name | Name of the trusted host assigned during the installation. |
| Host Configuration Object | Name of a host configuration object already defined in the Policy Server administrative UI. |
| Agent Configuration Object | Name of an existing Agent Configuration Object defined in the Policy Server administrative UI |
| IP address of the Policy Server where the host is registered | **Note**: Include a port number when SiteMinder is behind a firewall. For example, 111.12.12.2:12 |
| Host Configuration File name and location | Identifies the SmHost.conf file, which Web Agents and custom Agents use to act on behalf of the trusted host. The wizard lists the default location. |
| Name and location of the Web Agent configuration file | The wizard lists the default location. |
| Email address of the Apache web server administrator | The email address for the administrator<br>Default: admin@company.com |
| Fully qualified host name of the server | A fully qualified name in the following format: *computer_name.company.com* |
| Port number for Apache HTTP requests | The port listening for HTTP requests from Apache Default: 80 |
| Port number for Apache SSL requests | The port listening for SSL requests from Apache<br>Default: 443 |
| Port number for Tomcat HTTP requests | The port listening for HTTP requests from Tomcat<br>Default: 8080 |
| Port number for Tomcat SSL requests | The port listening for SSL requests from Tomcat<br>Default: 543 |
| Port number for Tomcat shutdown requests | The port listening for shutdown requests from Tomcat<br>Default: 8005 |

| Information Requested | Description/Default |
| --- | --- |
| Port number of AJP | The port number of AJP<br>Default: 8009 |

# Install SPS

Before you install SPS, verify that you have gathered the information required to install SPS.

## Install SPS on Windows

**Follow these steps:**

1. Copy the installation program from the download location on the CA Support site.
2. Right-click the executable and select Run as administrator.
3. Double-click ca-proxy-<*version*>-<*operating_system*>.exe.

   The installation program starts.
4. Follow the instructions from the installation wizard.

   **Note**: By default, SPS sets the instance name of the first installation as **default**. You cannot modify the default value and you cannot use the name for any other SPS instance.
5. Restart your system after the installation completes.

### Install SPS on Linux or Solaris

SPS supports installations on Linux and Solaris.

On Linux, the folder where you install SPS must have sufficient permissions (755). Do not install SPS under the /root folder, because its default permissions (750) are insufficient.

On Solaris, SPS runs as the "nobody" user. If you prefer not to run SPS as this user, create an alternate user and assign the necessary permissions.

**Follow these steps:**

1. Copy one of the following programs from the download location on the CA Support site to a temporary directory:

   `Solaris: ca-proxy-12.5-sol.bin`

   `Linux: ca-proxy-12.5-rhel30.bin`

2. Enter *one* of the following commands:

   ```
   sh ./ca-proxy-12.5-sol.bin
   ```

   ```
   sh ./ca-proxy-12.5-rhel30.bin
   ```

3. Follow the screen prompts provided by the installation wizard.

### Verify SPS Installation

You can check the InstallLog file to verify that SPS installation is successful. By default, the InstallLog is installed in the following location on all platforms:

*sps_home*`\install_config_info\CA_SiteMinder_Secure_Proxy_Server_InstallLog.log`

## Install Multiple Instances of SPS

You can install multiple SPS instances on the same computer. Each SPS instance uses a unique instance name and ports for communication, and creates a separate directory structure.

**Follow these steps:**

1. Double-click ca-proxy-<*version*>-<*operating_system*>.exe.

   The installation program starts.

2. Select the option to install a new instance.

3. Follow the instructions from the installation wizard.

   **Note**: Verify that you enter unique values for the instance name and the different ports that are used for communication.

## Upgrade SPS

You can run the installation program to upgrade from a previous version of SPS to the current version.

**Note**: If you configured filters or customized session schemes, take a back up of the lib directory from the Tomcat/Server/ path before you upgrade.

**Follow these steps:**

1. Double-click ca-proxy-<*version*>-<*operating_system*>.exe.

   The installation program starts.

2. Select OK to upgrade SPS version.
3. Follow the instructions from the installation wizard.
4. Restart your system after the installation completes.

## Configure SPS

After you install SPS, run the configuration wizard. The configuration wizard lets you register the trusted host for the embedded SiteMinder Web Agent and performs some administrative tasks for the embedded Apache web server.

**Important!** Before you run the wizard, verify that you have set up the required objects at the Policy Server where you want to register the host. If these objects are not configured, trusted host registration fails.

**Follow these steps:**

1. Open a console window and navigate to the directory sps_home/secure-proxy.
2. Enter *one* of the following commands:

   `Windows: ca-sps-config.exe`

   `UNIX: ca-sps-config.sh`

   The configuration wizard starts.
3. Select the version of the Policy Server with which you want to configure SPS.
4. Select the option to perform host registration immediately.
5. (Optional) Select the option to enable shared secret rollover.
6. Perform the following steps to register the trusted host registration:
   a. Specify the name and password of the SiteMinder administrator.

      **Note**: The information you enter must already be defined at the Policy Server where the trusted host is registered.

   b. Specify the name of the Trusted Host and the Host Configuration Object.

      **Note**: The name you enter for the trusted host must be unique. The name for the Host Configuration Object must already be defined at the Policy Server where the trusted host is registered.

   c. Enter the IP address of the Policy Server where you want to register the trusted host.

   d. Select a FIPS mode.

   e. Specify the name and location of the host configuration file, SmHost.conf. The wizard lists the default location.

   f. Specify the name of the Agent Configuration Object.

      **Note**: The Agent Configuration Object that you enter must already be defined at the Policy Server where the trusted host is registered.

7. Enter the following information for the Apache web server:
   - Server name
   - Web server administrator email address
   - HTTP port number
   - SSL port number
8. Enter the following information for the Tomcat server:
   - HTTP port number
   - SSL port
   - Shutdown port number
   - AJP port number

   **Note**: Users installing on systems running Solaris or Linux see an additional screen that prompts for the name of the user under which Tomcat and Apache runs. This user cannot be root. Create the user account manually; the installation program does not create it for you. The Tomcat user must have all privileges (rwa) for the log directories.
9. Select Yes if you want to enable the Web Agent.
10. Select Yes if you want SPS to act as a Federation Gateway.
11. Review the Configuration Summary
12. Click Install.

    SPS is configured and the configuration files are installed.
13. Click Done to exit the wizard.
14. Start the SiteMinder Secure Proxy and SiteMinder proxy engine services.

**Note**: If you run the Configuration Wizard again, SSL must be reinitialized.

## Additional Configuration on the SPS

After installing SPS and running the configuration wizard, you can modify SPS configuration to suit your environment. The following configuration files contain settings that affect SPS:

**httpd.conf**

Contains the settings for the Apache web server.

**server.conf**

Contains the settings that determine SPS behavior, including virtual hosts, and session scheme mapping.

**logger.properties**

Contains the settings that determine SPS logging behavior.

**proxyrules.xml**

Contains the rules that determine how SPS handles incoming requests.

**More information:**

Configuring the Apache Web Server (see page 81)
Configuring Proxy Rules (see page 141)

# Additional Tasks for Upgrades

At the end of the installation process, you can perform some additional steps to support the upgrade. Depending on the amount of customization in your SPS deployment, you can perform one or more of the following tasks:

- Verify that the SSL configuration paths inside the ssl.conf file and the server.conf file are correct for your environment. The automated portion of the upgrade assumes that all certificates are in the default location.
- If you enabled SSL and upgraded the SPS from a release earlier to SPS 12.5 on Linux, execute the following command perform to start Apache in the SSL mode:

  ```
  <install-path>/secure-proxy/proxy-engine/sps-ctl startssl
  ```

- Verify that all certificates, Certificate Authorities, and keys have been correctly copied to their folders in the *sps_home*\secure-proxy\SSL.
- Modify the path to the proxy rules DTD file in the proxyrules.xml file. The default path of the DTD file is *sps_home*\proxy-engine\conf\dtd\proxyrules.dtd.

### Customize JVM Parameters

You can customize Java Virtual Machine (JVM) parameters in the following files:

- On Windows, modify the SmSpsProxyEngine.properties file located in the directory *sps_home*\proxy-engine\conf.
- On UNIX, modify the proxyserver.sh file located in the directory *sps_home*/proxy-engine.

## Protect the Administrative User Interface

Before you access the Administrative User Interface, you must protect the Administrative User Interface using the WAMUI.

**Follow these steps:**

**Note**: For specifying an Agent, use the Agent you created for configuring SPS.

1. Create a user directory.
2. Create a domain.
3. Create an authentication scheme with the following details:
   a. Select HTML Form Template from Authentication Scheme Type.
   b. Select Use Relative Target.
   c. Type the following value in Target:

      `/context_path/forms/login.fcc;ACS=0;REL=1`

      **Note**: To get the context path, open the proxyui.xml file from *sps_home*/Tomcat/conf and note down the value mentioned in the path parameter.
   d. Type **smauthhtml** as Library.

4. Create a realm with the following details:
   a. Select the domain you created in Step 2.
   b. Select the Agent you created for SPS.
   c. Type the following value in Resource Filter:

      `/context_path`

   d. Select Protected as Default Resource Protection.
   e. Select the authentication scheme you created in Step 3 as Authentication Scheme.
5. Create a rule with the following details:
   a. Select the Domain and Realm you created in Step .
   b. Type **/*** as Resource.
   c. Ensure that the Effective Resource is <sps_agent>/proxyui*.
   d. Select Allow Access.
   e. Select the Get and Post actions.
6. Create a policy with the following details:
   a. Select the domain you created in Step 2.
   b. Select the SPSAdmin users.
   c. Select the rule you created in Step 5.
7. Open the Agent you created for SPS and perform the following steps:
   a. Add .css to the value list of the IgnoreExt ACO.
   b. Uncomment the LogoffUri ACO and set the value to /proxyui/logout.

## Launch the Administrative User Interface

You can launch the Administrative User Interface after you start the proxy engine services. To launch the URL, enter the following URL in a web browser:
`http://fullyqualifiedhostname:port/proxyui/`

SPS is installed or upgraded, and is configured.

If you want to perform a silent installation and configuration after the first installation, see Silent Installation and Configuration. If you want to uninstall SPS, see Uninstall SPS. If you want to start SPS in various modes, see Start SPS in Single-Process or Multiple Process Mode. If you want to modify the default location of the SiteMinder forms, see Modify the Default Location of the SiteMinder Forms.

## Silent Installation and Configuration

After you have installed and configured SPS for the first-time, you can reinstall it unattended at a later time or install another instance of SPS unattended using saved configuration data.

After installation, SPS creates a sample properties file in the sps-home/install_config_info folder. After configuration, the same properties file is updated with additional properties for configuration. This properties file is used for subsequent silent installation and configuration with customized values.

**Follow these steps:**

1. Open a command window.
2. Navigate to the folder where you installed the properties file. The default is sps-home/install_config_info.
3. Open the command prompt.
4. Perform one or both the following steps:
   a. Execute the following command to perform a silent installation:

      ```
      ca-proxy-12.5-operating_system -i silent -f ca-sps-installer.properties
      ```

      **operating_system**

      Defines the operating system, either win32, sol, or rhel30.

   b. Execute the following command to perform a silent configuration:

      ```
      ca-sps-config -i silent -f ca-sps-installer.properties
      ```

   The installation or configuration proceeds without further user interaction.

## Uninstall SPS

### Uninstall SPS from Windows

**Follow these steps:**

1. Open the command prompt and navigate to the root installation directory.
2. Execute the following command for each SPS instance you want to uninstall:

   ```
   ca-sps-uninstall.cmd
   ```

   SPS is uninstalled from your system.

**Note**: If you have modified any of SPS files such as server.conf, the uninstall program does not remove these files or their parent folders automatically.

## Uninstall SPS from UNIX

**Follow these steps:**

1. Open a console window.
2. Navigate to the root installation directory.
3. Run the following program at the command prompt:

   ```
   ./ca-sps-uninstall.sh
   ```

**Note**: If you have modified any SPS files, such as server.conf, the uninstall program does not remove these files or their parent folders automatically. Delete any files and folders for files you have changed.

# Start SPS in Single or Multi Process Mode

SPS supports single-process and multiple-process modes, which enable the embedded Web Agent to create the Low Level Agent Worker Process (LLAWP) at runtime.

SPS can be configured to start in a single-process or multiple-process mode. Single-process mode is the default.

The modes operate as follows:

**Single-Process Mode**

This mode causes the Web Agent to use local resources rather than shared operating system resources offered by the LLAWP to operate. No separate LLAWP processes are started in single-process mode. When multiple virtual hosts run, single-process mode results in an increase in the memory footprint of the SPS Java process.

**Note:** Single-process mode is supported only for host servers that run as a single server process.

**Multiple-Process Mode**

This mode causes the LLAWP framework to spawn a process for every virtual host. Because multiple-process mode uses shared memory, SPS uses shared operating system resources for logging, caching, and monitoring of multiple web server processes.

**To set the mode of operation**

1. Open the server.conf file in a text editor.
2. Set the singleprocessmode parameter as follows:
   - To use single process mode, keep singleprocessmode set to yes.
   - To use multiple-process mode, change singleprocessmode to no.
3. If you modify the server.conf file, restart SPS.

The SPS mode of operation is set.

## Modify the Default Location of the SiteMinder Forms

Beginning with SPS v6.0, the default location of the SiteMinder forms is no longer /siteminderagent/forms. To continue to use this location to serve forms, modify the SPS forms location.

**Follow these steps:**

1. Create the siteminderagent directory in the following location:

   *sps_home*/proxy-engine/examples/siteminderagent
2. Copy the forms folder from the following directory

   *sps_home*/proxy-engine/examples

   to the following directory:

   *sps_home*/proxy-engine/examples/siteminderagent

   The forms are copied to sps_home/proxy-engine/examples/siteminderagent/forms.

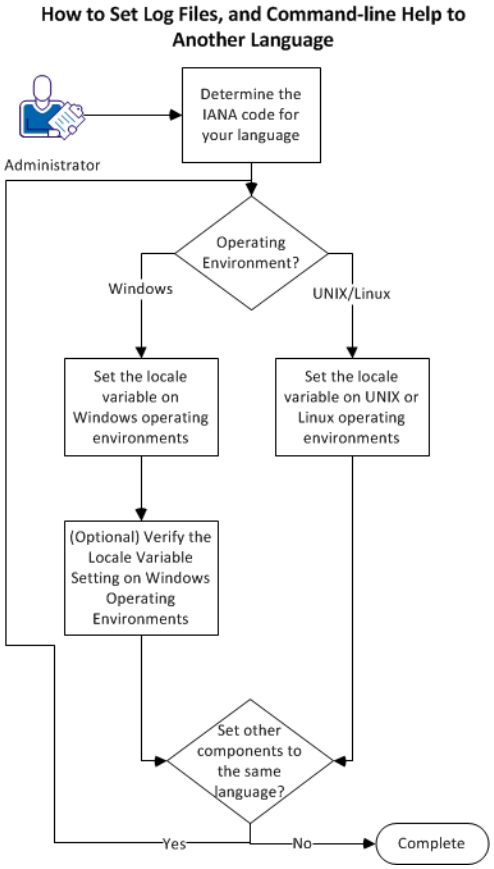
   **Note**: If you customize the location of the forms folder, ensure that you update the httpd.conf file with the location of the forms images.

# How to Set Log Files, and Command-line Help to Another Language

The following components support log files, and command-line help in other languages:

- The Policy Server
- The Web Agent
- The Report Server
- The CA SiteMinder Agent for SharePoint
- The CA SiteMinder SPS
- [set AGENT value for your book]s
- Any custom software that is created with the CA SiteMinder SDK.

The following graphic describes the work flow for setting log files, and command-line help to another language:



**Follow these steps:**

1. Determine the IANA code for your language (see page 43).
2. Create the environment variable for your operating environment using one of the following procedures:
   - Set the locale variable on Windows operating environments (see page 45).
   - Set the locale variable on UNIX or Linux operating environments (see page 47).
3. (Optional) Verify the locale variable setting on windows operating environments (see page 46).
4. (Optional) Repeat Steps 1 through 3 to set any other components in your environment to the same language.

## Determine the IANA Code for Your Language

Each language has a unique code. The Internet Assigned Numbers Authority (IANA) assigns these language codes. Adding a language code to a locale variable changes the language that the software displays. Determine the proper code for the language that you want before creating the locale variable.

The following table lists the IANA codes that correspond to the languages supported by the software:

| Language | IANA Code |
|---|---|
| Brazilian Portuguese | pt_BR |
| French | fr |
| German | de |
| Italian | it |
| Japanese | ja |
| Korean | ko |
| Simplified Chinese | zh-Hans |
| Spanish | es |

**Note**: A list of IANA language codes is available from this third-party website.

## Environment Variables

The environment variables are settings by which users can customize a computer to suit their needs. Examples of environment variables include the following items:

- A default directory for searching or storing downloaded files.
- A username.
- A list of locations to search for executable files (path).

Windows operating environments allow global environment variables, which apply to all users of a computer. The environment variables on UNIX or Linux operating environments must be set for each user or program.

To set the locale variable, pick the procedure for your operating environment from the following list:

- Set the locale variable on Windows operating environments (see page 45).
- Set the locale variable on UNIX or Linux operating environments (see page 47).

## Set the Locale Variable on Windows Operating Environments

The following locale variable specifies the language settings for the software:

```
SM_ADMIN_LOCALE
```

Create this variable and set it to the language that you want. Set this variable on *each* component for which you want to use another language. For example, suppose you want to have a Policy Server and an agent that is set to French. Set this variable on both of those components to French.

**Note**: The installation or configuration programs do *not* set this variable.

**Follow these steps:**

1. Click Start, Control Panel, System, Advanced system settings.

   The system properties dialog appears.
2. Click the Advanced tab.
3. Click Environment Variables.
4. Locate the System variables section, and then click New.

   The New System Variable dialog opens with the cursor in the Variable name: field.
5. Type the following text:

   ```
   SM_ADMIN_LOCALE
   ```
6. Click the Variable name: field, and then type the IANA language code (see page 43) that you want.
7. Click OK.

   The New System Variable dialog closes and the SM_ADMIN_LOCALE variable appears in the list.
8. Click OK *twice*.

   The locale variable is set.
9. (Optional) Repeat Steps 1 through 8 to set other components to the same language.

## Verify the Locale Variable Value on Windows Operating Environments

You can very the value to which the locale variable is set at any time. You can do this procedure after setting the variable to confirm that it is set correctly.

**Note**: Instructions for verifying the variable value on UNIX and Linux are in the setting procedure (see page 47).

**Follow these steps:**

1. Open a command-line window with the following steps:
   a. Click Start, Run.
   b. Type the following command:

      ```
      cmd
      ```

   c. Click OK.

      A command-line window opens.
2. Enter the following command:

   ```
   echo %SM_ADMIN_LOCALE%
   ```

   The locale appears on the next line. For example, when the language is set to German, the following code appears:

   ```
   de
   ```

   The value of the locale variable is verified.

## Set the Locale Variable on UNIX or Linux Operating Environments

The following locale variable specifies the language settings for the software:

`SM_ADMIN_LOCALE`

Create this variable and set it to the language that you want. Set this variable on *each* component for which you want to use another language. For example, suppose you want to have a Policy Server and an agent that is set to French. Set this variable on both of those components to French.

**Note**: The installation or configuration programs do *not* set this variable.

**Follow these steps:**

1. Log in to the computer that is running the component that you want.
2. Open a console (command-line) window.
3. Enter the following command:

   `export SM_ADMIN_LOCALE=IANA_language_code`

   The command in the following example sets the language to French:

   `export SM_ADMIN_LOCALE=fr`

   The locale variable is set.
4. (Optional) Verify that the locale variable is set properly by entering the following command:

   `echo $SM_ADMIN_LOCALE`

   The locale appears on the next line. For example, when the language is set to German, the following code appears:

   `de`
5. (Optional) Repeat Steps 1 through 4 to set other components to the same language.

# Chapter 3: FIPS-140 Support

This section contains the following topics:

FIPS Support Overview (see page 49)
Configuration Process for FIPS ONLY Mode (see page 50)
Migration to FIPS MIGRATE Mode (see page 50)
Migration to FIPS ONLY Mode (see page 51)

## FIPS Support Overview

The Secure Proxy Server supports the requirements for cryptographic modules specified in the FIPS 140-2 standard. When you install SPS, a dialog appears that prompts you to select the level of FIPS support your operating configuration requires.

During a new installation you can select one of these three FIPS modes:

- COMPAT — Specifies that the installation is not FIPS-compliant. Select this mode when interacting with clients running earlier versions of SPS.
- MIGRATE — Specifies that SPS operates both with FIPS-compliant algorithms and algorithms used in earlier version of the SPSSPS simultaneously while the data is migrated.
- ONLY — Specifies that only FIPS-compliant algorithms are used and accepted by SPS. When you install in this mode, additional manual configuration is required.

The FIPS mode you select during installation usually is the same as the FIPS mode configured on the Policy Server.When the Policy Server is in Migrate mode, it can operate with SPS in any mode.

If you are upgrading an existing SPS installation, SPS continues to work as before, that is, in COMPAT mode. You can change the mode manually using the smreghost command, as described in subsequent sections. Be sure to restart the system after a mode change so that the Web Agent, SPS server, and the Apache server pick up the changes.

**More information:**

Migration to FIPS MIGRATE Mode (see page 50)
Configuration Process for FIPS ONLY Mode (see page 50)

# Configuration Process for FIPS ONLY Mode

After you install SPS in FIPS ONLY mode, the following additional configuration steps are required:

- Verify that SPS is running in full SSL mode.
- Verify that the server key used to configure SPS in SSL mode was generated using a FIPS-compliant cryptographic algorithm.
- Follow the procedure for configuring SSL in FIPS ONLY mode.

# Migration to FIPS MIGRATE Mode

If you are upgrading from an earlier version and want to use FIPS-compliant algorithms, you can change the Web Agent inside SPS from COMPAT mode to MIGRATE mode.

**To set SPS to FIPS MIGRATE mode**

1. Stop the SPS services.
2. Open a command-line window.
3. Enter the following command:

   ```
   smreghost -i policy_server_ip_address -u administrator_user_name -p
   administrator_password -hn hostname_for_registration -hc host_config_object -f
   path_to_host_config_file -o -cf MIGRATE
   ```

   Example:

   ```
   smreghost -i localhost -u siteminder –p firewall -hn helloworld -hc host  -f
   "C:\Program Files\CA\secure-proxy\proxy-engine\conf\defaultagent\SmHost.conf"
   -o –cf  MIGRATE
   ```

4. Restart the machine.(Windows only)
5. Restart the SPS services.

The Web Agent inside SPS is changed from FIPS COMPAT to FIPS MIGRATE mode.

# Migration to FIPS ONLY Mode

If the SiteMinder Policy Server is in FIPS ONLY or FIPS COMPAT mode, you can change the FIPS mode of SPS from FIPS COMPAT to FIPS ONLY after you upgrade.

**Follow these steps:**

1. Stop SPS services.
2. Set the value of the OPENSSL_FIPS environment variable to 1.
3. Perform one of the following steps:
    1. If you are changing the FIPS mode on Windows, set the CA_SM_PS_FIPS140 environment variable to ONLY.
    2. If you are changing the FIPS mode on UNIX, perform the following steps:
        a. Open the proxyserver.sh file.

            **Default Path**: sps-home/proxy-engine/proxyserver.sh

        b. Set the value of the CA_SM_PS_FIPS140 environment variable to ONLY.
4. Execute the following command from the command prompt:

    ```
    smreghost -i policy_server_ip_address -u administrator_user_name -p
    administrator_password -hn hostname_for_registration -hc host_config_object -f
    path_to_host_config_file -o -cf ONLY
    ```

    Example:

    ```
    smreghost -i localhost -u siteminder –p firewall -hn helloworld -hc host  -f
    "C:\Program Files\CA\secure-proxy\proxy-engine\conf\defaultagent\SmHost.conf"
    -o –cf  ONLY
    ```
5. Determine whether SPS is running in full SSL mode. If SSL is already enabled on Apache inside SPS, SSL must be disabled and reconfigured for FIPS ONLY mode.
6. Open the httpd-ssl.conf file.

    **Default Path**: sps_home\httpd\conf\extra\httpd-ssl.conf
7. Set the value of the SSLPassPhraseDialog variable to custom.
8. Uncomment the following line:

    ```
    SSLCustomPropertiesFile "<sps_home>/Tomcat/properties/spsssl.properties"
    ```
9. Set the value of the SSLCustomPropertiesFile variable to <sps_home>\httpd\conf\spsapachessl.properties.
10. Set the value of the SSLSpsFipsMode variable to ONLY.
11. Restart the computer.
12. Start SPS services.

# Chapter 4: Using the SPS with Federation Security Services

This section contains the following topics:

Federation Security Services Introduction (see page 53)
SPS Use Cases in a SiteMinder Federated Environment (see page 53)
SPS Roles in a SiteMinder Federated Environment (see page 58)
Solutions for SPS Use Cases (see page 58)
Cookieless Federation (see page 67)
SPS as a Web Agent Replacement (see page 69)
SPS as a Federation Gateway (see page 71)

## Federation Security Services Introduction

SiteMinder Federation Security Services (FSS) allow the exchange of security information between business partners. The services provide seamless authentication and fine-grained authorization across enterprises.

Federation Security Services are implemented with the SPS in the following ways:

- As a replacement for a SiteMinder Web Agent.
- As a replacement for the SiteMinder Web Agent and the Web Agent Option Pack.

Federation services enable an organization and its partners to:

- Exchange user information in a secure manner
- Map user identities at one organization to user identities at other organizations
- Provide single sign-on across different organizations
- Control access to resources based on user information received from a partner
- Interoperate across heterogeneous environments, such as Windows, UNIX and various Web servers, such as IIS, Sun Java System and Apache

## SPS Use Cases in a SiteMinder Federated Environment

There are probably as many use cases for federated networks as there are business arrangements between partners. The use cases that follow demonstrate different ways of handling user identities to provide single sign-on between partners.

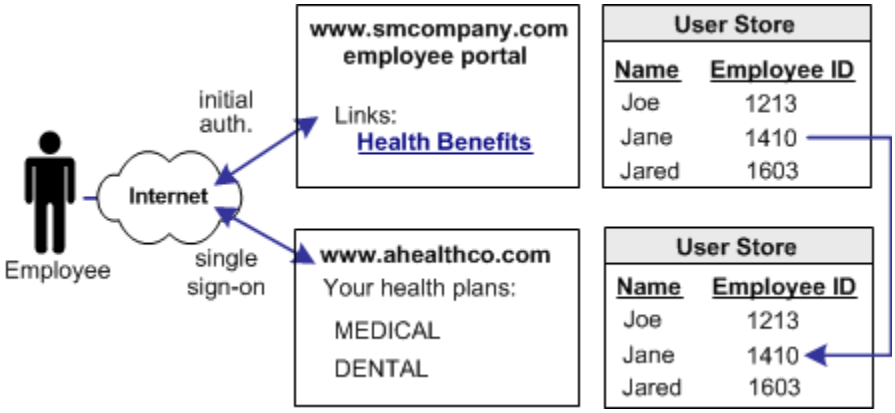
For more use cases, see the *CA SiteMinder Federation Security Services Guide*.

## Use Case 1: Single Sign-on Based on Account Linking

In Use Case 1, smcompany.com contracts with a partner company, ahealthco.com to manage employee health benefits.

An employee of smcompany.com authenticates at an employee portal at his company's site, www.smcompany.com and clicks a link to view her health benefits at ahealthco.com. The employee is taken to the ahealthco.com web site and is presented with her health benefit information without having to sign on to the website.

The following illustration shows this use case.




The company, ahealthco.com, maintains all health-related information for employees at smcompany.com. To do this, ahealthco.com maintains user identities for every employee of smcompany.com. When an employee of smcompany.com accesses ahealthco.com, an identifier for the employee is passed from smcompany.com to ahealthco.com in a secure manner. This identifier allows ahealthco.com to determine who the user is and the level of access to allow for that user.

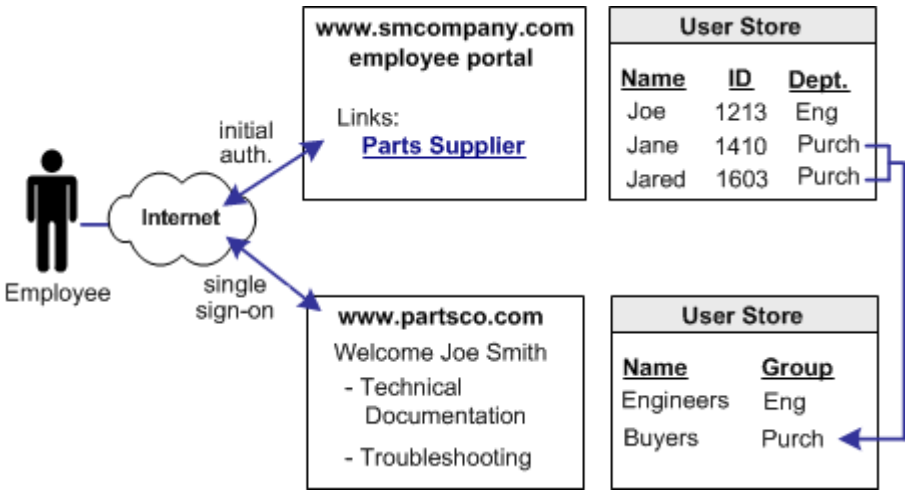
**More information:**

Solution 1: SSO Based on Account Linking (see page 58)

## Use Case 2: Single Sign-on Based on User Attribute Profiles

In Use Case 2, smcompany.com buys parts from a partner named partsco.com.

An engineer authenticates at the employee portal, smcompany.com and clicks a link to access information at partsco.com. Being an engineer at smcompany.com, the user is taken directly to the Specifications and Parts List portion of partsco.com website without having to log in.




When a buyer for smcompany.com authenticates and clicks a link for partsco.com, the buyer is taken directly to the ordering area of the partsco.com website. The buyer does not have to log in.

Additional attributes, such as the user name are passed from smcompany.com to partsco.com to personalize the interface for the individual user.

Partsco.com does not want to maintain user identities for all employees at smcompany.com, bu the company wants to control access to sensitive portions of the website. To control the access, partsco.com maintains a limited number of profile identities for users at smcompany.com. One profile identity is maintained for engineers and one profile identity is maintained for buyers.

When an employee of smcompany.com accesses partsco.com, smcompany.com sends user attributes in a secure manner to partsco.com. Partsco.com uses the attributes to determine what profile identity controls access.

**More information:**

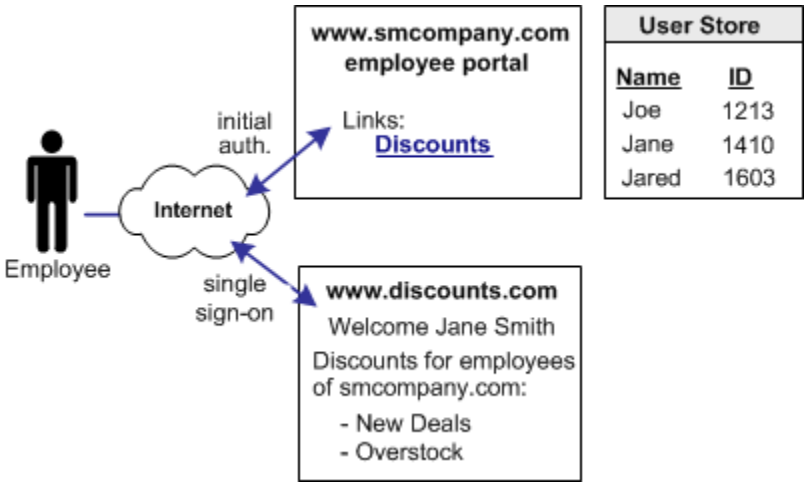
Solution 2: SSO Using User Attribute Profiles (see page 61)

## Use Case 3: Single Sign-on with No Local User Account

In Use Case 3, smcompany.com offers employee discounts by establishing a partnership with discounts.com.

An employee of smcompany.com authenticates at an employee portal at www.smcompany.com and clicks a link to access discounts at discounts.com. The employee is taken to the discounts.com website and presented with the discounts available for smcompany.com employees, without logging in to the discounts.com website.

The following illustration shows this use case.




Discounts.com does not maintain any identities for employees of smcompany.com. The company allows all employees of smcompany.com to access discounts.com as long as they have been authenticated at smcompany.com. When an employee of smcompany.com accesses discounts.com, authentication information is sent in a secure manner from smcompany.com to discounts.com. This information is used to allow access to discounts.com.

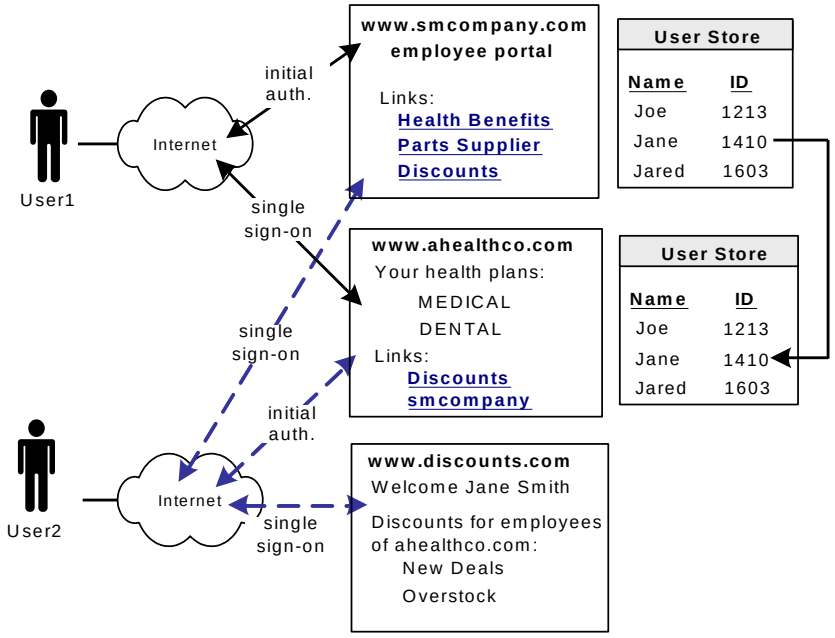
Additional attributes, such as the user name are passed from smcompany.com to discounts.com to personalize the interface for the individual user.

**More information:**

Solution 3: SSO with No Local User Account (see page 63)

## Use Case 4: Extended Networks

In Use Case 4, smcompany.com, ahealthco.com, and discounts.com all participate in an extended federated network. This case is an extension of the previous use cases.




In this network, not all customers of ahealthco.com work at smcompany.com. Ahealthco.com provides discounts only to its customers by establishing a relationship between themselves and discounts.com. Ahealthco.com maintains user identities for every customer so ahealthco.com manages local credentials, such as a password for each user. By managing local credentials, ahealthco.com can authenticate users and can provide single sign-on access to its partners.

In this extended network, the users access each website differently:

- User1 accesses health benefit information at the ahealthco.com website. User1 can access the partsco.com website by clicking the PartsSupplier link at smcompany.com, the employee portal. User1 can also click a link at the employee portal to access discounts at discounts.com.

User2 authenticates at the ahealthco.com website and clicks a link to access discounts at discounts.com, without logging in to the discounts.com website. The discounts the site presents to User2 reflect the business arrangement between ahealthco.com and discounts.com. Being employee of smcompany.com, User2 can also click a link at ahealthco.com and access the employee portal at smcompany.com without logging in to website.

- User3 (not shown in the example), is a customer of ahealthco.com, but is not an employee of smcompany.com. User3 authenticates at the ahealthco.com website and clicks a link to access discounts at discounts.com. User3 does not log in to the website. The discounts the site presents to User3 reflect the business arrangement between ahealthco.com and discounts.com. Because User3 is not an employee of smcompany.com, User3 cannot access the smcompany.com website.

**More information:**

Solution 4: SSO in an Extended Network (see page 65)

# SPS Roles in a SiteMinder Federated Environment

The SPS can provide solutions to federation use cases in one of two roles:

- As a standard proxy server that replaces the SiteMinder Web Agent
- As a federation gateway

The primary distinction between these two roles is the configuration and deployment effort required. The proxy server that replaces the Web Agent still requires that you set up a separate server and servlet engine to run the Federation Web Services application.

The proxy server acting as a federation gateway has the components of the Web Agent and the Federation Web Services application built-in. A dedicated server and servlet engine are not configured separately, which simplifies the federation setup.

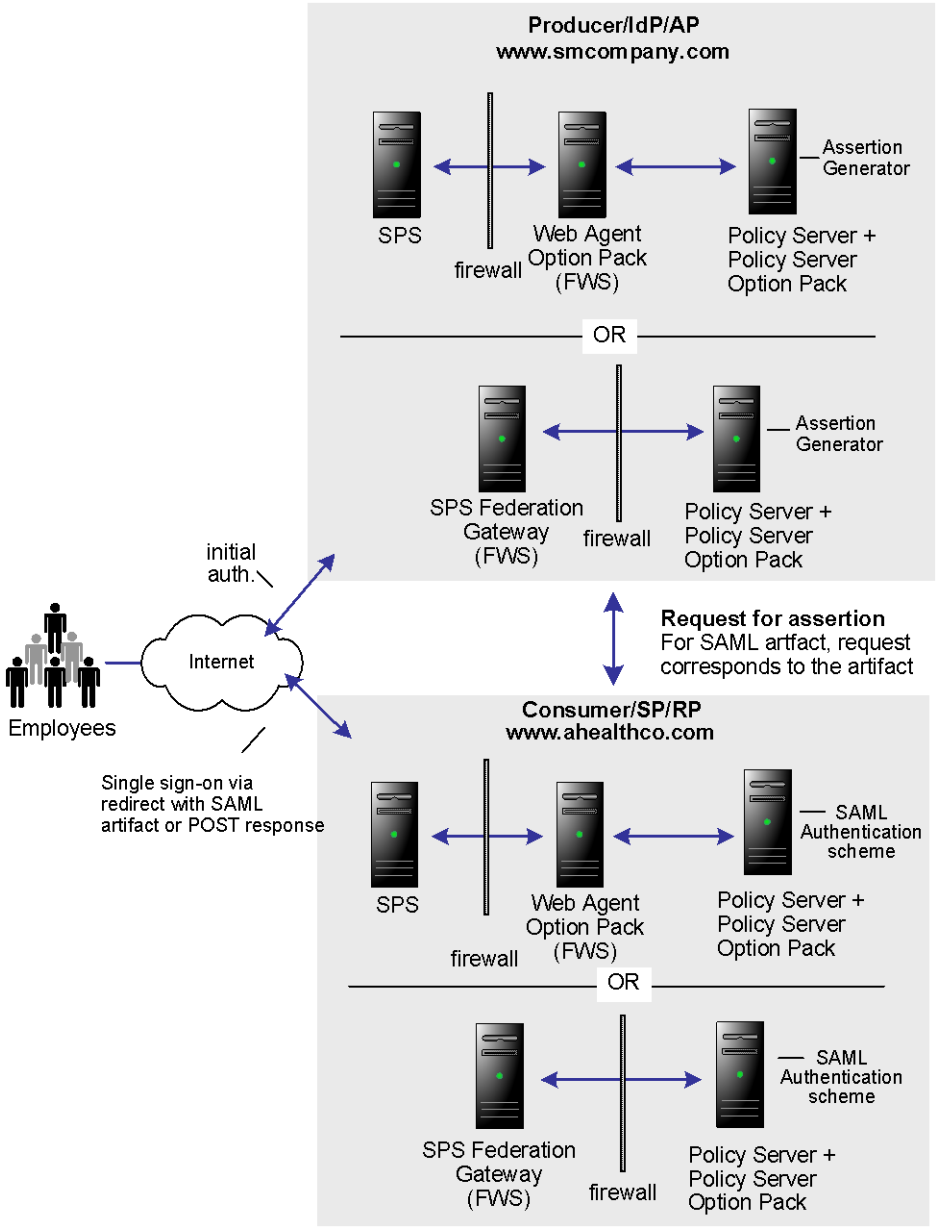
# Solutions for SPS Use Cases

The following sections show SPS solutions to the federation use cases.

## Solution 1: SSO Based on Account Linking

Solution 1 illustrates how Federation Security Services can be deployed at smcompany.com and ahealthco.com to solve Use Case 1 (see page 54): Single Sign-on Based on Account Linking.

The following figure shows the solution based on account linking.




SiteMinder v6.x is deployed at both sites and the installations are the same for both smcompany.com and ahealthco.com. The SPS with the Web Agent Option Pack or the SPS federation gateway can be installed on the Web server system and the Policy Server with the Policy Server Option Pack are installed on another machine.

The FWS application at the producing side provides the service that retrieves assertions. The FWS application at the consuming side provides the service that consumes assertions.

## Using SAML 1.x Artifact Authentication for Solution 1

The process that follows is one solution for single sign-on with account linking. This solution uses the SAML 1.x artifact profile. There are other solutions for this use case that involve other profiles (SAML 1.x POST and SAML 2.0 Artifact and POST). For these solutions, see the *CA SiteMinder Federation Security Services Guide*.

In this solution, smcompany.com is acting as the producer site. When an employee of smcompany.com accesses an employee portal at www.smcompany.com, the sequence of events is as follows:

1. The SPS provides the initial authentication.
2. When the employee clicks a link at smcompany.com to view her health benefits at ahealthco.com, the link makes a request to the Intersite Transfer Service at www.smcompany.com.
3. The Intersite Transfer Service calls the assertion generator, which creates a SAML assertion, inserts the assertion into the SiteMinder session server, and returns a SAML artifact.
4. The SPS redirects the user to www.ahealthco.com with the SAML artifact, in accordance with the SAML browser artifact protocol.

Ahealthco.com is acting as the consumer site. The redirect request with the SAML artifact is handled by the SAML credential collector Federation Web Services at ahealthco.com.

The sequence of events is as follows:

1. The SAML credential collector calls the SAML artifact authentication scheme to obtain the location of the assertion retrieval service at smcompany.com.
2. The SAML credential collector calls the assertion retrieval service at www.smcompany.com.
3. The assertion retrieval service at www.smcompany.com retrieves the assertion from the SiteMinder session server and returns it to the SAML credential collector at ahealthco.com.
4. The SAML credential collector then passes the assertion to the SAML artifact authentication scheme for validation and session creation and proceeds to issue a SiteMinder session cookie to the user's browser.
5. At this point the user is allowed access to resources at ahealthco.com based on policies defined at the Policy Server at ahealthco.com and enforced by the SPS at ahealthco.com.

In this example, the administrator at smcompany.com uses the Policy Server User Interface to configure an affiliate for ahealthco.com. The affiliate is configured with an attribute that is a unique ID for the user. This causes the assertion generator to include that attribute as part of the user profile in a SAML assertion created for ahealthco.com.

The administrator at ahealthco.com uses the Policy Server User Interface to configure a SAML artifact authentication scheme for smcompany.com. The authentication scheme specifies the location of the assertion retriever service at smcompany.com, how to extract the unique user ID from the SAML assertion, and how to search the user directory at ahealthco.com for the user record that matches the value extracted from the assertion.

## Solution 2: SSO Using User Attribute Profiles

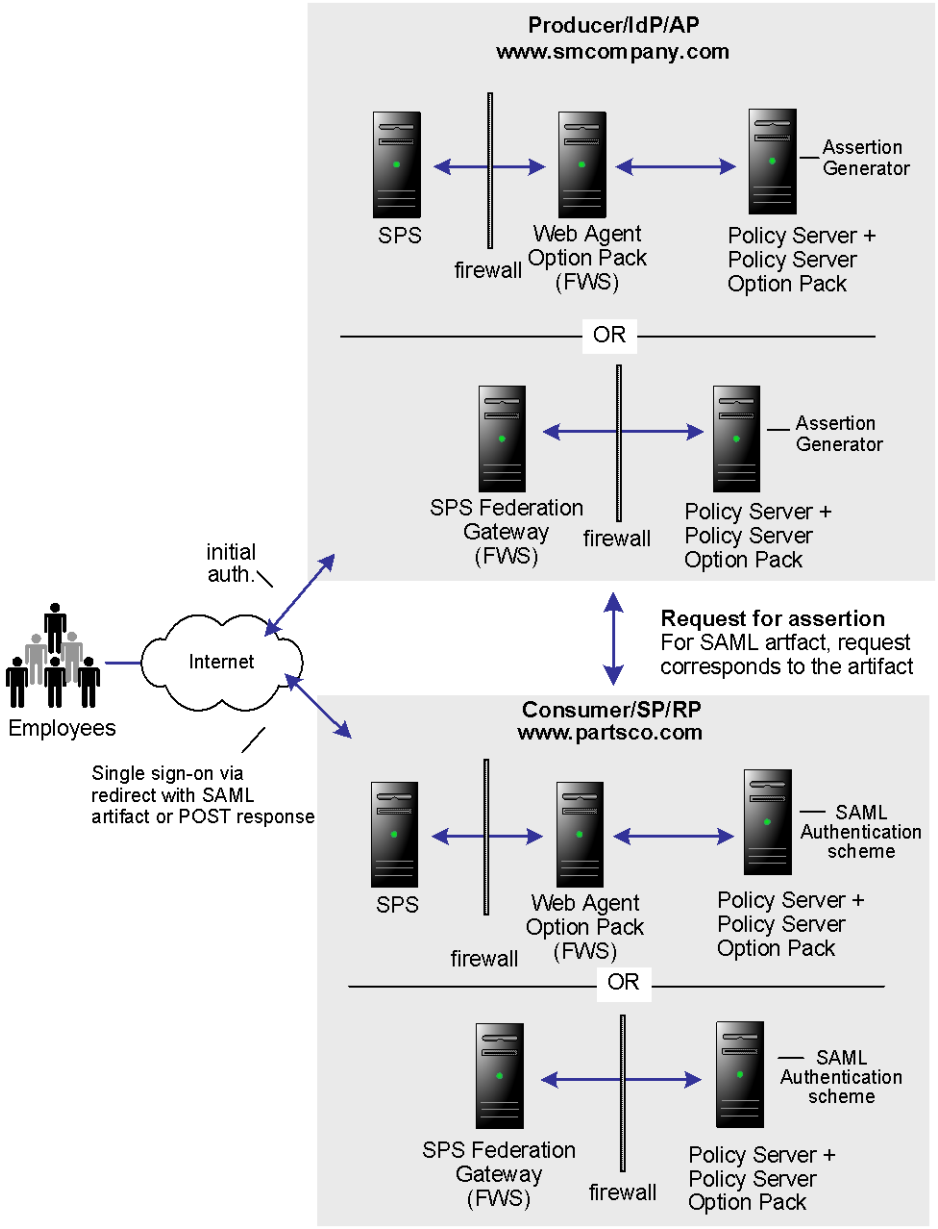
Solution 2 shows how SiteMinder Federation Security Services can be deployed at smcompany.com and partsco.com to solve Use Case 2 (see page 55): Single Sign-on Based on User Attribute Profiles.

SiteMinder v6.x is deployed at both sites. The interactions between the user and each site is similar, where partsco.com is acting as the consuming authority. The FWS application at the producing side provides the service that retrieves assertions. The FWS application at the consuming side provides the service that consumes assertions.

The following illustration is similar for SAML 1.x, SAML 2.0, and WS-Federation; however, the Federation Web Services components are different as follows:

- For SAML 1.x, the Assertion Retrieval Service (for artifact profile only) is at the Producer and the SAML credential collector is at the SP.
- For SAML 2.0, the Artifact Resolution Service (for artifact binding only) is at the IdP and the Assertion Consumer Service at the SP.
- For WS-Federation, the Single Sign-on Service is at the AP and the Security Token Consumer Service is at the RP.

  **Note:** WS-Federation only supports HTTP-POST binding.


Producer/IdP/AP
www.smcompany.com
SPS
firewall
Web Agent
Option Pack
(FWS)
Assertion
Generator
Policy Server +
Policy Server
Option Pack
OR
SPS Federation
Gateway
(FWS)
firewall
Assertion
Generator
Policy Server +
Policy Server
Option Pack
initial
auth.
Internet
Employees
Single sign-on via
redirect with SAML
artifact or POST response
Request for assertion
For SAML artfact, request
corresponds to the artifact
Consumer/SP/RP
www.partsco.com
SPS
firewall
Web Agent
Option Pack
(FWS)
SAML
Authentication
scheme
Policy Server +
Policy Server
Option Pack
OR
SPS Federation
Gateway
(FWS)
firewall
SAML
Authentication
scheme
Policy Server +
Policy Server
Option Pack

The configuration is similar to Solution 1: Single Sign-on based on Account Linking, except for the following:

- The administrator at smcompany.com defines the consumer/SP for partsco.com with an attribute specifying the user's department at the company. The assertion generator will include this attribute as part of the user profile in the SAML assertion created for partsco.com.
- The administrator at partsco.com defines an authentication scheme (artifact, post, or WS-federation) for smcompany.com. The scheme extracts the department attribute from the SAML assertion and searches the user directory at partsco.com for the user record that matches the department value taken from the assertion. The administrator defines one user profile record for each department that is allowed to access partsco.com's web site.

## Solution 3: SSO with No Local User Account

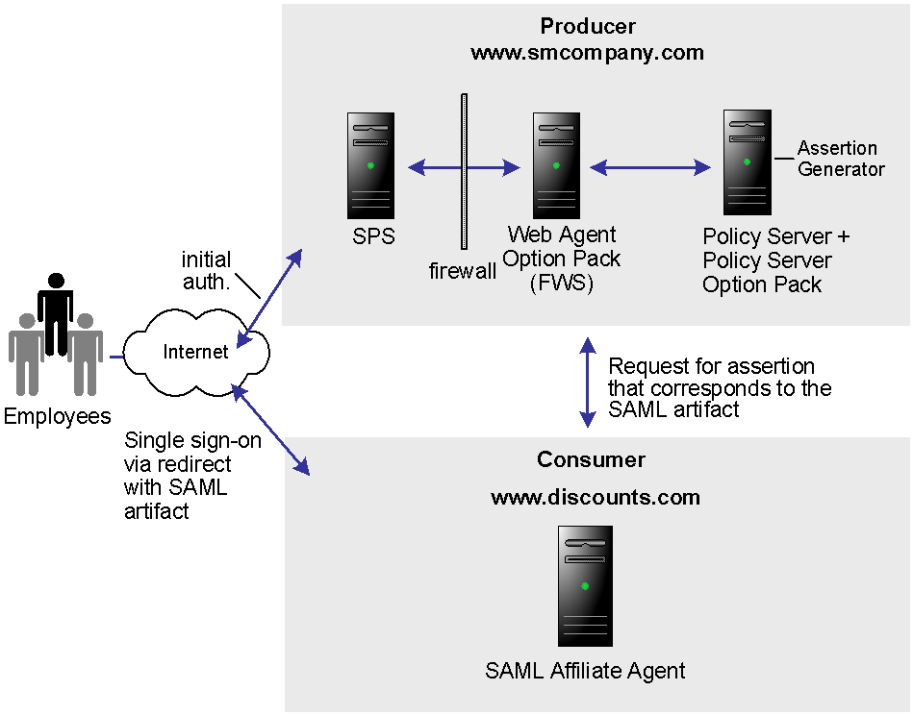
Solution 3 shows how SiteMinder Federation Security Services can be deployed at smcompany.com and discounts.com to solve Use Case 3 (see page 56): Single Sign-on with No Local User Account.

SiteMinder v6.x is deployed at smcompany.com by installing the SPS on one machine, the Web Agent Option Pack on another machine and installing the Policy Server with the Policy Server Option Pack on a third machine. The SAML Affiliate Agent is installed at discounts.com. It only supports SAML 1.0.

The FWS application at the producing side provides the assertion retrieval service. The FWS application at the consumer side provides the SAML credential collector.

**Note:** The SPS federation gateway does not support SAML 1.0 and therefore cannot act as a producer for the SAML Affiliate Agent.

The following figure shows single sign-on with no local user account.




Smcompany.com is acting as a SAML 1.x producer. When an employee of smcompany.com accesses an employee portal at www.smcompany.com, the following occurs:

1. The SPS provides the initial authentication.
2. When the employee clicks a link at www.smcompany.com to access deals at discounts.com, the link makes a request to the SPS at www.smcompany.com.
3. The SPS at www.smcompany.com calls the assertion generator, which creates a SAML assertion, inserts the assertion into the SiteMinder session server, and returns a SAML artifact.
4. The SPS redirects the user to www.discounts.com with the SAML artifact in accordance with the SAML browser artifact protocol.

Discounts.com is acting as the consumer site. The redirect request with the SAML artifact is handled by the SAML Affiliate Agent at www.discounts.com, as follows:

1. The SAML Affiliate Agent obtains the location of the assertion retrieval service at www.smcompany.com from a configuration file.
2. The SAML Affiliate Agent calls the assertion retrieval service at www.smcompany.com.

3. The assertion retrieval service at www.smcompany.com retrieves the assertion from the SiteMinder session server and returns it to the SAML affiliate agent at www.discounts.com.
4. The SAML Affiliate Agent then validates the SAML assertion and issues a SiteMinder affiliate session cookie to the user's browser.
5. The user is allowed access to resources at discounts.com.

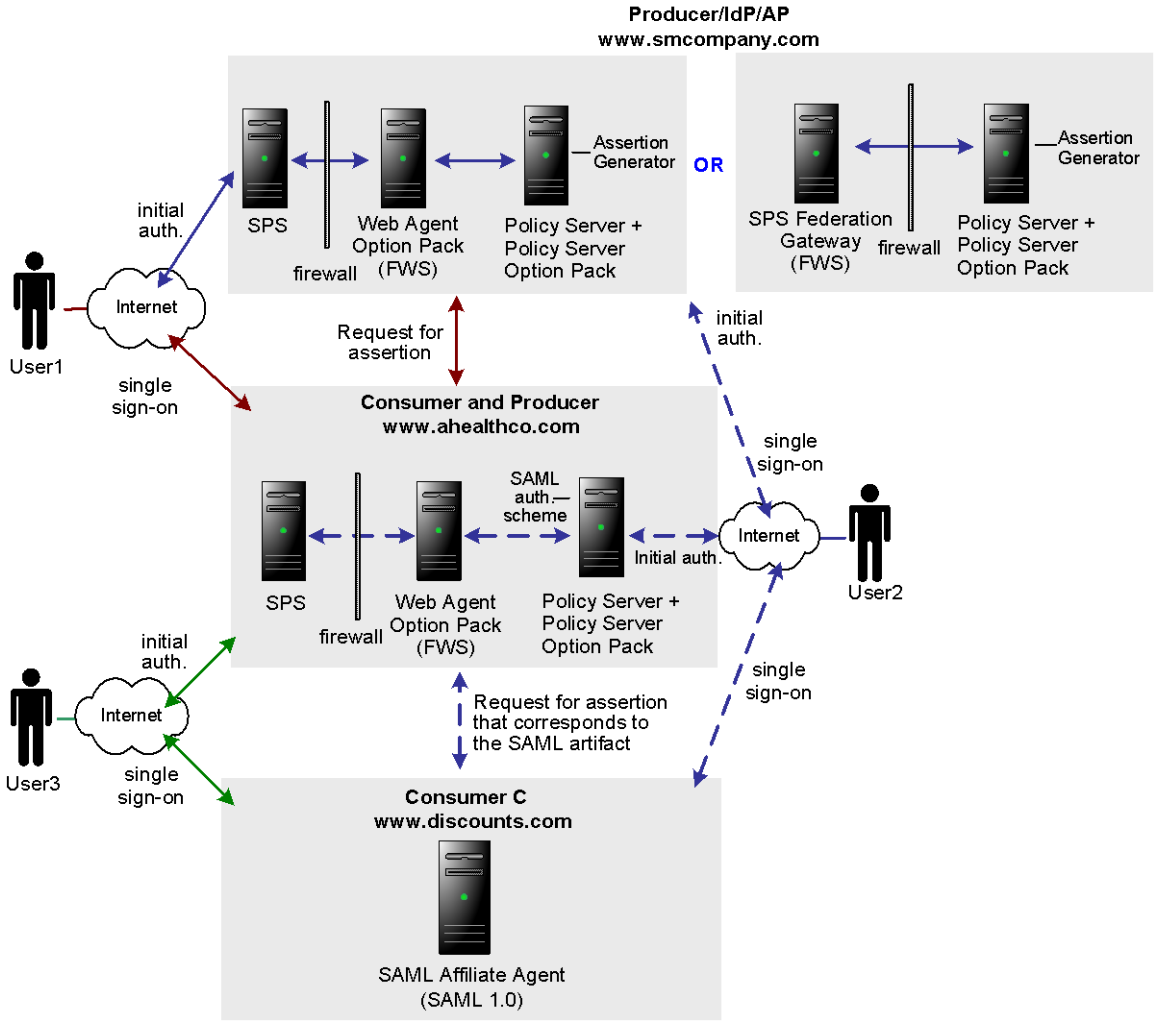
The administrator at smcompany.com uses the Policy Server User Interface to configure an affiliate for discounts.com. The affiliate is configured with attribute information to be passed to discounts.com. The assertion generator will include those attributes as part of the user profile in a SAML assertion created for discounts.com.

The administrator at discounts.com configures the SAML Affiliate Agent with information about the discounts.com site, the location of the assertion retriever service at smcompany.com, and what resources are to be protected by the affiliate defined at smcompany.com.

## Solution 4: SSO in an Extended Network

Solution 4 illustrates how SiteMinder Federation Security Services can be deployed at smcompany.com, ahealthco.com, and discounts.com to solve Use Case 4 (see page 57): Extended Networks.

The following illustration shows an extended network. SAML 1.x is the protocol being used.




SiteMinder is deployed at smcompany.com and ahealthco.com. At smcompany.com, the SPS with the Web Agent Option Pack can be installed across two machines or the SPS federation gateway can be installed on one machine. The Policy Server with the Policy Server Option Pack is installed on another machine. At ahealthco.com, the SPS with the Web Agent Option Pack can be installed across two machines and the Policy Server with the Policy Server Option Pack is installed on another machine. At discounts.com, the SAML Affiliate Agent is installed.

The FWS application at the producing side provides the service that retrieves assertions. The FWS application at the consuming side provides the service that consumes assertions.

In Solution 4:

- smcompany.com acts as a producer for User1 and a consumer for User2
- ahealthco.com acts as a consumer for User1 and a producer for User2 and a producer for User3
- discounts.com acts as a consumer for User1, User2, and User3

The administrator for smcompany.com has configured two entities in an affiliate domain, which represents ahealthco.com and discounts.com. These sites are configured in a similar manner as in Examples 1 and 3 described previously, but the configurations have been extended as follows:

- At smcompany.com, the administrator has configured a SAML authentication scheme (artifact or POST). For User2, the authentication scheme enables smcompany.com to act as a consumer for ahealthco.com.
- At ahealthco.com:
  - The administrator has configured an affiliate object that represents smcompany.com so an assertion is produced for User2. This makes single sign-on to smcompany.com possible.
  - The administrator has configured an affiliate object that represents discounts.com so an assertion is produced for User2 and User3. This makes single sign-on to discounts.com possible.
- At discounts.com, the administrator has configured the SAML Affiliate Agent to act as a consumer for smcompany.com, as in Example 3 (an arrow connecting the two sites is not shown in the illustration). The administrator at discounts.com has also added configuration information about ahealthco.com so that the SAML Affiliate Agent can consume assertions from ahealthco.com for User2 and User3.

# Cookieless Federation

Certain devices or environments cannot use cookies to establish user session and provide single sign-on.

One type of session scheme you can use in a federated environment is a cookieless scheme. The cookieless federation scheme is used to establish single sign-on. Verify that FWS-generated cookies (session and attribute) are not sent back to clients using mobile devices that do not support cookies.

### Cookieless Federation at the Producing Site

At the site producing assertions, the process for a cookieless transaction is as follows:

1. The SPS verifies if cookieless federation is enabled for the virtual host requesting the redirect.
2. The SPS verifies if the session scheme is a rewritable scheme, such as the simple_url scheme.
3. If the scheme is rewritable, SPS determines whether a session key has been created for the session and if this key is available to use.
4. SPS checks to see if the Location header in the HTTP response meets one of the following conditions:
   - It is being rewritten.
   - It is the same as the host of the request.
5. SPS rewrites the redirect response to include the session key information in the redirected URL.

### Cookieless Federation at the Consuming Site

At the site consuming assertions, if cookieless federation is enabled, the SPS replacing the Web Agent processes redirects using SAML authentication at the backend server.

In a cookieless federation, the SPS processes the request as follows:

1. The SPS receives a request from cookieless device, such as a mobile phone.
2. The SPS verifies if the cookieless federation is enabled for the virtual host requesting the redirect.
3. SPS then checks to see if the following conditions have been met:
   - The response from the backend server is a redirect.
   - The response contains an SMSESSION cookie.

   If these two conditions are met at the same time, it indicates that a SAML authentication has occurred at the backend server from the FWS application.
4. The SPS retrieves the session scheme being used.
5. The SPS creates an associated cookieless session and adds the session information to its session store.
6. If the session scheme is rewritable, such as a simple URL session scheme, the SPS rewrites the location header with the session key.

7. If the SPS determines that a cookieless federated session conversion has occurred, the SPS deletes the SMSESSION cookie from the response going to the browser.
8. The SPS then checks to see if attribute cookies should also be deleted. It does this by checking the deleteallcookiesforfed parameter (see page 121). If this parameter is set to yes, SPS deletes all the other cookies from the response going to the browser.

## Enable Cookieless Federation at the Consuming Side

When the SPS replaces the Web Agent at the side consuming assertions, the cookieless federation parameters are enabled for any cookieless session scheme implemented by the SPS.

**To enable cookieless federation for SPS at the consuming side**

1. Open noodle.properties file from *sps_home*/secure-proxy/Tomcat/properties.
2. Remove the '#' from the following two lines, and save the file.
   - filter._cookielessfederation_.class=org.tigris.noodle.filters.CookielessFedFilter
   - filter._cookielessfederation_.order=1

   The settings are saved.
3. Open the server.conf file located at *sps_home*/secure-proxy/proxy-engine/conf.
4. Add the following code to the virtual host section for the virtual host that is serving the FWS:

   cookielessfederation="yes"
5. Save the file.

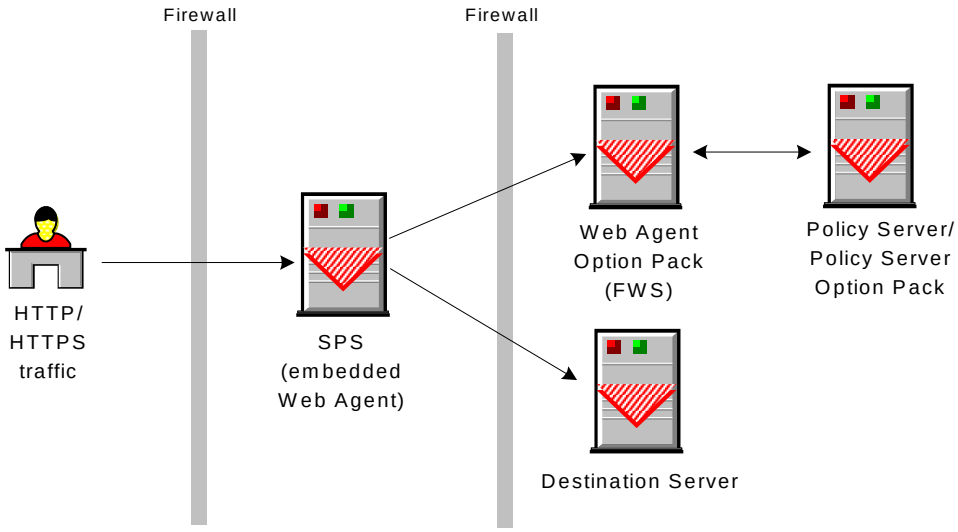
   The SPS is configured for cookieless federation at the consuming partner.

# SPS as a Web Agent Replacement

To provide federated single sign-on, the SPS may be used as a substitute for the SiteMinder Web Agent. The SPS, and the Web Agent Option Pack combine to provide the Federation Web Services (FWS) application, which is a collection of servlets packaged as a Web application. This application provides much of the SiteMinder federation functionality.

Knowledge of SiteMinder Federation Security Services is required for anyone configuring SPS in a federated environment. For more information on Federation Security Services, see the *CA SiteMinder Federation Security Services Guide*.

The following figure shows an environment where the SPS replaces a SiteMinder Web Agent.




**Important!** If you choose to use the SPS in place of the Web Agent for a federated environment, the Web Agent Option Pack requires a dedicated web server and servlet engine separate from the web server and servlet engine included in the SPS.

## Prerequisites for Using the SPS as a Web Agent Replacement

Before you configure SPS for use in a SiteMinder Federation Security Services environment, consider the following:

- The SiteMinder environment must be configured according to the information in the *CA SiteMinder Federation Security Services Guide*. We recommend that you configure a Federation environment with a standard Web Agent to confirm that Federation Security Services is configured properly.
- After you confirm that the federation environment is working properly, install the Web Agent Option Pack on the SPS system, or on a separate system.
- Install a Servlet Engine for use by the Web Agent Option Pack.

  For more information about FSS and servlet engines, see the *CA SiteMinder Federation Security Services Guide*.
- In the SiteMinder Policy Server User Interface, define the host information (server and port number) for the SPS system that generates assertions. The SPS host is defined in the Server field of the appropriate properties dialog for the federated partner you are specifying.

## Configuring the SPS as a Web Agent Replacement for Federation

The configuration process for the SPS to operate in a federated environment is similar to the standard SPS configuration process.

The overall configuration process for the SPS federation gateway is as follows:

1. Install the SPS.
2. Run the configuration wizard.
3. Specify the general server settings in the server.conf file. Though there are defaults for most of the server.conf settings, you can modify such settings as logging, session schemes, or virtual host settings.
4. Define proxy rules in the proxyrules.xml file so that requests are directed to the backend servers.

   At the enterprise producing assertions, define a proxy rule that forwards requests to the backend server hosting FWS. At the side consuming assertions, there must be a rule that forwards requests to the destination server after the user is permitted access to the target resource.
5. (Optional) If you want to configure virtual hosts for the SPS, you can modify the Apache web server file (httpd.conf), for example,

**More information:**

Configuring the Apache Web Server (see page 81)
Configuring the SPS Server Settings (see page 83)
Configuring Proxy Rules (see page 141)

# SPS as a Federation Gateway

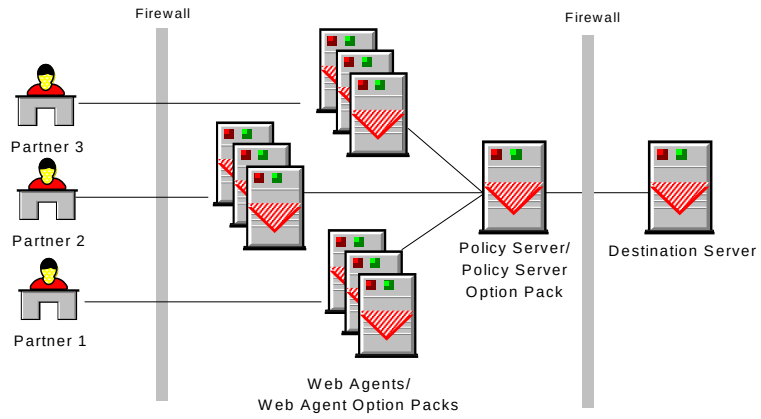
The SPS federation gateway simplifies the configuration involved in a federated environment. Typically, you have a federated environment where partners are communicating through many web servers. Each web server requires that you install and configure the Web Agent and the Web Agent Option Pack.

If you enable the SPS as a federation gateway, the number of components that you have to install and set-up is reduced. The SPS federation gateway has the standard embedded components of the SPS and the Federation Web Services application provided by the Web Agent Option Pack.

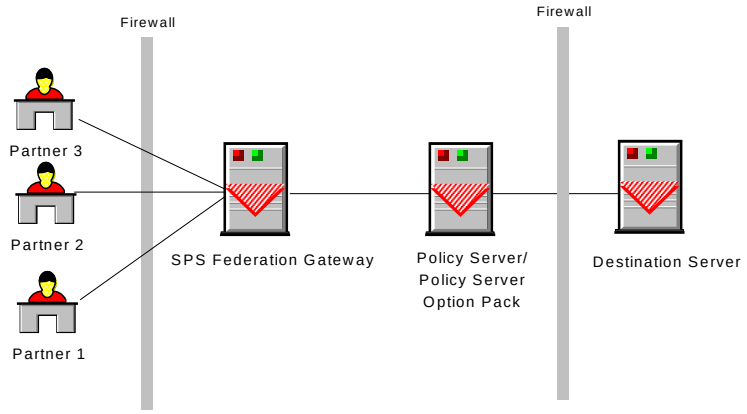
**Note:** Knowledge of SiteMinder Federation Security Services is required for anyone configuring SPS in a federated environment. For more information about Federation Security Services, see the *CA SiteMinder Federation Security Services Guide*.

The following illustration shows the difference with or without the SPS federation gateway.

**Federated Environment without the SPS Federation Gateway**




**Federated Environment with the SPS Federation Gateway**



## Prerequisites for Using the Federation Gateway

Before you set up the SPS as a federation gateway, consider the following:

- The SiteMinder environment must be configured according to the information in the *CA SiteMinder Federation Security Services Guide*. Verify that the Policy Server side components for federation are configured.
- Install the SPS and enable the enablefederationgateway setting when prompted.
- In the SiteMinder Policy Server User Interface, be sure to define the host information (server and port number) for the SPS system that generate assertions. The SPS host is defined in the Server field of the appropriate properties dialog for the federated partner you are specifying.

## Configuring the SPS Federation Gateway

The SPS federation gateway can sit at the producer site and consumer site.

The overall configuration process for the SPS federation gateway is as follows:

1. Install the SPS.
2. Run the configuration wizard.
3. Specify the general server settings in the server.conf file. Though there are defaults for most of the server.conf settings, you may want to modify such settings as session schemes or virtual host settings.
4. Define proxy rules in the proxyrules.xml file so that requests are directed to the backend servers.

   At the enterprise producing assertions, federation requests are forwarded to the Tomcat server embedded in the SPS. The Tomcat server hosts the FWS application. Proxy rules and filters have no relevance when the federation request gets processed.

   At the enterprise consuming assertions, you need to define a proxy rule that forwards requests to the destination server after the user is permitted access to the target resource.
5. (Optional) You can modify the Apache web server file (httpd.conf).

## Limitations of the SPS Federation Gateway

Note the following limitations when using the SPS federation gateway:

- The prefilters and postfilters (both built-in and custom-configured) do not execute when federation resources are being requested. For non-federated requests that are fired for the default context, these filters execute as usual.
- Proxy rules do not execute when federated resources are being requested. For non-federated requests that are fired for the default context, these rules execute as usual.

# Chapter 5: Security Zones on SPS

This section contains the following topics:

Overview Single Sign-on Security Zones (see page 75)
Security Zones Benefits (see page 76)
Security Zone Basic Use Case (see page 77)
Parameters for Security Zones (see page 77)
Configure the SPS Security Zones (see page 79)

## Overview Single Sign-on Security Zones

SSO security zones provide configurable trust relationships between groups of applications within the same cookie domain. Users have single sign-on within the same zone, but can be challenged when entering a different zone, depending on the trust relationship defined between the zones. Zones included in a trusted relationship do not challenge a user that has a valid session in any zone in the group.

CA SiteMinder Web Agents implement single sign-on security zones. Each zone must reside on a separate Web Agent instance. All Web Agents configured through the same agent configuration object belong to the same single sign-on zone.

Cookies generated by the Web Agent identity security zones. By default, the Web Agent generates two cookies: a session cookie named SMSESSION, and an identity cookie named SMIDENTITY. When you configure security zones, the Web Agent generates session cookies and identity cookies with unique names so that the zone affiliation is reflected in the cookie names.

**Note:** For detailed information about SSO security zones, see the *CA SiteMinder Web Agent Guide*.

# Security Zones Benefits

The SSO Security Zones feature is intended for use in situations where CA SiteMinder administrators wish to segment their single sign-on environments within the same cookie domain. For example, consider the CA.COM domain. Under standard CA SiteMinder SSO functionality, all CA SiteMinder protected applications in CA.COM would use the cookie SMSESSION to manage single sign-on. Consider the following scenario in which SSO Security Zones do not exist:

1. A user accesses an application (APP1). The user is challenged by CA SiteMinder, logs in to CA SiteMinder, and creates an SMSESSION cookie.
2. The user accesses a second application (APP2) and is once again challenged by CA SiteMinder. (Rules prevent SSO from occurring because the user does not have access to APP2 using the APP1 user credentials.) The user logs in and creates a new SMSESSION cookie overwriting the old one with the new logged in session for APP2.
3. The user now returns to APP1 and is challenged yet again, since the user lost the original APP1 session and the APP2 session might not be accepted for APP1. Therefore, SSO does not occur between APP1 and APP2, causing a very frustrating situation.

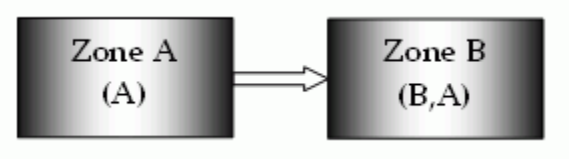
With SSO Security Zones, APP1 can be placed in zone Z1 and APP2 can be placed in zone Z2. Now logging into APP1 creates a Z1SESSION cookie and access to APP2 results in a Z2SESSION cookie. With different names, the cookies no longer overwrite each other so there is only one login per application now, not one for each time the user moves between the applications as in the example above.

Prior to the SSO Security Zones feature, the only way to perform the same grouping of SSO for applications was to create different network domains and therefore different cookie domains (CA1.COM, CA2.COM, and so on), and use various multi-cookie domain configurations with cookie providers. This is not desirable in most enterprises, since using multiple network domains has certain IT maintenance and support consequences.

# Security Zone Basic Use Case

Single sign-on can, on a controlled basis, be broken into several security zones that have configurable trust relationships. For example, consider Zone A and Zone B:

- Zone A has only one trusted zone, its own Zone A.
- Zone B has two trusted zones, its own Zone B as well as Zone A.




The trust relationship in the above illustration is indicated by the arrow, meaning that the user sessions established in Zone A can be used for single sign-on in Zone B.

In this example, Zone A might be an administrator-only zone, while Zone B might be a common access zone. An administrator authenticated in Zone A gains access to Zone B without being rechallenged. However, a user authenticated in Zone B is re-challenged when trying to access Zone A.

User sessions in different zones are independent of each other. Suppose a user authenticates in Zone B first, and then authenticates again in Zone B. Two different sessions are created. In fact, the user may have different identities in both sessions. When the user returns to Zone A, the session established in that zone is used.

Consider what would happen if a user is validated using single sign-on in a zone where that user does not yet have a session. If the user authenticates in Zone A and then visits Zone B for the first time, then a user session is created in Zone B, based on the session information in Zone A, possibly updated by the Policy Server. Note that the user session in Zone A is not updated until the user returns to Zone A.

# Parameters for Security Zones

The two single sign-on parameters listed following are manually added to the Web Agent configuration objects in the policy store. These settings can also be used in local configuration files and are added to the sample local configuration files laid down during installation.

**SSOZoneName**

Specifies the (case-sensitive) name of the single sign-on security zone a Web Agent supports. The value of this parameter is prepended to the name of the cookie a Web Agent creates. When this parameter is not empty, CA SiteMinder generates cookies using the following convention: *ZonenameCookiename*. The default is empty and uses SM as a zone name, which gives the cookies the following default names:

- SMSESSION
- SMIDENTITY
- SMDATA
- SMTRYNO
- SMCHALLENGE
- SMONDENIEDREDIR

Example: Setting the value to Z1 creates the following cookies:

- Z1SESSION
- Z1IDENTITY
- Z1DATA
- Z1TRYNO
- Z1CHALLENGE
- Z1ONDENIEDREDIR

**SSOTrustedZone**

Defines an ordered (case-sensitive) list of trusted SSOZoneNames of trust for a single sign-on security zone. Use SM to add the default zone if necessary. Agents always trust their own SSOZoneName above all other trusted single sign-on zones. The default is empty, or can be SM or the SSOZoneName if provided.

# Configure the SPS Security Zones

You can configure security zones on the SPS in one of the following methods:

- Configure security zones on multiple SPS servers that are configured to a Policy Server based on their ACO objects.
- Configure security zones on multiple web servers that are deployed behind an SPS server.

To configure security zones on multiple SPS servers, perform the following steps:

1. Configure the SSOZoneName parameter on the first SPS server.
2. Configure the SSOZoneName and SSOTrustedZone parameters on the SPS servers you want to group as one security zone or different security zones.

To configure security zones on multiple Web servers of an SPS server, perform the following steps:

1. Create an ACO for each Web server that must belong to a security zone.
2. Create a virtual host for a single or group of Web servers that must belong to a security zone.
3. Verify that a unique ACO points to a virtual host so that each virtual host belongs to a different security zone.
4. Configure the SSOZoneName parameter on the ACO of the first Web server.
5. Configure the SSOZoneName and SSOTrustedZone parameters on the ACOs of the virtual hosts you want to group as one security zone or different security zones.

# Chapter 6: Configuring the Apache Web Server

This section contains the following topics:

Apache Web Server Configuration File (see page 81)

## Apache Web Server Configuration File

The SPS proxy engine works with an embedded Apache web server. If, for example, you want to configure virtual hosts for the SPS, you can modify the Apache web server configuration.

The configuration file for the Apache web server is the httpd.conf file, which is located in:

*sps_home*/secure-proxy/httpd/conf/

**Important!** If you change any Apache setting while upgrading the SPS, or during any other reconfiguration scenario, restart the SPS services to reflect the changes. In addition, restart the SPS with the new settings (for example, the new port number).

# Chapter 7: Configuring the SPS Server Settings

## SPS server.conf File Overview

The SPS is configured through settings contained in the server.conf file. These settings in the file are groups of name/value pairs or *directives* that the SPS reads at startup.

After the SPS is operating, it verifies the values in this file to determine if any changes have been made to the SPS Web Agent log level settings. If changes are detected, the affected settings are reloaded so that the SPS can be dynamically updated without interrupting network traffic.

The server.conf file is located in the following directory:

*sps_home*/secure-proxy/proxy-engine/conf

The file contents are grouped into the following sections:

- Server—Includes settings for the server operation, federation gateway operation, and SSL.
- Session Store—Defines the session store.
- Service Dispatcher—Defines setting for this global server parameter.
- Proxy and Redirect Services—Specifies the connection pools and filters for the proxy service and the class for the redirect services.
- Session schemes—Defines the session schemes.
- User agents—Specifies types of user agents.
- Virtual Hosts—Identifies the default virtual host and its settings.

Each section is an XML-like element tag. The name of the section is the beginning tag of the XML element and the section ends with a corresponding ending tag. The directives contained in each section follow the format name=value.

Any lines beginning with the # symbol are comments, and are not read when the SPS loads configuration settings.

**Note:** Pathnames on Windows systems use double backslashes (\\), such as \\logs\\server.log

# Modifying the server.conf File

The settings for the SPS are maintained in the server.conf file located in the following directory:

*sps_home*/secure-proxy/proxy-engine/conf

**To change the settings in the server.conf file**

1. Open the file in a text editor.
2. Edit the directives, as necessary.
3. Restart the SPS.

The settings are changed.

# General Server Settings in the server.conf File

The <Server> section of the server.conf file contains parameters for server connectors, Federation, and SSL.These parameters are described in the sections that follow.

## HTTP Connection Parameters

```
#Define the listeners between #HTTP listener and proxy engine.

worker.ajp13.port=8009

worker.ajp13.host=localhost

worker.ajp13.reply_timeout=0

worker.ajp13.retries=2
```

**Note:** The values for the connector directives are not contained in quotes. Values for other types of directives are contained in quotes.

The name/value pairs are:

**worker.ajp13.port=8009**

Specifies the port for the ajp13 connector.

**worker.ajp13.host=localhost**

Specifies the local ajp13 host as the local host.

Additional tuning parameters can be defined for the connection between the HTTP listener and the proxy engine, including:

**worker.ajp13.reply_timeout**

Specified the maximum time in milliseconds that can elapse between any two packets received from the proxy engine after which the connection between HTTP

listener and the proxy engine is dropped. A value of zero makes it wait indefinitely, until a response is received.
**Default:** 0

**worker.ajp13.retries**

Specifies the maximum number of times that the worker sends a request to the proxy engine in a communication error.

**Default:** 2

## Tomcat Tuning Parameters in the server.conf File

A Tomcat server is embedded in the SPS. The Tomcat server provides a servlet container and servlet engine.

The following excerpt is from the Tomcat tuning section in the server.conf file:

```
#Define AJP13 tuning parameters
        #Number of request waiting in queue (queue length)
        #Number of threads created at initialization time
        #Maximum number of concurrent connections possible
        worker.ajp13.accept_count=10
        worker.ajp13.min_spare_threads=10
        worker.ajp13.max_threads=100
        worker.apj13.connection_pool_timeout=0
        worker.apj13.max_packet_size=8192
```

The Tomcat tuning directives are listed following.

**worker.ajp13.accept_count**

Defines the number of requests waiting in the queue when all possible request processing threads are in use. Any requests received when the queue is filled are refused.

**Default:** 10

**worker.ajp13.min_spare_threads**

Defines the minimum number of idle threads at any time, waiting for new requests to arrive. min_spare_threads must be greater than 0.

**Default:** 10

**worker.ajp13.max_threads**

Defines the maximum number of concurrent connections possible, the pool will not create more than this number of threads.

**Default:** 100

**worker.apj13.connection_pool_timeout**

Defines the maximum time (in seconds) that the idle connections (between apache and tomcat over mod-jk) remain in the connection pool before timing out. The default is zero, which means that connections never timeout.

**Default**: 0

**worker.ajp13.max_packet_size**

Defines the maximum packet size in Bytes. The maximum value is 65536

**Default**: 8192

## Resolve Differences in Cookie Specification for Different Version of Tomcat

Tomcat version 5.5 changed its behavior for handling cookies. Tomcat version 5.5 by default puts quotes around a cookie. Previous versions of Tomcat did not put quotes around a cookie. Tomcat sends the cookie back to the browser. SPS r12.0 SP 3 uses Tomcat version 5.5. If your deployment requires visiting an earlier version of SPS, the cookie cannot be decoded. The earlier version of SPS is using Tomcat version 5.0, which does not put quotes around the cookie.

To ensure that the cookie behavior is compatible between different versions of SPS, set the addquotestobrowsercookie parameter in the server.conf file to "no". The Tomcat org.apache.catalina.STRICT_SERVLET_COMPLIANCE variable is set to "TRUE". Tomcat parses the cookie according to the servlet specification, which means that no quotes are added. When the addquotestobrowsercookie parameter is set to "yes", the SPS enables the default Tomcat version 5.5 cookie behavior.

## Parsing the Equal Sign in a Cookie

Tomcat 5.5 and later adds an equals (=) sign to the cookie. The SPS allows this practice and parses cookie values that contain an equal sign. The default value for the allowequalsincookievalue parameter in the server.conf file is "yes".

Set the allowequalsincookievalue parameter to "no" if you want parsing of the cookie value to terminate when the parser encounters an equal sign.

## Federation Settings in the server.conf File

The federation settings in the server.conf file enable the SPS to act as a federation gateway within a SiteMinder federation network.

The code excerpt that follows is the <federation> section on the server.conf file:

```
# Provide the values for the Federation related parameters here
#
# enablefederationgateway - "yes" or "no" - Enable or Disable SPS Federation Gateway
# fedrootcontext - Name of the Federation root context ("affwebservices" by default)
# authurlcontext - Path of the Authentication URL (without the jsp file name)
  (siteminderagent/redirectjsp by default)
# protectedbackchannelservices - Names of protected Backchannel services


<federation>

    enablefederationgateway="yes"
    fedrootcontext="affwebservices"
    authurlcontext="siteminderagent/redirectjsp"
        protectedbackchannelservices="saml2artifactresolution,saml2certartifactre
    solution,
    saml2attributeservice,saml2certattributeservice,assertionretriever,certassert
    ionretriever"
</federation>
```

The federation parameters are as follows:

**enablefederationgateway**

Enables the SPS to act as a federation gateway proxy server.

**Limits:** yes or no

This parameter is set during the installation.

**fedrootcontext**

Specifies the root context of the federation web services application. Do not change this parameter.

**Default:** affwebservices

**authurlcontext**

Specifies the alias to the redirect.jsp file. When a user requests a protected federation resource and they do not have a SiteMinder session at the site that produces assertions, the user is sent to this URL which points to a redirect.jsp file. The user is redirected to the Web Agent at the producing site where they are presented with an authentication challenge and upon successfully logging in, establish a session.

**Default:** siteminderagent/redirectjsp.

**protectedbackchannelservices**

Lists the services that require a secure back channel for communication.

## HttpClient Logging

You can enable HttpLogging by setting the httpclientlog parameter to "yes". This parameter in located in the <Server> section of the server.conf file. By default, this parameter is set to "no".

We recommend that you enable HttpClient logging only for debugging. In a production environment, enabling logging can cause performance degradation.

### Configure HttpClient Logging

You can configure various aspects of HttpClient logging by setting values to parameters in the httpclientlogging.properties file. This file is located in the *sps_home*\Tomcat\properties directory.

**Important!** Because of potential performance degradation, do not enable HttpClient logging in a production environment.

The httpclientlogging.properties file has the following configurable parameters:

**java.util.logging.FileHandler.formatter**

Description: Specifies the name of the formatter class

Limits:

java.util.logging.SimpleFormatter — writes brief summaries of log records

java.util.logging.XMLFormatter — writes detailed descriptions in XML format

Default: java.util.logging.SimpleFormatter

**java.util.logging.FileHandler.pattern**

Description: Specifies the name of the HttpClient log file.

Limits:

sps_home\proxy-engine\logs\httpclient%g.log

%g represents the generation number of the rotated log file.

**java.util.logging.FileHandler.count**

Description: Specifies the number of output files in a cycle

Default: 10

**java.util.logging.FileHandler.limit**

Description: Specifies an approximate maximum number of bytes to write to any on log file.

Limits: If set to zero, there is no limit.

Default: 5,000,000

## SSL Settings in the server.conf File

The <sslparams> section in the server.conf file contains the settings that are required to enable Secure Sockets Layer (SSL) communications between the SPS and destination servers.

The SSL configuration section is listed following.

```
<sslparams>
    # Set the SSL protocol version to support:SSLv3, TLSv1
    # NOTE: SSL version 2 is no longer supported versions="SSLv3"

    ciphers="-RSA_With_Null_SHA,+RSA_With_Null_MD5,-RSA_With_RC4_SHA,+RSA_With_RC
    4_MD5,+RSA_With_DES_CBC_SHA,+RSA_Export_With_RC4_40_MD5,-RSA_Export_With_DES_
    40_CBC_SHA,+RSA_Export_With_RC2_40_CBC_MD5,-DH_RSA_With_DES_CBC_SHA,-DH_RSA_W
    ith_3DES_EDE_CBC_SHA,-DH_RSA_Export_With_DES_40_CBC_SHA,-DH_DSS_With_DES_CBC_
    SHA,-DH_DSS_Export_With_DES_40_CBC_SHA,-DH_Anon_With_RC4_MD5,-DH_Anon_With_DE
    S_CBC_SHA,-DH_Anon_With_3DES_EDE_CBC_SHA,-DH_Anon_Export_With_DES_40_CBC_SHA,
    -DH_Anon_Export_With_RC4_40_MD5,-DHE_RSA_With_DES_CBC_SHA,-DHE_RSA_Export_Wit
    h_DES_40_CBC_SHA,-DHE_DSS_With_DES_CBC_SHA,-DHE_DSS_Export_With_DES_40_CBC_SH
    A"

    fipsciphers="+DHE_DSS_With_AES_256_CBC_SHA, +DHE_RSA_With_AES_256_CBC_SHA,
    +RSA_With_AES_256_CBC_SHA, +DH_DSS_With_AES_256_CBC_SHA,
    +DH_RSA_With_AES_256_CBC_SHA, +DHE_DSS_With_AES_128_CBC_SHA,
    +DHE_RSA_With_AES_128_CBC_SHA, +RSA_With_AES_128_CBC_SHA,
    +DH_DSS_With_AES_128_CBC_SHA, +DH_RSA_With_AES_128_CBC_SHA,
    +DHE_DSS_With_3DES_EDE_CBC_SHA, +DHE_RSA_With_3DES_EDE_CBC_SHA,
    +RSA_With_3DES_EDE_CBC_SHA, +DH_DSS_With_3DES_EDE_CBC_SHA"

    # Covalent SSL CA certificate bundle and certs path to be converted
    # The bundle and/or certs located at defined location will be converted
    # to binary (DER) format and loaded as SSLParams.
    # NOTE: Only put Base64 (PEM) encoded cert files/bundles in the covalent
    # certificate directory.
    cacertpath="<install-dir>\SSL\certs"
    cacertfilename="<install-dir>\SSL\certs\ca-bundle.cert"

    # This certificate configured below is used as SPS client certificate for the
    backend servers when
    # SSL client authentication is enabled.
    # Location of the Key file : <install-dir>\SSL\clientcert\key\
    # Location of public certs : <install-dir>\SSL\clientcert\certs\
    # NOTE: Only put DER encoded, password encrypted pkcs8 keyfile.
    # Client pass phrase should be encrypted using EncryptUtil tool.
    #ClientKeyFile=
    #ClientPassPhrase=

</sslparams>
```

The SSL parameters include:

**versions**

Determines the SSL versions supported by the SPS. The entry can be one or more of the following.

- SSLV3
- TLSV1

If you specify more than one version, separate the values by commas.

**ciphers**

Specifies the list of ciphers that can be enabled or disabled. If a cipher is enabled, it is preceded by a + symbol. If a cipher is disabled, it is preceded by a - symbol. If you specify more than one cipher, separate each entry by commas.

**cacertpath**

Specifies the path of the directory that contains the trusted certificate authority information. This path is relative to the install path of the SPS. This value is configured when you run the configuration wizard during the SPS installation; do not change it.

**cacertfilename**

Specifies the fully qualified path name of the file that contains the Certificate Authority bundle of certificates. This file must have a file extension of .cer or .cert, and must be PEM encoded. It must also include the full path to the Certificate Authority (CA) bundle. This value is configured when you run the configuration wizard during the SPS installation.

**ClientKeyFile**

Specifies the file name of the SPS client certificate key in DER-encoded and password-encrypted pkcs8 format. Verify that the file is located in the following location:

*<SPS Installation Path>*/SSL/clientcert/key

**ClientPassPhrase**

Specifies the passphrase that extracts the key from the SPS client certificate key file using the EncryptUtil tool.

**maxcachetime**

Specifies the duration, in milliseconds, that the SSL session ID is cached for re-use by the SPS HTTPS client. When a user requests a file via an HTTPS connection, an SSL handshake occurs and an SSL session ID is created. This SSL session ID is used by the SPS and the backend server to identify a user session. When the HTTPS connection is terminated for the user, the SPS caches the SSL session ID for the maximum duration specified by this parameter.

When the same user requests a new HTTPS connection to the backend server, the user can send the SSL session ID that is cached for a faster response. In this case, the SSL session ID provided by the user is compared with the cached SSL session ID. If the SSL session ID is available in cache, the new HTTPS connection is established faster.

**Default:** 120000 milliseconds

**Note**: Before you enable SSL communication, verify that you have installed the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files in the JDK location that was used to install the SPS.

## Client Certificate Authentication

You can also establish a client certificate authentication for a secured SSL connection between the SPS and a web server. If you enable the client certificate authentication, the SPS uses a client certificate to authenticate itself when requesting a resource from a web server.

## Prerequisites

Before you configure the SPS to enable client certificate authentication, verify that the following tasks are complete:

- Client certificate authentication is enabled on the web server you want to direct user requests.
- A client certificate is generated for the SPS in a pkcs8 standard using the openssl.exe utility that is installed during the SPS installation.
- If an existing client certificate is used, it is converted into the pkcs8 DER format.
- The public certificate and the root certificates of the SPS client certificate are at <*SPS_Installation_Path*>\SSL\Clientcert\certs.
- The public certificate of the configured web server is placed at <*SPS_Installation_Path*>\SSL\certs\ca-bundle.cert.
- The public certificate of the SPS client certificate is placed in the trusted store of the configured web server.

## Enable Client Certificate Authentication

Configure the SPS to enable the client certificate authentication.

**Follow these steps:**

1. Encrypt the password of the private key of the SPS client certificate by performing the following steps:
   a. Open the command prompt.
   b. Naviagte to the <*SPS_Installation_Path*>\SSL\bin location.
   c. Execute the following command:

      **Windows**

      ```
      EncryptUtil.bat <SPSCertificatePrivateKey_Password>
      ```

      **UNIX**

      ```
      EncryptUtil.sh <SPSCertificatePrivateKey_Password>
      ```

      The encrypted password is displayed.
2. Configure client certificate authentication details in the server.conf file by performing the following steps in the sslparams section:
   a. Enter the key file name of the SPS client certificate in pkcs8 format in ClientKeyFile.
   b. Enter the encrypted password that you generated in Step 1 in ClientPassPhrase.

   The client certificate authentication is configured in the server.conf file.
3. Configure the proxyrules.xml file to forward the client requests to the configured web server.
4. Restart the SPS.

   Client certificate authentication is enabled between the SPS and the web server.

## Setting for Special Characters within the Cookie

The server.conf file includes a parameter, addquotestocookie, for preserving the SPS practice of enclosing the cookie parameter value in double quotes when the value is not zero. You can change the value of addquotestocookie to "no" when you do not want the SPS to add double quotes around the cookie value before sending it to the backend.

The entry in the server.conf file appears as follows:

```
<Server>
.
.
<sslparams>
.
.
</sslparams>
#This parameter is applicable to the cookie added by backend.
#"yes"--- Default Value. Quotes will be added to the cookie parameter value
#which contains special characters if the cookie version is other than "0"
#"no" --- Quotes will not be added to the cookie.
addquotestocookie="yes"

</Server>
```

## Select Character Set for Code Headers

You can specify the character set for the appropriate locale by setting the value of the requestheadercharset parameter. The HttpClient application reads this value to determine how to encode the headers to send to the backend server. Possible values are:

- US-ASCII — Specifies the locale uses US English
- Shift_JIS — Specifies the locale uses Japanese, including support for Japanese usernames.

The default is requestheadercharset="US-ASCII".

## Caching POST Data

The following two parameters are included in the server.conf to enable POST data caching and to set the size of cached data:

**enablecachepostdata**

Description: Specifies whether the SPS caches POST data.

Limits: Yes, No

Default: No

**maxcachedpostdata**

Description: Specifies the size (in KB) of the POST data to cache.

Limits: None

Default: 1024 KB

## Ignoring the SSL Exceptions

You can configure SPS to ignore benign SSL exceptions when the backend web server returns the disconnect notification to SPS. To ignore benign SSL exceptions, set the ignoresslbackendexception parameter to yes.

# Custom Error Pages Parameters

The SPS supports the custom error pages feature, which lets you customize the error pages that the Web Server displays when a client request fails.

The custom error pages section has the following format:

```
<customerrorpages>
#possible values are: "yes","no"
#default value is "no"
enable="no|yes"
#custom error pages implementation class
class="com.netegrity.proxy.errorpages.ErrorPageImpl"
#defines type of locale.
#possible values are: "0" (for Server specific), "1" (for Browser specific)
#default value is "0"
locale_type="0|1"
#this value should be the language code that will be understood by the java
#locale object, say "zh" for Chinese, "fr" for French, "es" for Spanish, "en" for
#english, etc.
#default value is "en"
locale_language="en"
#this value should be the country/region code that will be understood by the
#java locale object, say "CN" for China, "CH" for Switzerland, "AR" for
#Argentina, "US" for United States.
#default value is "US"
locale_country="US"
#display the complete call stack for exceptions
          #possible values: "true" or "false".
          #default value: "false"
          showcallstack="true"
</customerrorpages>
```

**no|yes**

Specifies how the SPS manages the web server error pages. If the value is set to yes, you can customize the Web Server error pages and the Web Server displays the customized error pages when a client request fails. If the value is set to no, the Web Server displays the default HTTP 1.1 error pages when a client request fails. The SPS reads the value at the startup.

**Default**: no

**0|1**

Specifies the locale of the custom error pages. If the value is set to 0, the SPS uses the locale settings of the SPS server to display the custom error messages. If the value is set to 1, the SPS uses the locale settings of your browser to display the custom error messages.

**Default**: 0

**showcallstack**

Specifies if the call stack must be retrieved to track exceptions that you want to debug. If the value is set to yes, the SPS retrieves the call stack and displays it on the custom error page. If the value is set to false, the call stack is hidden.

**Default**: false

**Important!** For security reasons, we highly recommend that you do not enable the showcallstack option unless you want to track the complete details of an exception that you want to debug. Disable showcallstack after you complete debugging.

## Enable Custom Error Messages

Enable the feature to let the web server display a customized error page when a client request fails and customize the message or code of an error. By default, the SPS supports all HTTP 1.1 error codes.

**Follow these steps:**

1. Open the server.conf file.
2. Navigate to the customerrorpages section.
3. Set the following command:

   ```
   enable="yes"
   ```

4. Save the changes.
5. Restart the SPS server.

## Default Custom Error Pages

By default, the SPS provides a list of possible request errors that are received from the SPS and web servers. The error messages in the SPSErrorMessages.properties and WebServerErrorMessages.properties files are in the following format:

`exception|error code=`*`error message`*`|`*`URL`*

Where

**exception|error code**

Defines the exception or the error code that the SPS or web server returns when a user request fails.

**Limits**: 100 characters

***error message*|*URL***

Defines the error message the SPS or web server displays when an exception or error code is returned. You can specify either the error message in plain text or the targeted URL that the user must use.

**Limits**: 4096 characters

### Default SPS Error Pages

By default, the SPS displays an error page when a user request fails due to inappropriate configuration settings of the SPS server. Each error page is mapped to an error code. When a user request fails, the SPS checks for the error code and displays the corresponding error page.

The following table lists the default errors that the SPS displays due to inappropriate configuration settings of the SPS server:

| Error Code | Error Message |
|---|---|
| VirtualHostNotFound | Virtual host might not have been configured properly. Verify the virtual host settings in the server.conf file |
| SessionSchemeNotFound | The defined session scheme is not found. Verify the session scheme settings in the server.conf file |
| SessionCreateError | The session store might have been corrupted during creation. Restart the SPS |
| SessionUpdateError | The session store might have been corrupted during update. Restart the SPS |

| Error Code | Error Message |
|---|---|
| WebAgentException | The SPS might not be communicating with the Web agent. For more information about the error, see SPS logs |
| Noodle_GenericException | There might be an error during processing a request at noodle stage. For more information about the error, see SPS logs |
| Noodle_FilterException | There might be an error during pre/post processing of filters at noodle. |
| Noodle_UnknownHostException | The IP address of the targeted host could not be determined |
| Noodle_ConnectException | An error occurred while attempting to connect a socket to a remote address and port |
| Noodle_NoRouteHostException | An error occurred while attempting to connect a socket to a remote address and port because of an intervening firewall or unavailability of an intermediate router |
| Noodle_InteruptedIOException | An input or output transfer is terminated because the thread that is performing the transfer was interrupted |
| Noodle_SocketException | An error occurred in the underlying protocol |
| Noodle_NoSuchMethodException | The SPS does not support the method |
| Noodle_ProxytoProxyNotSupported | The SPS does not support proxy Web Server |
| Noodle_CouldNotConnectToBackEndServer | Could not connect to backend web server due to lack of processing threads at the SPS |
| Noodle_NoAvailableConnections | There are no connections available to serve the request |
| Redirect_NoTargetURLParameter | No URL is specified during Redirect |
| FedRequest_Redirect_ImproperURL | The redirect URL is not specified or is malformed. Verify the federation settings or the RelayState in the client request |
| FedRequest_Redirect_RewriteLocationHeaderFailure | Unable to create a redirect session because the memory holding the session keys might have crashed while processing the federation request |

| Error Code | Error Message |
|---|---|
| FedRequest_CookieProcessingError | Unable to create a cookie while processing the federation request |
| FedRequest_ResponseProcess_AddSessionError | An error occurred while adding the session during the federation request processing, the memory holding the session keys might have crashed |
| FedRequest_ResponseProcess_LocHeaderModifyError | An error occurred while updating the location header during federation request processing, the memory holding the session keys might have crashed |
| SPSException | An error occurred while processing the request. For more information about the error, see SPS logs |

## Modify a SPS Error Page

You can modify an error message of a SPS error page.

**Follow these steps:**

1. Open the SPSErrorMessages.properties file.

   **Default Path**: <*SPS_installation_folder*>\Tomcat\properties\
2. Navigate to the error record you want to modify.
3. Make the necessary change.
4. Save the changes.

## Default Web Server Error Pages

By default, the SPS supports all HTTP 1.1 error codes and the web server displays the default HTTP 1.1 error pages when a client request fails. When you enable the custom error pages feature, you can customize the error pages and the web server displays the customized error details when a client request fails. Each error page is mapped to an error code.

If you enable the feature and customize an error page, the web server displays the customized error page when the error occurs. If you enable the feature and do not customize an error page, the web server displays the default error page returned by the web server when the error occurs.

### Modify a Web Server Error Page

You can add, modify, or delete an error code or an error message of a web server error page. If you delete the error message of an error code, the SPS displays a page with the following message:

```
No custom message to display. Find more details in logs.
```

**Follow these steps:**

1. Open the WebServerErrorMessages.properties file.

   **Default Path**: <*SPS_installation_folder*>\Tomcat\properties\

2. Navigate to the error record you want to modify.
3. Make the necessary change.
4. Save the changes.

## Session Store Settings in the server.conf File

The <SessionStore> section of the server.conf file specifies settings for storing user sessions. The session store configuration has the following format:

```
<SessionStore>
    # Session Store Information
    class="com.netegrity.proxy.session.SimpleSessionStore"
    max_size="10000"
    clean_up_frequency="60"
</SessionStore>
```

The SessionStore parameters are:

**class**

Indicates the implementation used to maintain user session. Do not modify this value.

**Default:** com.netegrity.proxy.session.SimpleSessionStore

**max_size**

Specifies the maximum size of the session store. The number specified is the maximum number of concurrent sessions in the in-memory session store.

**Default:** 10000

**clean_up_frequency**

Sets the interval, in seconds, that the SPS waits before cleaning out expired sessions residing in the session store cache.

**Note:** A long session timeout can decrease the number of session cookies of encrypted and decrypted by the server, but can increase the total number of sessions maintained in cache. If there are users who connect infrequently, specify a shorter cache time and smaller cache size. However, if there are many users who return to your site frequently, use a longer cache time and larger cache size.

# Service Dispatcher Settings in the server.conf File

The <ServiceDispatcher> section determines how the SPS provide proxy services. It also specifies the location of the proxy rules XML configuration file.

**Note:** This parameter is a global server configuration parameter and is not configured for each individual virtual host.

The <ServiceDispatcher> section is listed following.

```
# Service Dispatcher
# This is new since proxy 6.0
# Service Dispatcher is now a global server configuration parameter and is no longer
# configured on a per virtual host basis.
<ServiceDispatcher>
    class="com.netegrity.proxy.service.SmProxyRules" rules_file=
    "C:\Program Files\CA\secure-proxy\proxy-engine\conf\proxyrules.xml"
</ServiceDispatcher>
```

The parameters in this section are:

**class**

Specifies the service dispatcher used by the SPS to route user requests. Do not change the default value.

**Default**: com.netegrity.proxy.service.SmProxyRules

**rules_file**

Specifies the location of the proxyrules.xml. file

**Default:** *sps_home*/secure-proxy/proxy-engine/conf/proxyrules.xml

# Proxy and Redirect Settings in the server.conf File

The <Service> section of the server.conf file consists of the Proxy Service and the Redirect Service.

The two proxy services predefined for the SPS are:

- forward
- redirect

These services each have a section in the file defined by the <Service name> element. Custom services are similarly defined in the server.conf file, including any parameters set by an administrator.

## Proxy Service Configuration

The forwarding service of the SPS forwards requests to the appropriate destination servers according to the conditions and cases in the proxy rules XML configuration file. The parameters for this service are defined in the <Service name="forward"> section of the server.conf file.

Many of the directives manage the connection pool maintained by the SPS. These directives help improve server performance by maintaining connections and alleviating the overhead of establishing a new connection for each request to a destination server.

Additional directives define proxy filters. Proxy filters can be defined here to perform processing tasks before a request is passed to a destination server, and after the destination server returns data to the SPS. Filter names are unique.

The following is an excerpt of the <Service name="forward"> section.

**Note:** The excerpt does not include most of the comments that you see when you look at the actual server.conf file.

```
# Proxy Service
<Service name="forward">
    class="org.tigris.noodle.Noodle"
    protocol.multiple="true"
    http_connection_pool_min_size"4"
    http_connection_pool_max_size="20"
    http_connection_pool_incremental_factor="4"
    http_connection_pool_connection_timeout="1"
    http_connection_pool_wait_timeout="0"
    http_connection_pool_max_attempts="3"
    http_connection_timeout="0"
    http_connection_stalecheck="false"

    # Proxy filters may be defined here to perform pre/post processing tasks.
    # The following format must be used to configure filters:
    # filter.<filter name>.class=<fully qualified filter class name> (required)
    # filter.<filter name>.init-param.<param name1>=<param value1> (optional)
    # filter.<filter name>.init-param.<param name2>=<param value2>
    # filter.<filter name>.init-param.<param name3>=<param value3>

    # The following example illustrates the use of custom filters in a group
    # Defines filter groups with valid Custom filter names.
    #groupfilter.group1="filter1,filter2"
</Service>
```

The parameters in the forward section are:

**class**

Specifies the implementation that provides forwarding services for the SPS. Do not change this value. This value is only exposed to accommodate the rare occasion when a custom service can forward requests specified in the proxy rules XML configuration file.

**Default:** org.tigris.noodle.Noodle

**protocol.multiple**

Indicates whether the SPS supports protocols other than HTTP. Specify one of the following values:

**true**

Indicates that protocols other than HTTP are supported. Currently, only HTTPS is supported as an additional protocol in the SPS. True is the default value for this directive.

**false**

Indicates that only the HTTP protocol is supported.

**http_connection_pool_min_size**

Sets the minimum number of connections to a single destination server that are available for processing user requests.

**Default:** 4

**http_connection_pool_max_size**

Sets the maximum number of connections between the SPS and a destination server.

**Default:** 20

**Important!** Each connection established by the SPS creates a socket. For UNIX operating systems, if the maximum size of the connection pool is large, you can increase the limit on file descriptors to accommodate the large number of sockets.

**http_connection_pool_incremental_factor**

Sets the number of connections to a destination server that the SPS opens when all available connections are being used to process requests.

**Default:** 4

**http_connection_pool_connection_timeout_unit**

Sets the timeout unit to seconds or minutes.

**Default:** Minutes

**http_connection_pool_connection_timeout**

Defines the time, in minutes, the system waits before closing idle connections in the connection pool.

**Default:** 1

**http_connection_pool_wait_timeout**

Defines the time, in milliseconds, that the SPS waits for an available connection.

**Default:** 0

The default, 0, specifies that SPS waits for a connection until notified and invalidates the use of http_connection_pool_max_attempts.

**http_connection_pool_max_attempts**

Indicates the number of attempts that the system makes to obtain a connection. This directive is only applicable if wait timeout is not zero.

**Default:** 3

Specify one of the following values:

**0**

Indicates that the SPS makes attempts indefinitely.

**3**

Indicates that the SPS makes three attempts.

**http_connection_timeout**

Defines the time, in milliseconds, spent on host name translation and establishing the connection with the server when creating sockets.

**Default:** 0; indicates that the system does not enforce a limit.

**Note:** This timeout explicitly refers to the HTTP connection and not to the connection pool.

**http_connection_stalecheck**

Specifies if a stale connection check must be performed. If you set the value to true, a stale connection check is performed before each request execution. If you set the value to false, an I/O error may appear when you execute a request over a connection that is closed at the backend web server.

**Default**: false

**filter.*filter name*.class=*fully qualified filter class name***

Specifies the filter configured in the server.conf file for each unique filter that is invoked in the proxy rules.

**Example:** filter.PreProcess.class=SampleFilter

**filter.*filter name*.init-param.*param name1=param value1***

Specifies the initialization parameters for a filter based on how the filters are defined using the Filter API. Configure the server.conf file to define parameters for each filter.

**Example:** filter.PreProcess.init-param.param1=value1

**groupfilter.<groupname> = "*filtername1,filtername2,…….filtername*"**

Specifies the filter groups to implement one or more filters for a given proxy rule. The SPS reads the filter names declared in the group filter and processes the filters in a chain. The groupfilter name can be similarly used as a filter name in proxyrules.xml. When SPS processes a group filter, the pre-filters are processed before post filters even if the order in which they are defined in the groupfilter is reverse.

The following limitations are applicable:

- The filter names must be valid and unique.
- The group filter name must be unique. If you give the same group name for more than one group, only last group survives.
- The group filter name and filter names must be different.

**Example**:

groupfilter.BatchProcess="SampleFilter1, SampleFilter2, SampleFilter3"

## Connection Pooling Recommendations

Connection pooling is an important part of managing SPS performance. For SPS to provide the best possible service in an enterprise, destination servers must be configured with Keep-Alive messages enabled for connections. Enabling Keep-Alive messages for a destination server allows the SPS to use its connection pooling features.

Keep Alive messages are managed differently for each type of web server.

In addition to enabling Keep-Alive messages, the following settings are recommended for destination servers and the SPS. The table lists the timeout and connection pool recommendations:

| Settings | HTTP | HTTPS |
|---|---|---|
| Destination Server Keep-Alive Maximum Requests<br>(http_connection_pool_max_attempts) | Unlimited | Unlimited |
| Destination Server Timeout | Does not time out | Equal to or greater than the HTTP Connection Pool Timeout |
| Secure Proxy Server HTTP Connection Pool Timeout Unit<br>(http_connection_pool_connection_timeout_unit) | Set to seconds or minutes; default is minutes. | Set to seconds or minutes; default is minutes. |

| Settings | HTTP | HTTPS |
|---|---|---|
| Secure Proxy Server HTTP Connection Pool Timeout<br>(http_connection_pool_connection_timeout) | 1 minute | 1 minute |
| Secure Proxy Server HTTP Connection Pool Wait Timeout<br>(http_connection_pool_wait_timeout) | 0<br>waits until notified | 0<br>waits until notified |
| Secure Proxy Server HTTP Connection Pool Maximum Attempts<br>(http_connection_pool_max_attempts) | 3<br>value is only useful if the HTTP Connection Pool Timeout is greater than 0 | 3<br>value is only useful if the HTTP Connection Pool Timeout is greater than 0 |
| Secure Proxy Server HTTP Connection Timeout<br>(http_connection_timeout) | 0<br>does not timeout | 0<br>does not timeout |

## Redirect Service Configuration

The redirect service of the SPS sends requests to destination servers. Unlike the forward service, the destination server handles subsequent requests, not the SPS.

The redirect service has the following format:

```
<Service name="redirect">
        class=com.netegrity.proxy.service.RedirectService
</Service>
```

The directive is:

**class**

Indicates the implementation that handles redirected requests. This directive must not be modified.

**Default:** com.netegrity.proxy.service.RedirectService

## Connection-oriented Connection Pools Configuration

If the web server uses a connection-oriented authentication scheme, configure a connection-oriented connection pool for secure forward request processing.

**Important!** We highly recommend that you do not configure a connection-oriented connection pool.

**Follow these steps:**

1. Verify that the value for the JK environment variable REMOTE_PORT is set in the httpd.conf file.
2. Open server.conf and add the following lines in <Service name="forward"> section:

```
# Pool configuraiton for connection oriented authentication backend
    # connections eg: NTLM.
    <connection-pool name="connection oriented authentication">
    connection-timeout="connection_timeout_value"
    max-size="maximum_connections"
    enabled="yes|no"
</connection-pool>
```

***connection_timeout_value***

Defines the time in seconds the connection times out. We recommend that you set a lower value.

**Default**: 5

***maximum_connections***

Defines the number of connections in the connection pool.

**Default**: 50

***yes|no***

Specifies the status of the connection-oriented connection pools. Set the value to yes to enable the connection-oriented connection pools.

**Default**: yes

3. Open proxyrules.xml and add the connection-auth attribute to the forward rule.

```
Example: <nete:forward connection-auth="yes">hostname:port$1</nete:forward>
```

# Session Scheme Settings in the server.conf File

Session schemes determine how a user's identity is maintained, providing single sign-on during the course of a session. Each potential session scheme must be included in a SessionScheme section of the server.conf. file. Session schemes must be associated with a Java class file that defines the behavior of the session. If no session scheme is specified for a particular type of user agent, the default session scheme is used.

One challenge for enterprise transactions is maintaining user sessions. SiteMinder uses cookies to encapsulate session information. Unlike SiteMinder, the SPS uses several methods and provides a set of APIs to support alternative methods of maintaining sessions that does not rely on cookies. The cookieless session schemes involve some sort of token that reference session information maintained in the SPS in-memory session store. The session store resides in the SPS server's memory, and can be cleared by restarting the server.

The SPS provides the following out-of-the-box session schemes that you can configure in the server.conf file. These schemes may be associated with user agent types for each virtual host defined in the server.conf file. The association of a session scheme with a user agent type is called a session scheme mapping.

SPS includes the following schemes:

- Default scheme
- SSL ID
- IP Address
- Mini-Cookies
- Simple URL Rewriting
- Device ID

**Note**

- To create additional custom session schemes, you can use the session scheme APIs. If you create your own session schemes using the session scheme API, you must add a <SessionScheme> section to the server.conf file with specific information about the name and Java class associated with your custom session scheme.
- If you want to create custom session schemes using makefile.cygwin, set the values for JAVA_HOME and SPS_HOME as per the cygwin guidelines. For example, if you want to use makefile.cygwin on Windows 2008 R2 computer, you can set the following values:

  ```
  JAVA_HOME=C:/Progra~2/Java/jdk1.7.0_03

  SPS_HOME=C:/Progra~2/CA/secure-proxy
  ```

## Establishing a User Session

There are distinct phases for establishing a user session, as follows:

1. Discovery phase

   During this phase of a session, the SPS looks for an appropriate session key based on the user agent type. Session keys are either SiteMinder cookies, or a token that points to the appropriate information in the SPS in-memory session store. As previously discussed, tokens may be in the form of mini-cookies, SSL IDs, device IDs, or other tokens. If no session key can be identified, the Web Agent in the SPS takes over and forwards the request for authentication and authorization and establishes the identity and entitlements of the user.

2. Agent Handling phase

   The SPS contains a Web Agent that communicates with SiteMinder. The Web Agent is responsible for decrypting SiteMinder session information and validating a session with SiteMinder. If a user's request is accompanied by an SMSESSION cookie, or the SPS has located a user's session in the session store, the Web Agent validates a user's request with SiteMinder.

3. Reverse Proxy phase

   In this phase, after the user's session has been validated, the SPS uses one of its defined services (forward, redirect, or another service) to handle the user's request. The action of the SPS in this phase is determined by the proxy rules contained in the proxy rules XML configuration file.

   **Note:** For URL rewriting session schemes, content is forwarded to the rewriting mechanism in this phase before being sent back to the user.

## Default Session Scheme

The default session scheme is the scheme that the SPS uses to establish and maintain user sessions when no other scheme is specified for a user agent type. The <SessionScheme> element contains the name attribute, which is used to identify the session scheme when assigning schemes to user agent types. The server.conf file must contain a default session scheme configuration.

You can configure the default session scheme to use any available session scheme.

The default session scheme section has the following format:

```
#Session Schemes
<SessionScheme name="default">
    class="com.netegrity.proxy.session.SessionCookieScheme"
    accepts_smsession_cookies="true"
</SessionScheme>
```

The <SessionScheme> element has the following directives:

**class**

Indicates the Java class that contains the default session scheme.

**Default:** com.netegrity.proxy.session.SSLIdSessionScheme

**accepts_smsession_cookies**

Indicates that if a user agent type is associated with the SiteMinder cookies session scheme, users that access resources via that user agent type will maintain session using traditional SiteMinder cookies.

SiteMinder uses cookies to track sessions so a cookies scheme is supported by the SPS. Indicates if SMSESSION cookies are accepted.

Specify one of the following values:

**true**

Indicates that SMSESSION cookies are accepted and used by the session scheme.

**false**

Indicates that SMSESSION cookies are not supported by the session scheme.

## Specifying the Default Session Scheme

The default session scheme is used when no other session scheme is specified for a user agent type.

Default Session Scheme directives are as follows:

**defaultsessionscheme**

Specifies a session scheme other than SiteMinder cookie session scheme as the default scheme. You can modify this entry to include any of your session schemes as the default session scheme.

**Default:** default

**enablewritecookiepath**

Instructs the SPS to rewrite the cookie path from the URI set by the server sitting behind the proxy to the URI of the initial request.

**Default:** no

**enablewritecookiedomain**

Instructs the SPS to rewrite the cookie domain from the domain set by the server sitting behind the proxy to the domain of the initial request.

**Default:** no

## SSL ID Session Scheme

A secure sockets layer (SSL) connection includes a unique identifier that is created when an SSL connection is initiated. The SPS can use this unique ID as a token to refer to the session information for a user which is maintained in the SPS in-memory session store. This scheme eliminates cookies as a mechanism for maintaining a user's session.

A limitation of the SSL ID session scheme is that the initial contact with the SPS establishes an SSL session ID. If a user's SSL session is interrupted, and a new SSL connection is established, the user must be re-authenticated and re-authorized, since the new SSL connection has a connection to a new server, even though it is a virtual server on the same system. This also means that forms used by HTML Forms Authentication Schemes must be served from the same host name as the protected resource.

### SSL ID Session Scheme Configuration

The SSL ID section lists the session scheme using the SSL ID.

SSL ID session schemes can be supported without any custom work using the Java classes that are packaged with SPS.

**Important!** To use the SSL ID authentication scheme, you also have to enable a setting in the Apache Web server's httpd.conf file.

The SSL ID session scheme has the following format:

```
<SessionScheme name="ssl_id">
    class="com.netegrity.proxy.session.SSLIdSessionScheme"
    accepts_smsession_cookies="false"
</SessionScheme>
```

The directives for the ssl_id are as follows:

**class**

Specifies the Java class that handles SSL ID session schemes.

**Default:** com.netegrity.proxy.session.SSLIdSessionScheme

**accepts_smsession_cookies**

Indicates if SMSESSION cookies are accepted. Specify one of the following values:

**true**

Indicates that SMSESSION cookies are accepted and used by the session scheme.

**false**

Indicates that SMSESSION cookies are not supported by the session scheme.

### Modifying the httpd.conf File for the SSL ID Scheme

In addition to configuring the SSL ID session scheme in the server.conf file, you have to modify the Apache Web server httpd.conf file to enable SSL.

**To modify the httpd.conf file for the SSL ID scheme**

1. Open the httpd.conf file located in the directory *sps_home*/secure-proxy/httpd/conf.
2. Locate the line in the file that reads:

   #SSLOptions +StdEnvVars +ExportCertData +CompatEnvVars
3. Delete the # symbol from the beginning of the line.

   **Note**: For SPS r6.0 SP 3 and later, also remove +CompateEnvVars so that the line reads as follows:

   SSLOptions +StdEnvVars +ExportCertData
4. Save the httpd.conf file.
5. Restart the SPS.

## IP Address Session Scheme

In environments where IP addresses are fixed, the SPS can use an IP address to refer to a user's session information in the session store. This scheme eliminates cookies, but may only be used in environments where a user is assigned a fixed IP address.

## Mini-cookies Session Scheme

One of the disadvantages of a traditional SiteMinder cookie-based session scheme is the size of the cookies. When the amount of data transferred with each request increases, the cost of access for certain types of devices such as wireless phones increases.

A mini-cookie is a small cookie, approximately 10 bytes in size that contains a token which can be used to reference session information in SiteMinder in-memory session store. The mini-cookie is a fraction of the size of a standard SiteMinder cookie, and provides an alternative for standard SiteMinder cookies.

## Mini-cookie Session Scheme Configuration

The mini-cookies session scheme stores session information in the SPS in-memory session store and creates a cookie that contains an encrypted token that the SPS returns to the user.

This section has the following format:

```
<SessionScheme name="minicookie">
    class="com.netegrity.proxy.session.MiniCookieSessionScheme"
    accepts_smsession_cookies="false"
    # The name of the small cookie to be stored in the client.
    cookie_name="SMID"
</SessionScheme>
```

The directives in the mini-cookies session scheme are listed following.

**class**

Specifies the java class that defines the session scheme. This directive is not modified when you want to use the mini-cookies session scheme provided with the SPS.

**Default:** com.netegrity.proxy.session.MiniCookieSessionScheme

**accepts_smsession_cookies**

Indicates if SMSESSION cookies are accepted. Specify one of the following values:

**true**

Indicates that SMSESSION cookies are accepted and used by the session scheme.

**false**

Indicates that the SMSESSION cookies are not supported by the session scheme. Use this setting to verify that only a mini-cookie session is used for the session scheme.

**cookie_name**

Indicates the name of the mini-cookie that contains the token for the user session.

**Note:** This name is not configured using the same value for all SPS that provides single sign-on.

## Simple URL Rewriting Session Scheme

Simple URL rewriting is a method for tracking a user session by appending a token to the requested URL. This token is used to retrieve session information from the in-memory session store.

## Simple URL Rewriting Configuration

The simple_url schemes support simple URL rewriting, which can be accomplished without any custom work.

**Note:** The CGI-based and FCC-based password schemes are supported with the simple_url session scheme.

### Example

A user accesses a host and the user session is established through the simple URL rewriting session scheme. An initial request can look like the following example:

http://banking.company.com/index.html

If the user provides appropriate credentials and is authenticated and authorized, the URL requested by the user is rewritten and returned to the user in a form similar to the following:

http://banking.company.com/SMID=*nnnnnnnnnn*/index.html

**nnnnnnnnnn**

Represents a hashed, randomly generated token that the SPS uses to identify the user session.

**Important!** For the simple URL rewriting session scheme to work, any links defined in the enterprise must be relative links. If links are absolute, the simple URL rewriting scheme fails. Also, the token that the SPS appends to a URL is stripped from the URL when the request is forward. The token is only appended at the SPS interaction level so that it does not interfere with back-end server processing.

The format of the SimpleURL scheme is:

```
<SessionScheme name="simple_url">
    class="com.netegrity.proxy.session.SimpleURLSessionScheme"
    accepts_smsession_cookies="false"
    session_key_name="SMID"
</SessionScheme>
```

The directives in the SimpleURL scheme are listed following.

**class**

Specifies the Java class that defines the session scheme. This directive is not modified when you want to use the cookieless rewriting session scheme.

**Default:** com.netegrity.proxy.session.SimpleURLSessionScheme

**accepts_smsession_cookies**

Indicates whether SMSESSION cookies are accepted. Specify one of the following values:

**true**

Indicates that SMSESSION cookies are accepted and used by the session scheme.

**false**

Indicates that SMSESSION cookies are not supported by the session scheme. Use this setting to verify that only a cookieless rewriting session is used for this session scheme.

**session_key_name**

Specifies the SiteMinder ID (SMID) session identifier.

**Note:** When a cookieless federation transaction is being processed by the SPS federation gateway and the simple_url session scheme is used, the SMID is added to the request as a query parameter instead of the being appended to the URI.

## Enable Cookieless Federation for Rewriteable Session Schemes

For the SPS to use rewritable session schemes, such as simple URL session scheme, in a federated environment, configure cookieless federation.

**To configure cookieless federation**

1. Open the server.conf file in a text editor. This file is located in the directory *sps_home/*secure-proxy/proxy-engine/conf
2. Add the following code to the virtual host section for the virtual host that is serving FWS.

   cookielessfederation="yes"
3. Save the file.
4. Restart the SPS.

**Note:** No separate post filter, such as the CookielessFedFilter needs to be enabled for the SPS federation gateway. This functionality is provided out-of-the-box when you enable the federation gateway functionality. You have to enable this post filter when the SPS is not acting as a federation gateway.

### Rewrite FWS Redirects for Simple URL Session Schemes

If you deploy the SPS in a federated environment, one of the session schemes you can at the site producing assertions is a simple URL session scheme. If you use this scheme, you may be required to rewrite the links that direct the user to the appropriate site so that the session key is added to the link. In SiteMinder documentation, these links for SAML 1.x are called intersite transfer URLs. For SAML 2.0, these links are referred to as an unsolicited response or an AuthnRequest link.

For rewriting the links so that the session key information is added to the base of the URLs, a sample post filter, RewriteLinksPostFilter, is provided along with the SPS filter examples. This filter can be compiled and be attached to the appropriate proxy rule, which handles the forwards to the intersite transfer URL, unsolicited response, or AuthnRequest.

The RewriteLinksPostFilter provided with the SPS is a sample filter. You must configure the filter to suit your requirements.

**Note:** If you use the simple_url session scheme for transactions involving the SPS federation gateway, the session key (SMID) gets added to the request as a query parameter instead of being appended to the URI. However, the SMID gets added to the URI when the final target resource is accessed at the back-end server.

## Wireless Device ID Session Scheme

Some wireless devices have a unique device identification number. This number is sent as a header variable with any requests for resources. The SPS can use this device ID as a token to refer to session information in the session store.

Because device IDs differ by wireless device vendor, define a device ID session scheme in the server.conf file. This scheme must include the class information and a device_id_header_name directive that is set to the vendor-specific device ID.

The format of the device ID scheme is:

```
<SessionScheme name="device_id">
    class="com.netegrity.proxy.session.DeviceIdSessionScheme"
    accepts_smsession_cookies="false"
    device_id_header_name="vendor_device_id_header_name"
</SessionScheme>
```

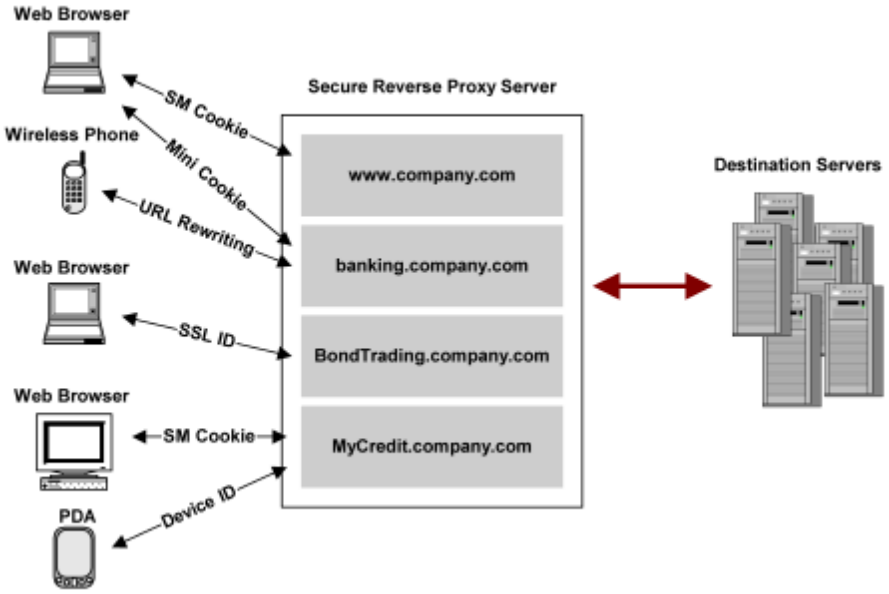
## Uses for Each Session Scheme

The following table illustrates the scenarios in which each session scheme is used. The session schemes are based on the sensitivity of resources on a virtual host.

| Session Scheme | Security Level | Recommendation |
|---|---|---|
| SSL Session ID | High | This scheme provides a clean and highly secure means of holding user sessions. Although the most secure of the available schemes, it is limited in scalability. All content must be served over SSL and the user must continue to access the same SPS server for the session to persist. Also, some browsers (some versions of IE), by default, terminate the SSL session after 2 minutes. This scheme is ideal for intranet and extranet applications with high security needs. |
| SiteMinder Cookies | Medium or High | This scheme is the traditional SiteMinder session mechanism, which has proven highly secure in many enterprise deployments.<br>For maximum security, the WebAgent SecureCookie setting are set to "Yes". |
| IP Address | Low | This scheme is only for applications where users are retrieving information (with HTTP GET) from protected resources and not sending (with HTTP POST) information to a secure application. An example of an appropriate application would be an online library. An example of an in-appropriate application would be a bond trading application. |
| Mini-Cookies | Medium or High | This scheme is ideal for applications where user clients accept cookies but are accessing the application over connections of limited speed and bandwidth.<br>For maximum security, the WebAgent SecureCookie setting is set to "Yes". |
| Simple URL Rewriting | Medium | This scheme is ideal for environments that do not support or want to use cookies. |
| Device ID | Medium | This scheme is designed for wireless environments where a device ID is sent with every client request to identify a user. |

## Multiple Session Schemes for Virtual Hosts

The SPS supports multiple session schemes for each virtual host in an enterprise. Session schemes can be assigned to each user agent type that has access to a virtual host. The following illustration shows the SPS configured for four virtual hosts.




The preceding illustration shows that the session scheme varies for a user agent depending on the virtual host accessed by the user. For example, when accessing www.company.com with a browser, the SPS uses SiteMinder cookies to maintain a user session. But when accessing BondTrading.company.com, browser sessions are maintained using the SSL ID of the user HTTPS connection. The sessions for PDA and wireless users are maintained using cookieless session schemes, while browser user sessions are maintained through cookies or mini-cookies.

## Deleting Attribute Cookies for Cookieless Federation

To support environments that do not want to exchange cookies, the SPS maps the SiteMinder session cookies to a cookieless session scheme and deletes the cookies from the response. However, the attribute cookies are not mapped and remain in the response.

The FWS application processes the federation requests and inserts attribute cookies and SiteMinder-created session cookies in its responses. The responses are forwarded to the SPS.

The SPS can be configured to delete the attribute cookies inserted by the FWS application.

**To configure the SPS to delete session and attribute cookies**

1. Open server.conf file in a text editor.

   The file is located in the directory *sps_home*/secure-proxy/proxy-engine/conf.

2. Add the following code to the virtual host section for the virtual host that is serving the FWS.

   deleteallcookiesforfed="yes"

3. Save the file.
4. Restart the SPS.

# User Agent Settings in the Server.conf

User agents define the device types such as web browsers, wireless phones, and PDAs by which users can access network resources. All user agents must be defined in <UserAgent> elements. Each <UserAgent> element includes a name attribute that identifies the user agent type. By default no user agent types are predefined in the server.conf file.

The user agent configuration section has the following format:

```
#<UserAgent name="user_agent_name_1">
#  header_name_1=some regular expression
# </UserAgent>
```

The directive in the UserAgent section is:

**header_name**

This directive contains the user-agent header of an HTTP request. This header indicates the type of device making the request. You can use regular expressions and provide a partial name as part of the expression. This allows you to specify user agent types whose user-agent header may contain slight differences, such as version numbers.

Device types must be defined in a <UserAgent> element before they can be associated with a session scheme in a <SessionSchemeMapping> element.

## Nokia User Agent Settings

You can specify a Nokia user agent type, which is for Nokia wireless phones. The name attribute for the <UserAgent> section is the name used to identify the user agent type when specifying session scheme mappings.

The Nokia user agent entry format would be as follows:

```
# Nokia
<UserAgent name="Nokia">
    User-Agent="Nokia."
        attribute_name="value"
</UserAgent>
```

The directives for the Nokia user agent are as follows:

**User-Agent**

This directive contains the contents of the user agent header of an HTTP request. This header indicates the type of device making the request. You can use regular expressions and provide a partial name as part of the expression. This directive allows you to specify user agent types whose User-Agent header can possibly contain slight differences, such as version numbers.

**Default:** Nokia

**attribute_name**

The sections of the server.conf for wireless devices and other user agent types can possibly contain additional attributes and values for those attributes. Attributes are not required, but can possibly be desirable for some applications.

# Virtual Host Settings in the server.conf File

The <VirtualHostDefaults> element of the server.conf file specifies default settings for virtual hosts. These settings are used for each virtual host you add to the SPS.

To specify non-default values for a virtual host, add a <VirtualHost> element following the <VirtualHostDefaults> element. The <VirtualHost> element must contain directives and values that are different from the default virtual host.

The default virtual host settings are broken up into the following sections:

- Default session scheme
- Session scheme mappings
- WebAgent.conf settings
- Default virtual host name

The format of the <VirtualHostDefaults> section is as follows:

```
<VirtualHostDefaults>
    # default session scheme
    defaultsessionscheme="default"
    enablerewritecookiepath="no"
    enablerewritecookiedomain="no"
    enableproxypreservehost="no"

    # specify the block size for request and response in KBs
    requestblocksize="4"
    responseblocksize="4"

    #TO-DO: Define any session scheme mappings
    #<SessionSchemeMappings>
    #    user_agent_name=session_scheme_name
    #</SessionSchemeMappings>



    # Web Agent.conf
    <WebAgent>
    sminitfile="C:\Program Files\netegrity\secure-proxy\proxy-engine\
    conf\defaultagent\WebAgent.conf"
    </WebAgent>
</VirtualHostDefaults>
```

## Setting Virtual Host Cookie Path and Domain to the Correct URI

The virtual host configuration in the server.conf file includes the enablerewritecookiepath and enablerewritecookiedomain parameters, which you can use to manage cookies generated by a destination server that sits behind the SPS. When the SPS receives a request from a client, the SPS authenticates the user and directs the client to the requested destination server. The destination server generates a cookie that it places in the browser, then the SPS sends the user back the client response with the cookie. After receiving the response from the SPS, the client stores the cookie.

When the client sends a subsequent request, the browser retrieves the stored cookie associated with the URL. In some cases, the destination server can possibly have set the cookie path to its own resource URI and not to the URI of the initial request. As a result, when the client sends the subsequent request, the browser contains the wrong cookie or does not even have a cookie. The request is received at the destination server with the wrong cookie or with no cookie at all.

To help ensure that the correct cookie is set in the browser, you can configure the SPS to rewrite the cookie path and cookie domain. The destination server sets the cookie path and cookie domain to the URI of the resource on the SPS server. The client can send the correct cookie back with subsequent requests to SPS.

The two parameters operate as follows:

**enablerewritecookiepath**

Instructs the SPS to rewrite the cookie path to the URI of the initial request from the URI set by the server that sits behind the proxy.

**Default:** no

**enablerewritecookiedomain**

Instructs the SPS to rewrite the cookie domain from the domain set by the server sitting behind the proxy to the domain of the initial request.

**Default:** no

### Example

The client requests an SPS resource http://mysps.ca.com/basic/test/page0.html. With the enablerewritecookiepath set to yes, the cookie path is rewritten to /basic/test before the browser is sent back to the client. This cookie is rewritten regardless of the cookie path that was originally in the cookie received by SPS from the destination server.

**To rewrite backend cookie paths and domain**

1. Open the server.conf file in a text editor.
2. Set one or both of the following parameters to yes:
   - enablerewritecookiepath
   - enablerewritecookiedomain
3. Save the file.
4. Restart the SPS.

## Preserve the HOST Header File

You can preserve the HTTP HOST header file and instruct the SPS to send it to the backend by setting the value of the enableproxypreservehost parameter to YES. This parameter is located in the <VirtualHostDefaults> section of the server.conf file. Its default value is NO.

## Handling Large Files Using Data Blocks

The SPS handles the transfer of large files to users by breaking up the data transferred between the SPS and the backend server into blocks. You control the block size read by the SPS using two parameters in the Virtual Host section of the server.conf file:

- requestblocksize
- responseblocksize

When a user sends a file to a backend server, the SPS verifies the requestblocksize specified for that virtual host. Based on the value of requestblocksize, the SPS breaks down the data into blocks and then forwards the blocks to the backend server.

Similarly, when the backend server sends data to the user, the SPS verifies the responseblocksize specified for that virtual host. Based on the value of responseblocksize, the SPS reads the data in blocks from the backend server before further processing the blocks. This enables the user to control the number of read-write operations for such file transfers. To handle large file transfers, use large block sizes.

**Note:** The requestblocksize and responseblocksize parameters should be defined in proportion to the available and allocated JVM heap size for the SPS java process.

## Define File Data Block Size for Large File Handling

To define the block size to handle large files when you configure the block size for a virtual host, modify the request and response block sizes for each virtual host. These parameters are valid only for that virtual host. The data block sizes can be different for different virtual hosts, but the settings only apply to the associated virtual host you are configuring.

**Follow these steps:**

1. Open the server.conf file in a text editor.
2. Edit the following parameters under the virtual host configuration:

   **requestblocksize**

   Defines the block size of the request data that will be read at a time before the data blocks are sent to the backend server. The block sizes are in KB.

   **Limits:** 1KB to approximately 352000 KB. For any value greater than or equal to 8 KB, chunks of 8 KB are created. A corresponding chunk size is create for values between 1 KB and 8 KB.

   **Default:** 4

   **responseblocksize**

   Defines the block size of the response data that will be read at a time before the data blocks are forwarded from the backend server to the user. The block sizes are in KB.

   **Limits**: 1KB to approximately 352000 KB.

   **Default:** 8
3. Save the server.conf file.
4. Restart the SPS.

## Adjust JVM Heap Size for Data Blocks

The requestblocksize and responseblocksize parameters are defined in proportion to the available and allocated JVM heap size for the SPS java process.

**To define SPS JVM heap size**

1. Navigate to the appropriate directory:
   - *Windows: sps_home*/secure-proxy/proxy-engine/conf
   - UNIX: sps_home/secure-proxy/proxy-engine
2. Open one of the following files
   - For Windows systems: SmSpsProxyEngine.properties file
   - For UNIX systems: proxyserver.sh file

3. Add the following parameters in the Java section of the file:
   - –Xms256m
   - –Xmx512m
4. Save the file.

## Session Scheme Mapping for the Default Virtual Host

Session scheme mappings associate session schemes with user agent types. The user agent types defined in <UserAgent> elements of the server.conf file must be mapped to session schemes defined in the <SessionScheme> elements.

The format of the session scheme mapping associated with user agents is as follows:

```
#<SessionSchemeMappings>
    #     user_agent_name=session_scheme_name
#</SessionSchemeMappings>
```

The directive in this section is:

**user_agent_name**

Associates a user agent with a session scheme. To set the values:

**user_agent_name**

Specifies a name defined in a <UserAgent> section of the file

**session_scheme_name**

Specifies a scheme defined in a SessionScheme element.

**Example:**

User Agents named browser, phone1, and phone2 have been defined and mapped to any of the session schemes defined in the file. For this example, browser is mapped to the default session scheme, phone1 is mapped to the simple_url scheme, and phone2 is mapped to the minicookie session scheme.

The resulting <SessionSchemeMappings> element appears as follows:

```
# Session Scheme Maps
<SessionSchemeMappings>
  browser="default"
  phone1="simple_url"
  phone2="minicookie"
</SessionSchemeMappings>
```

## Web Agent Settings for the Default Virtual Host

The server.conf file includes a <WebAgent> section for the <VirtuallHostDefaults>. The sminitfile directive specifies the configuration file, WebAgent.conf for the default web agent. If local configuration is allowed, the WebAgent.conf file points to a local configuration file, LocalConfig.config.

If you create more than one virtual host, you can use the default Web Agent when you do not intend to use alternate settings in the Web Agent configuration file. If you plan to set any directive differently, for example, to specify a different log level, use a different Web Agent for the new virtual host.

**To configure a new Web Agent for a new virtual host**

1. Create a directory with the name of the new virtual host, for example, serverb.
2. Copy the contents of the directory for the default virtual host into the new directory.

   Run smreghost if the new Web Agent points to a different SiteMinder installation.
   **Note:** If the Web Agent configuration objects for both virtual hosts point to the same SiteMinder installation, you do not need to run smreghost. You can use the same smhost file for both the Web Agents.
3. Use a text editor to modify WebAgent.conf to reflect the new agent configuration object. Verify that the Web Agents have different log files.
4. Open the WebAgent.conf file and add the following required directive with a unique value.

   ServerPath="*path*"

   **path**

   Specifies is the fully qualified path to the WebAgent.conf file you are editing

   - For Windows, this value must be a unique alphanumeric string. The backslash '\' character is not permitted in this string.
   - For UNIX, this value must be the fully qualified path to the WebAgent.conf file you are editing.
5. Access the Agent Configuration Object at the Policy Server that corresponds to the first host configuration object in the server.conf file. Verify the Agent cache settings for MaxResoureceCacheSize and MaxSessionCacheSize and also that the cache limits take into account all Agent Configuration Objects.

**Note:** For detailed information about the Web Agent settings, see the *CA SiteMinder Web Agent Guide*.

### Setting the requirecookies Parameter

The requirecookies setting in the server.conf file is a special Web Agent setting that is useful only if basic authentication was set during the Policy Server configuration. This setting instructs the agent to require either an SMSESSION or an SMCHALLENGE cookie to process HTTP requests successfully, including basic Authorization headers.

If you configure the embedded Web Agent to require cookies, the browser must accept HTTP cookies. If the browser does not, the user receives an error message from the Agent denying them access to all protected resources.

Set the requirecookies setting to yes when all user agent types for the associated virtual server use the default session scheme. If an agent type uses a cookieless session scheme, set the requirecookies parameter to no.

## Handling Redirects by Destination Servers

Some destination servers can respond to a request from the SPS with a redirection.

**Note:** A redirection that is the result of a request to the SPS is not the same as a redirect that occurs in a proxy rule. For information about a redirect in a proxy rule, see nete:redirect.

Because the redirection initiated by the destination server is likely to a server behind the DMZ, the URL specified in the redirection results in an error. However, you can include parameters in a virtual host configuration that substitute the virtual host server name and port number in place of a redirect from a destination server.

To substitute the virtual host server and port for redirect writing, configure the following:

**enableredirectrewrite**

Enables or disables redirect rewriting. If this directive is set to a value of yes, the URL for a redirect initiated by a destination server is examined by the SPS. If the redirect URL contains a string found in the list of strings specified in the associated redirectrewritablehostnames parameter, the server name and port number of the redirect are replaced by the server name and port number of the virtual host.

If the parameter is set to a value of no, any redirects initiated by destination servers are passed back to the requesting user.

**redirectrewritablehostnames**

Contains a comma-separated list of strings that the SPS searches for when a redirection is initiated by a destination server. If any of the specified strings are found in the server or port portion of the redirect URL, the SPS substitutes the name and port number of the virtual host for the server name and port portion of the redirect URL.

If you specify a value of "ALL" for this parameter, the SPS substitutes the server name and port number of the virtual host for all redirects initiated by the destination server.

For example, consider a virtual host configuration in the server.conf file that contains the following parameters:

```
<VirtualHost name="sales">
    hostnames="sales, sales.company.com"
    enableredirectrewrite="yes"
    redirectrewritablehostnames="server1.company.com,domain1.com"
</VirtualHost>
```

When a user makes a request from http://sales.company.com:80, the SPS forwards the request to a destination server according to proxy rules. If the destination server responds with a redirect to server1.internal.company.com, the redirect is rewritten before being passed to the user as sales.company.com:80.

**Note:** The proxy rules for the SPS must be configured to handle the redirected requests.

## Virtual Host Names Configuration

For the SPS to act as a virtual host for one or more host names, configure a <VirtualHost> element for related hostnames and IP addresses. Each server.conf file must contain one <VirtualHost> element for the default virtual host, in addition to additional elements for hostnames at other IP addresses.

The following is an example of a <VirtualHost> element for the default virtual host in a server.conf file.

```
# Default Virtual Host

<VirtualHost name="default">
hostnames="home, home.company.com"
addresses="123.123.12.12"
</VirtualHost>
```

The default virtual host in the preceding example is the host named home.company.com residing at IP address 123.123.12.12. You can add hostnames that resolve as the default virtual host by adding to the comma-separated list of values in the hostnames. To add additional virtual hosts to the proxy configuration, add additional <VirtualHost> elements that include hostname directives for the additional virtual hosts.

**Example:**

To add a sales virtual host for the server, sales.company.com virtual host, add the following element:

```
<VirtualHost name="sales">
hostnames="sales, sales.company.com"
</VirtualHost>
```

## Override Default Values for Virtual Hosts

The <VirtualHostDefaults> settings are used for all virtual hosts defined in the server.conf file unless you explicitly enter settings for a particular virtual host.

You do not have to reconfigure all the virtual settings for the single virtual host. Any settings that you do not redefined in the <VirtualHost> element are applied from the <VirtualHostDefaults> settings.

**To override virtual host default values**

1. Add any directive in the default virtual host configuration to the <VirtualHost> element that you want to modify.
2. Specify a new value for the directive in the <VirtualHost> element.
3. Save the file.
4. Restart the SPS.

**Example**

The virtual host named "sales" requires a default session scheme from what is configured for the default virtual host. The <VirtualHost> element could be modified as follows:

```
<VirtualHost name="sales">
    hostnames="sales, sales.company.com"
    addresses="123.123.22.22"
    defaultsessionscheme="minicookie"
</VirtualHost>
```

# Chapter 8: Configuring the SPS Log Settings

## SPS logger.properties File Overview

The SPS log settings are configured through the logger.properties file. These settings in the file are groups of name/value pairs or directives that the SPS reads at run time. You can update the logger.properties file without restarting the SPS.

The following is the default location of the logger.properties file:

*sps_home*/Tomcat/properties

## Modifying the logger.properties File

The log settings for the SPS are maintained in the logger.properties file located in the following directory:

*sps_home*/Tomcat/properties

**Follow these steps:**

1. Open the file in a text editor.
2. Edit the directives, as necessary.
3. Save the file.

The log settings are changed.

# Logging Settings

The logger.properties file contents are grouped into the following sections:

- SvrConsoleAppender settings
- SvrFileAppender settings
- Server log settings
- Server log rolling settings

The directives contained in this file follow the format name=value. Any lines beginning with the # symbol are comments, and are not read when the SPS loads configuration settings.

**Note**: Pathnames on Windows systems use double backslashes (\\).

## SvrConsoleAppender Settings

The SvrConsoleAppender Settings section contains settings for logging events on to a console. This section has the following format:

```
# SvrConsoleAppender is set to be a ConsoleAppender.
log4j.appender.SvrConsoleAppender=org.apache.log4j.ConsoleAppender
log4j.appender.SvrConsoleAppender.layout=org.apache.log4j.PatternLayout
log4j.appender.SvrConsoleAppender.layout.ConversionPattern=<log_message_display_format_on_console>
```

***log_message_display_format_on_console***

Specifies the display format of a log message on the console. The SPS supports all the log4j date pattern strings.

**Default Value**: [%d{dd/MMM/yyyy:HH:mm:ss-SSS}] [%p] - %m%n

## SvrFileAppender Settings

The SvrFileAppender Settings section contains settings for logging events in a file. This section has the following format:

```
# SvrFileAppender is set to be a FileAppender.
log4j.appender.SvrFileAppender=org.apache.log4j.FileAppender
log4j.appender.SvrFileAppender.layout=org.apache.log4j.PatternLayout
log4j.appender.SvrFileAppender.layout.ConversionPattern=<log_message_display_form
in_file>
```

***log_message_display_format_in_file***

Specifies the display format of a log message in the file. The SPS supports all the log4j date pattern strings.

**Default Value**: [%d{dd/MMM/yyyy:HH:mm:ss-SSS}] [%p] - %m%n

## Log Settings

The server log settings section contains settings for enabling and disabling logging, setting logging level, and setting the output format of the log messages. This section has the following format:

```
# Server.conf settings:
# details of setting "log4j.rootCategory":
# For First attribute:
# Depending on the logging level needed, set the appropriate level
# Possible values : OFF, FATAL, ERROR, WARN, INFO, DEBUG, ALL
# For Second attribute:
# if you want to enable log console, then add SvrConsoleAppender, else don't  add this.
# For Third attribute:
# if you want to enable logging into file, theb add SvrFileAppender, else don't add
this.
log4j.rootCategory=<log_level>,<output_format>
```

***log level***

Specifies the log level of a message. The following list displays the possible values in the increasing order of priority:

- OFF
- FATAL
- ERROR
- WARN
- INFO
- DEBUG
- ALL

If the value is set to OFF, logging is disabled. If the value is set to any other value, logging is enabled.

**Default**: INFO

***output format***

Specifies how a log message is displayed. You can display a log message on a console, or store it in a file, or both.

**Default**: SvrFileAppender

For example, if the log level is INFO and you want to display a log message on a console and store it in a file, use the following command:

```
log4j.rootCategory=INFO,SvrConsoleAppender,SvrFileAppender
```

## Log Rolling Settings

The server log rolling Settings section contains settings for enabling log rolling. You can enable log rolling based on *one* of the following mechanisms:

- Log rolling based on file size
- Log rolling based on file age

This section has the following format:

```
# Enable the below setting only if file logging is enabled above. if not make it as an comment by adding "#" at the begging of the line.
log4j.appender.SvrFileAppender.File=<logfile_path>
# Enable this only if file logging is enabled above.
# set vale to "true" if messages are to be appended to the existing file. else set to "false"
log4j.appender.SvrFileAppender.Append=true|false
#Configurations to rollover server log file based on file size
log4j.appender.SvrFileAppender=org.apache.log4j.RollingFileAppender
log4j.appender.SvrFileAppender.MaxFileSize=<maximum_logfile_size>
log4j.appender.SvrFileAppender.MaxBackupIndex=<maximum_number_of_logfile>
```

The Log Rolling Based on File Size section contains the following settings for enabling log rolling based on a file size:

***logfile path***

Specifies the name and path of the log file.

**Default Name**: server.log

**Default Path**: <*sps_home*>/secure-proxy/proxy-engine/logs/

**true|false**

Specifies how the SPS manages the log file. If this value is set to true, the SPS appends new log messages to the existing log file when it starts. If this value is set to false, the SPS rolls over the existing log file and creates a log file for new log messages when it starts.

**Default Value**: true

***MaxFileSize***

Specifies the maximum size of the log file after which the SPS must create a new log file.

**Default Value**: 1 MB

***MaxBackupIndex***

Specifies the maximum number of log files that the SPS creates. If the number of log files exceeds the maximum number that is specified , the SPS deletes the oldest log file and creates a new log file.

**Default Value**: 10

The Log Rolling Based on File Age section contains the following settings for enabling log rolling based on a file age:

***date_pattern***

Specifies the date when the SPS must create a new log file.

**Default**: yyyy-MM-dd

The SPS creates a new logfile in the following format:

*<logfile_name>.<date_format>*

***logfile_name***

Specifies the name of the log file.

**Default**: server.log

***date_format***

Specifies the date when the SPS created a logfile. The SPS supports all the log4j date pattern strings.

**Default**: yyyy-MM-dd

# Modify ServerPath in WebAgent.conf for Logging

If you configure a Web Agent for virtual hosts, each host must have its own Web Agent cache, log file, and health monitoring resources. To help ensure that resources are unique, configure the ServerPath parameter.

The ServerPath parameter specifies a unique identifier for the Web Agent resources of cache, logging, and health monitoring. For each server instance to have its own set of these Agent resources, the value of the ServerPath parameter must be unique.

For example, you can set the ServerPath parameter to the directory where the web server log file is stored, such as *server_instance_root*/logs.

If you have virtual hosts in your environment, verify that the ServerPath parameter is in each WebAgent.conf file.

**To verify that the ServerPath parameter is in each WebAgent.conf file**

1. Navigate to the WebAgent.conf file in the directory *sps_home*\secure-proxy\proxy-engine\conf\defaultagent.
2. Open the file.
3. Check that the ServerPath setting is configured to a unique string or path.

   For Windows, you can specify any unique string. For UNIX, specify a unique system path.
4. Save the WebAgent.conf file.

# Chapter 9: Configuring Proxy Rules

This section contains the following topics:

Proxy Rules Overview (see page 141)
Establish a Proxy Rules Configuration File (see page 144)
Proxy Rules DTD (see page 145)
How nete:xprcond Elements Works (see page 156)
Regular Expression Syntax (see page 156)
Header Values in Forwards, Redirects, and Results Filters (see page 160)
Response Handling (see page 162)
Modify Proxy Rules (see page 162)
Sample Proxy Rules Configuration Files (see page 162)

## Proxy Rules Overview

The primary purpose of the SPS is to route requests to the appropriate destination servers in the enterprise. The SPS routes requests using the proxy engine, which is built into the server. The proxy engine interprets proxy rules and provides both a forward and a redirect service to handle the disposition of all user requests for back end resources.

Proxy rules define the details of how the SPS forwards or redirects requests to resources located on destination servers within the enterprise. A set of proxy rules is defined in an XML configuration file according to the proxy rules DTD, which is installed with the SPS.

**Note:** The proxyrules.xml file contains a default rule that forwards requests to http://www.ca.com$0, where $0 appends the entire URI from the original request to the destination, which is www.ca.com in this case. You must modify this rule to suit your environment.

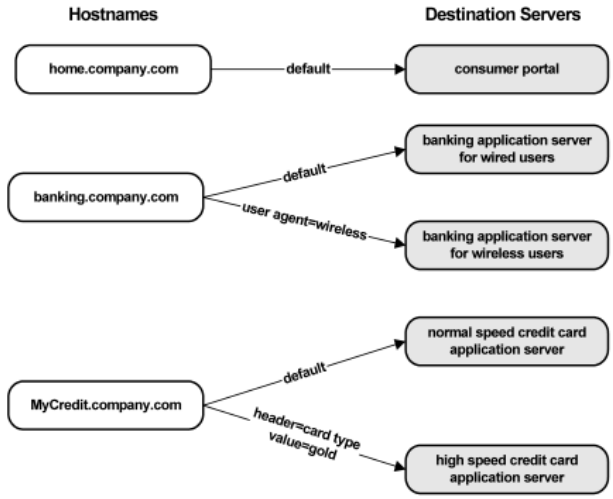
**More information:**

Proxy Rules DTD (see page 145)

### Planning Routes for Incoming Requests

Before you set up the proxyrules.xml file, you should map out the routing for incoming requests. Depending on the virtual host that contains the requested resource, the proxy rules can use a default destination; forward a request based on the user agent type, or uses a HTTP header value to determine the destination server that will fulfill the request. The SPS can provide access to a number of virtual hosts.

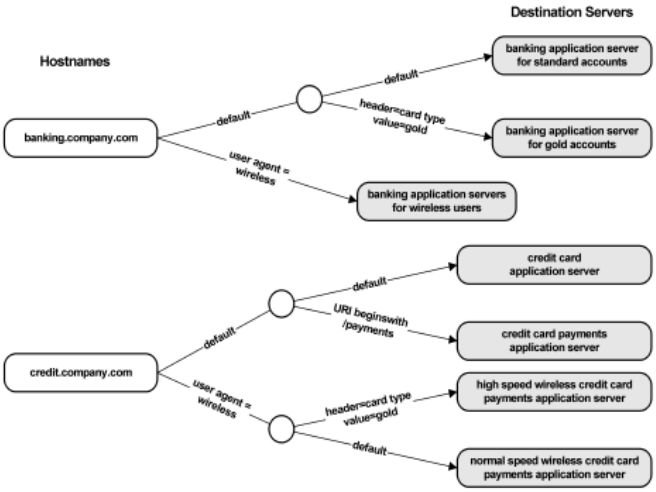
The following illustration shows how proxy rules can be used to route requests to appropriate destination servers.




**Note:** A proxy rules configuration file is processed from top to bottom by the SPS. The first condition that is met by an incoming request determines the behavior of the proxy engine. For example, if a set of proxy rules has two conditions based on a string contained in the URI of a request, and part of the URI of an incoming request matches both of the strings, the first condition listed in the proxy rules file will be used to route the request.

Proxy rules can also be used to control more complex conditions for directing requests to destination servers.

The following illustration shows how proxy rules can be used with a second level of conditions nested within parent conditions.



## Proxy Rules Terminology

The proxy rules configuration file is a description of the XML elements that the SPS uses when routing user requests. The following terms are used to describe the proxy rules:

**Destinations**

A destination is a URL that fulfills a request. The SPS forwards a request to a destination, or sends a redirect response to a user that specifies a destination. A set of proxy rules must contain destinations that can be reached according to the conditions and cases defined in the proxy rules.

**Conditions**

A condition is an attribute of a request that allows the SPS to determine the destination of a request. Each condition may have many cases that the SPS evaluates to direct a request to the appropriate destination. Each condition must contain a default element that defines the behavior if a request does not match any of the cases defined in the condition.

The conditions may include any *one* or *more* of the following:

**URI**

The SPS uses the portion of the requested URL after the hostname to determine how to route a request. Using criteria described in the DTD, a portion of a URI, such as the file extension of the requested resource, can be used to route requests.

**Query String**

The SPS uses the query string portion of a URI to determine how to route a request. The query string includes all the characters following a '?' in the request.

**Host**

The SPS uses the requested server hostname to determine how to route a request. The port number of the hostname can also be used as criteria for routing requests. This condition is used when the proxy has more than one virtual server.

**Header**

The Sps uses the value of any HTTP header to determine how to route a request. To route requests based on the type of device being used to access resources, requests may be routed according to the USER_AGENT HTTP header.

**Note:** HTTP headers derived from SiteMinder responses may be used to determine how to route a request.

**Cookie**

The SPS uses the existence or value of a cookie to determine how to route a request. If a cookie value is encoded, specify the encoding scheme in encoding parameter. The SPS supports only the base64 encoding scheme.

**Cases**

A case is a set of specific values for conditions that provide the information the SPS needs to determine the ultimate destination for a request. For example, if a set of proxy rules uses the host condition, cases include the virtual hosts configured for the system, such as home.company.com and banking.company.com.

# Establish a Proxy Rules Configuration File

The proxy rules configuration file is an XML configuration that is identified by the rules_file directive of the <ServiceDispatcher> element in the server.conf file. The rules_file directive indicates the relative path from the installation directory to the proxy rules configuration file. At installation, the relative path to the default proxy rules configuration file is generated automatically and inserted into the rules_file directive for the default virtual host.

The generated path and proxy rules file name is the following:

*sps_home*/secure-proxy/proxy-engine/conf/proxyrules.xml

All the entries in the proxyrules.xml file must be well-formed and satisfy the syntax rules as per the XML specifications. Changes to the proxy rules configuration file do not require a server restart to take affect; the SPS detects when changes are made to the file and loads the new proxy rules file.

If the SPS detects an error in the proxy rules when parsing the rules, the SPS records an error in the server log, ignores the changes, and uses the existing proxy rules. The server log file location is specified in the logger.properties file.

**More information:**

Service Dispatcher Settings in the server.conf File (see page 102)

# Proxy Rules DTD

You must create the proxyrules.xml file using the proxy rules DTD. To view the proxy rules DTD, go to the following directory:

*sps_home*\secure-proxy\proxy-engine\conf\dtd

The following elements are configurable in the DTD:

- nete:proxyrules
- nete:description
- nete:case
- nete:cond
- nete:default
- nete:forward
- nete:redirect
- nete:local
- nete:xprcond

## nete:proxyrules

The full definition for the nete:proxyrules element is:

```
<!ELEMENT nete:proxyrules (nete:description?,(nete:cond | nete:forward |
nete:redirect | nete:local)) >
```

This element is the root element for a proxy rules XML configuration file. It contains an optional nete:description element and one of the following elements:

- nete:cond
- nete:xprcond
- nete:forward
- nete:redirect
- nete:local

The nete:proxyrules element must be present in a proxy rule configuration file.

## Debug Attribute

The nete:proxyrules element has the following attribute:

```
<ATTLIST nete:proxyrules
debug (yes|no) "no"
```

This attribute enables or disables logging that aids in debugging proxy rules. The default for this attribute is no. To enable logging, set this attribute to yes.

For example:

```
<nete:proxyrules xmlns:nete="http://www.company.com/" debug="yes">
```

When enabled, trace logging information for the SPS is included in trace logs. The location of the log file is determined by the TraceFileName parameter in the agent configuration object that you specified during the SPS installation process. The TraceConfigFile parameter in the same agent configuration object must point to the Secure Proxy-specific trace logging configuration file.

By default this file is located in:
<*install-dir*>\proxy-engine\conf\defaultagent\SecureProxyTrace.conf

**Note:** This change does not require a SPS restart for the changes to take effect.

## nete:description

The full definition for the nete:description element is:

```
<!ELEMENT nete:description (#PCDATA)>
```

This is an optional element that contains a parsed character data (PCDATA) description of another element.

**Note:** PCDATA is any text that is not markup text.

A nete:description element can be a child of the nete:proxyrules element and may contain a description of your choosing.

## nete:case

The nete:case element provides the destination associated with a specific value for a condition defined in the nete:cond parent element. The SPS uses the value of the nete:case element to evaluate a case.

The definition for the nete:case element is:

```
<!ELEMENT nete:case (nete:cond | nete:xprcond | nete:forward | nete:redirect |
nete:local)>
```

All nete:case elements must contain one of the following child elements:

**nete:cond**

nete:case elements can contain additional nete:cond elements. This allows you to nest multiple conditions in a set of proxy rules.

**nete:xprcond**

nete:case elements can contain additional nete:xprcond elements for regular expression matching of URIs. This allows you to nest a regular expression condition in a set of other conditions.

**nete:forward**

Provides a destination to which requests that fulfill the nete:case comparison will be forwarded.

**nete:redirect**

Provides a destination to which requests that fulfill the nete:case comparison will be redirected. Once redirected, requests are fulfilled directly by the destination server, rather than through the SPS.

In the following example, a nete:cond element specifies URI matching, and the nete:case elements demonstrate possible uses of the comparison type attribute.

```
<nete:cond type="uri" criteria="beginswith">
    <nete:case value="/hr">
      <nete:forward>http://hr.company.com$0</nete:forward>
        </nete:case>
        <nete:case value="/employee">
           <nete:forward>http://employees.company.com$1
                        </nete:forward>
        </nete:case>
    </nete:cond>
```

**Note:** The <nete:forward>*URL*</nete:forward> elements must be located on the same line. In the example, the </nete:forward> closing tags sometimes appear on a separate line due to space constraints, however, a line break in an actual proxy rules file causes an error. The SPS interprets line breaks before the </nete:forward> closing tag as characters that are part of the URL contained in the nete:forward element.

## Forward and Redirect Syntax

When forwarding or redirecting a request, the SPS uses a system for maintaining some part or all of the Universal Resource Indicator (URI) specified by a user. This URI points to a resource that lies on a destination server and must be interpreted by the SPS to fulfill a request.

Either of the following may be appended to a URL specified in a forward or redirect destination:

**$0**

Appends the entire URI string from a user's request to the destination specified in the proxy rule.

For example, if a proxy rule forwards all user requests for www.company.com to proxy.company.com$0 and a user request for proxy.company.com/employees/hr/index.html, that request is forwarded to www.company.com/employees/hr/index.html.

**$1**

May only be used in nete:case elements where the parent nete:cond element specifies a URI substring match using the begins with comparison. $1 indicates that everything to the right of the matching text is appended to the forwarded or redirected request.

For example, consider a proxy rules configuration file that has a nete:cond element of:

<nete:cond type="uri" criteria="beginswith">

Assume this condition is the child of a condition that is evaluating URIs for a hostname of www.company.com and a nete:case element of:

```
<nete:case value="/hr">
    <nete:forward>http://hr.company.com$1</nete:forward>
    </nete:case>
```

If a user requests:

http://www.company.com/hr/employees/index.html

The request is forwarded to:

http://hr.company.com/employees/index.html

**Note:** Because this example specifies the $1 parameter, the /hr portion of the URI is omitted when the request is forwarded to hr.company.com.

## nete:cond

The definition for the nete:cond element is:

```
<!ELEMENT nete:cond (nete:case+,nete:default?)>
```

In addition, the nete:cond element has the following attributes:

```
<!ATTLIST nete:cond type (header | host | uri | query | cookie) #REQUIRED
criteria (equals | beginswith | endswith | contains | exists) "equals"
headername CDATA #IMPLIED>
```

The nete:cond element specifies a condition that will be evaluated to determine how the SPS handles incoming requests. This element must include an attribute to be evaluated by the SPS.

Possible attributes, as defined in the ATTLIST element, include:

**header**

Specifies an HTTP header. HTTP headers are name-value pairs which can be retrieved from a user directory by SiteMinder. If you select header as the type, you must also specify the name of the header. The following is an example of a nete:cond element using header as the type:

```
<nete:cond type="header" headername="USER_AGENT">
```

This element indicates that a header will be used to determine the destination of a request, and that the header to be evaluated by the SPS is USER_AGENT. The actual destinations for requests are determined by nete:case elements which are children of nete:cond elements as described in the next section.

Conditions that use headers as the comparison require an additional argument of headername="*value*" where *value* is the name of the HTTP or SiteMinder header.

**Note:** HTTP headers generated by SiteMinder responses may be specified in nete:cond elements.

**host**

Specifies a host name in deployments where the SPS proxies for multiple virtual hosts.

Port numbers are considered part of the host, so you can use the endswith criteria, described later, in conjunction with the host condition to route requests by port number.

**query**

Specifies the query string portion of the URI that follows the '?' character. This is similar to a nete:cond that makes use of the URI as follows:

**URI**

Specifies universal resource indicator, which is the portion of a requested URL that follows the server name.

You can use the endswith criteria in conjunction with the URI condition to route requests by file extension.

**cookie**

Specifies a cookie attribute to determine how to route a request. If a cookie value is encoded, specify the encoding scheme in encoding parameter. The SPS supports only the base64 encoding scheme and does not support cookie creation. The following are the possible cases of an encoded cookie:

- If a cookie is encoded using base64, specify the cookie value in the value attribute and base64 as the encoding parameter in the nete:case element. The SPS uses the base64 encoding algorithm to decode the cookie value received from a httprequest and compares the result the decoded value with the value specified in the value attribute.
- If a cookie is not encoded, enter the cookie value in plain text in the value attribute and you can specify the encoding parameter as blank in the nete:case element. The SPS accepts the specified plain text as the cookie value and verifies the nete:cond.

If a cookie is encoded using a different encoding scheme, enter the encoded cookie value in the value attribute and specify the encoding parameter as blank in the nete:case element. The SPS accepts the specified encoded cookie value as plain text, and uses the plain text cookie value to verify the nete:cond

One of the type attributes described above must be included in the nete:cond element. In addition, the nete:cond element must specify a criteria that defines the comparison that the proxy engine executes on the value of the condition for an incoming request. Possible criteria are:

**equals**

Indicates that the value of the type attribute of the nete:cond parent element must equal the contents of the value attribute of the nete:case element to act on a request.

**beginswith**

Indicates that the value of the type attribute of the nete:cond parent element must begin with the contents of the value attribute of the nete:case element to act on a request.

**endswith**

Indicates that the value of the type attribute of the nete:cond parent element must end with the contents of the value attribute of the nete:case element to act on a request.

**contains**

Indicates that the value of the type attribute of the nete:cond parent element must contain the contents of the value attribute of the nete:case element to act on a request.

**exists**

Indicates that the nete:cond parent element must exist and the value attribute of the nete:case element must be true to act on a request. You can use the exists criteria with only header and cookie attributes.

**Note:** If no criteria are specified, the SPS assumes the default criteria of equals.

Each nete:cond element must have one or more nete:case child elements. The nete:case children provide the unique values that the SPS uses to route requests to appropriate destinations.

## nete:default

The definition of the nete:default element is:

```
<!ELEMENT nete:default (nete:cond | nete:xprcond | nete:forward | nete:redirect |
nete:local)>
```

This element is required and must be a child element of each nete:cond element. If a request does not meet the requirements of any nete:case elements contained in nete:cond elements, the nete:default element determines how to handle the request.

The possible child elements associated with the nete:default element are identical to the elements available for a nete:case element. If you create nete:cond elements as children to a nete:default, be careful to take into account a default case so that all possible client requests may be handled by the SPS.

In the following example, the nete:default element forwards all requests that do not meet the criteria of any other proxy rules to a home page of general information.

```
<nete:default>
    <nete:forward>http://home.company.com/index.html
    </nete:forward>
</nete:default>
```

The opening and closing tags, <nete:forward>*URL*</nete:forward>, elements must be located on the same line. In the example, the </nete:forward> closing tags sometimes appear on a separate line due to space constraints, however, a line break in an actual proxy rules file causes an error. The SPS interprets line breaks before the </nete:forward> closing tag as characters that are part of the URL contained in the nete:forward element.

## nete:forward

The definition of the nete:forward element is:

```
<!ELEMENT nete:forward (#PCDATA)>
```

The nete:forward element forwards a request to a specified URL.

**Note:** The <nete:forward> and </nete:forward> tags must be located on a single line in the proxy rules file. If they are not located on the same line, the SPS interprets line breaks as part of the URL contained in the element. This causes the forward service to fail.

In the following example, the nete:forward element forwards requests while maintaining the URI requested by the user.

```
<nete:forward>http://home.company.com$0</nete:forward>
```

If the user's request satisfies the nete:case parent element's criteria, that request is forwarded to home.company.com. Therefore, a request for http://server.company.com/hr/benefits/index.html that is forwarded by the previous nete:forward element is fulfilled by forwarding the request to http://home.company.com/hr/benefits/index.html.

If you want to forward a request over SSL, be sure to use https instead of http when defining the destination contained in the <nete:forward> element.

The nete:forward element contains the following attribute:

```
<!ATTLIST nete:forward filter CDATA #IMPLIED>
```

This attribute allows you to specify the name of a Java filter class that can be invoked during a forward from the SPS to a destination server. Filters can be written using the Filter API.

**More information:**

Forward and Redirect Syntax (see page 148)
Filter API Overview (see page 246)

The nete:forward element contains the following attribute:

```
<!ATTLIST nete:forward filter CDATA #IMPLIED>
```

This attribute allows you to specify the name of a java filter class that can be invoked during a forward from the SPS to a destination server. Filters can be written according to the Filter API.

## nete:redirect

The definition of the nete:redirect element is:

```
<!ELEMENT nete:redirect (#PCDATA)>
```

The nete:redirect element specifies a response that is returned to a user which redirects the user request to an appropriate destination server. The PCDATA follows the standard forward and redirect syntax. Once redirected, the SPS does not handle the completion of the request. Instead, the request is handled by the server that is the target of the redirection.

The <nete:redirect> and </nete:redirect> tags must be located on a single line in the proxy rules file. If they are not located on the same line, the SPS interprets line breaks as part of the URL contained in the element. This causes the redirect service to fail.

In the following example, the nete:redirect element redirects requests while maintaining the URI requested by the user. Unlike a nete:forward element, once the request has been redirected, the SPS is removed from the transaction, and the destination server provides resources directly to the user.

```
<nete:redirect>http://home.company.com$0</nete:redirect>
```

If a user's request for http://www.company.com/hr/index.html meets a parent nete:case element's criteria and is redirected by the above example, the user is redirected and the user's browser displays the URL of the destination server fulfilling the request:

http://home.company.com/hr/index.html

**Note:** If you want to redirect a request over SSL, be sure to use https instead of http when defining the destination contained in the <nete:redirect> element.

**More information:**

Forward and Redirect Syntax (see page 148)

## nete:local

The nete:local element is included to support future functionality, and should not be included in proxy rules configuration files.

## nete:xprcond

The nete:xprcond element may be used like a nete:cond element in situations where you want to apply regular expressions as conditions in your proxy rules. Regular expressions can be used to evaluate the URI string and any attached query string in your proxy rules.

The definition of the nete:xprcond element is:

```
<!ELEMENT nete:xprcond (nete:xpr+, nete:xpr-default)>
```

This element must contain one or more nete:xpr elements and a single nete:xpr-default element.

### nete:xpr and nete:xpr-default

A nete:xpr element is similar to a nete:cond element, and contains other elements for a rule based on a regular expression as well as a resulting behavior when the SPS finds a match for the expression. A nete:xpr-default element contains the default behavior for URI or query string combinations that do not match any of the regular expressions contained in the nete:xpr elements within a nete:xprcond element.

The definition of a nete:xpr element is:

```
<!ELEMENT nete:xpr (nete:rule, nete:result)>
```

A nete:xpr element contains a nete:rule element that defines a regular expression, and a nete:result element that specifies the behavior for strings that match the regular expression.

The definition of a nete:xpr-default element is:

```
<!ELEMENT nete:xpr-default (nete:forward|nete:redirect|nete:local)>
```

The nete:xpr-default element specifies a forward or redirect that should be completed if the URI or query string being evaluated by the SPS does not match the conditions stated in any of the nete:xpr elements contained in the parent nete:xprcond element.

## nete:rule and nete:result

The nete:rule and nete:result elements must be contained in a nete:xpr element. The nete:rule element specifies the regular expression that the SPS evaluates against incoming requests. The nete:result element determines the behavior for matching requests.

The definition of the nete:rule element is:

```
<!ELEMENT nete:rule (#PCDATA)>
```

This element contains a string that is a regular expression. The URI and query string of a request is matched against this regular expression to determine if a request satisfies the nete:xpr condition.

The definition of the nete:result element is:

```
<!ELEMENT nete:result (#PCDATA)>
```

nete:result elements may have the following attributes:

```
<!ATTLIST nete:result service (forward|redirect) "forward">
```

This element contains a string consisting of the substitution string (URL) by which the SPS creates a URL to pass to a service (forward or redirect). The service attribute is used to specify the appropriate service that will receive the URL. The default service is the forward service defined in the server.conf configuration file.

The substitution URL in the nete:result element should be a URL that optionally contains $#, where # is 0, 1, 2, etc.

The substitution URL in the nete:result element should be a URL that optionally contains $#, where # is 0, 1, 2, etc.

- $0 is the entire URI and query string being evaluated.
- $n is the part of the requested URI that matched the nth set of parentheses in the regular expression described in the associated nete:rule element.

For example, a set of proxy rules can contain the following:

```
<nete:xprcond>
    <nete:xpr>
        <nete:rule>^/realma(.*)</nete:rule>
        <nete:result>http://server1.company.com$1</nete:result>
    </nete:xpr>


    <nete:xpr-default>
        <nete:forward>http://www.company.com$0</nete:forward>
    </nete:xpr-default>
</nete:xprcond>
```

The <nete:result>URL</nete:result> tags must be located on a single line in the proxy rules file. If they are not located on the same line, CA SiteMinder SPS interprets line breaks as part of the URL contained in the element. This causes the result service to fail.

In the previous nete:xprcond proxy rule example, a request for:

http://www.company.com/realma/index.html

will be forwarded to:

http://server1.company.com/index.html

# How nete:xprcond Elements Works

The processing of a nete:xprcond element is similar to the processing of all other nete:cond elements. As the SPS processes requests, and it encounters a nete:xprcond element in the proxy rules configuration file, the following occurs:

1. The SPS examines the first nete:xpr element in the nete:xprcond element.
2. The proxy engine evaluates the regular expression described in the nete:rule element against the URI and query string of the request.
3. If the requested URI and query string matches the regular expression specified in the nete:rule element, then the SPS resolves the result string using the results of the match, and the request is forwarded to the specified service with the URL derived from the nete:result element.
4. If the requested URI and query string do not match the regular expression in the first nete:xpr element, the proxy rules engine evaluates the next nete:xpr element (repeat steps 2 and 3). This process continues until the rules engine finds a match or reaches the nete:xpr-default element.
5. If no match is found before reaching the nete:xpr-default element, then the contents of the nete:xpr-default element determine how to route the request.

# Regular Expression Syntax

This section describes the syntax that should be used to construct regular expressions for nete:rule elements. A nete:xprcond element takes the following form:

```
<nete:xprcond>
    <nete:xpr>
        <nete:rule>regular_expression</nete:rule>
        <nete:result>result</nete:result>
    </nete:xpr>
    <nete:xpr-default>forward_destination</nete:xpr-default>
</nete:xprcond>
```

In the nete:xpr element, the nete:rule element must consist of a regular expression that uses the syntax described in the following table. This syntax is consistent with the regular expression syntax supported by Apache and described at http://www.apache.org.

| Characters | Results |
|---|---|
| unicode character | Matches any identical unicode character |
| \ | Used to quote a meta-character like '*') |
| \\ | Matches a single '\' character |
| \0nnn | Matches a given octal character |
| \xhh | Matches a given 8-bit hexadecimal character |
| \\uhhhh | Matches a given 16-bit hexadecimal character |
| \t | Matches an ASCII tab character |
| \n | Matches an ASCII newline character |
| \r | Matches an ASCII return character |
| \f | Matches an ASCII form feed character |
| [abc] | Simple character class |
| [a-zA-Z] | Character class with ranges |
| [^abc] | Negated character class |
| [:alnum:] | Alphanumeric characters |
| [:alpha:] | Alphabetic characters |
| [:blank:] | Space and tab characters |
| [:cntrl:] | Control characters |
| [:digit:] | Numeric characters |
| [:graph:] | Characters that are printable and are also visible (A space is printable, but not visible, while an 'a' is both) |
| [:lower:] | Lower-case alphabetic characters |
| [:print:] | Printable characters (characters that are not control characters) |
| [:punct:] | Punctuation characters (characters that are not letter, digits, control characters, or space characters) |
| [:space:] | Space characters (such as space, tab, and formfeed) |
| [:upper:] | Upper-case alphabetic characters |
| [:xdigit:] | Characters that are hexadecimal digits |

| Characters | Results |
|---|---|
| [:javastart:] | Start of a Java identifier |
| [:javapart:] | Part of a Java identifier |
| . | Matches any character other than newline |
| \w | Matches a "word" character (alphanumeric plus "_") |
| \W | Matches a non-word character |
| \s | Matches a whitespace character |
| \S | Matches a non-whitespace character |
| \d | Matches a digit character |
| \D | Matches a non-digit character |
| ^ | Matches only at the beginning of a line |
| $ | Matches only at the end of a line |
| \b | Matches only at a word boundary |
| \B | Matches only at a non-word boundary |
| A* | Matches A 0 or more times (greedy) |
| A+ | Matches A 1 or more times (greedy) |
| A? | Matches A 1 or 0 times (greedy) |
| A{n} | Matches A exactly n times (greedy) |
| A{n,} | Matches A at least n times (greedy) |
| A{n,m} | Matches A at least n but not more than m times (greedy) |
| A*? | Matches A 0 or more times (reluctant) |
| A+? | Matches A 1 or more times (reluctant) |

| Characters | Results |
|---|---|
| A?? | Matches A 0 or 1 times (reluctant) |
| AB | Matches A followed by B |
| A\|B | Matches either A or B |
| (A) | Used for subexpression grouping |
| \1 | Backreference to 1st parenthesized subexpression |
| \n | Backreference to nth parenthesized subexpression |

All closure operators (+, *, ?, {m,n}) are greedy by default, meaning that they match as many elements of the string as possible without causing the overall match to fail. If you want a closure to be reluctant (non-greedy), you can simply follow it with a '?'. A reluctant closure will match as few elements of the string as possible when finding matches. {m,n} closures don't currently support reluctancy.

## Regular Expression Examples in nete:rule and nete:result

Regular expressions offer a very flexible and powerful tool that can be employed in SPS proxy rules. This section provides a few examples nete:rule elements in proxy rules. In addition, the examples also contain various uses of the nete:result element to show how groupings in a nete:rule can be used to affect the destination generated by the children of a nete:xprcond element.

### Map Single Rule to Many Destination Servers

In the following example, a nete:rule element contains a regular expression that can be used to forward requests to many different destinations. This example assumes that the SPS will receive URIs that take the following form:

/GOTO=*some path and or filename*

Consider a nete:xprcond element contains the following child elements:

```
<nete:xpr>
    <nete:rule>/GOTO=(.*)/(.*)</nete:rule>
    <nete:result>http://$1/$2</nete:result>
</nete:xpr>
```

The regular expression in the nete:rule element and the associated nete:result element match URIs that produce a /GOTO=string. Upon finding a match, the SPS uses the first string after the = symbol in the URI as the $1 value of the result, and the value following the first / symbol that appears after the = symbol as the $2 result. The nete:result element combines these elements to create a URL. By default, nete:result elements use the SPS forward service.

For example, if the URI of a request evaluated by the nete:xpr element described above were as follows:

/GOTO=server1.company.com/index.html

Then the regular expression in the nete:rule element would find a match and assign the value of $1 as server1.company.com and the value of $2 as index.html. The nete:result element assembles these values into the following URL:

http://server1.company.com/index.html

This URL is the target which the SPS uses to resolve the request.

### Regular Expressions to Redirect Users

The nete:result element can also be used to create a redirect response that is returned to the user requesting the resource. This forces the fulfillment of a request to be handled by a server other the SPS after authentication and authorization. The following is an example of a nete:xpr element that specifies a redirect in the nete:result child element.

```
<nete:xpr>
    <nete:rule>/REDIR=(.*)/(.*)</nete:rule>
    <nete:result service="redirect">http://$1/$2</nete:result>
</nete:xpr>
```

**Note:** The service attribute instructs the SPS to use the redirect service in place of the default forward service.

## Header Values in Forwards, Redirects, and Results Filters

The value of an HTTP header or a SiteMinder response header can be substituted directly into a nete:forward, nete:redirect, or nete:result element. When a URI in a forward or redirect element, or a rule in a result filter element contains {{*HEADER_NAME*}}, the proxy engine searches for a header in a request that matches the specified header and substitute's the header value before resolving the forward, redirect, or result. If no matching header is found in a request, the proxy engine substitutes an empty string in place of the header value.

**Note:** Header names are case sensitive.

## Dynamic Header Value in a nete:forward

To use a dynamic header value as part of a nete:forward element, simply insert *{{HEADER_NAME}}* into the URL portion of the forward. For example:

```
<nete:forward>http://www.company.com/{{RESPONSE1}}$1</nete:forward>
```

You can use multiple headers in a single nete:forward element. For example:

```
<nete:forward>http://www.company.com/{{RESPONSE1}}/{{RESPONSE2}}$1
</nete:forward>
```

## Dynamic Header Value in a nete:redirect

To use a dynamic header value as part of a nete:redirect element, simply insert *{{HEADER_NAME}}* into the URL portion of the redirect. For example:

```
<nete:redirect>http://www.company.com/{{RESPONSE1}}$1</nete:redirect>
```

You can use multiple headers in a single nete:redirect element. For example:

```
<nete:redirect>http://www.company.com/{{RESPONSE1}}/{{RESPONSE2}}$1
</nete:redirect>
```

## Dynamic Header Value in a nete:result

To use a dynamic header value as part of a nete:result element, simply insert *{{HEADER_NAME}}* into the URL portion of the result. For example:

```
<nete:result>http://www.company.com/{{HEADER_NAME}}$1</nete:result>
```

You can use other features of proxy rules, such as filters, in conjunction with a dynamic header value. For example:

```
<nete:result filter="filter1">http://$1/$2?{{HEADER_NAME}}</nete:result>
```

# Response Handling

The SPS uses SiteMinder responses to determine a destination for a request. Since transactions that are routed through the SPS include an interaction between the SPS web agent and SiteMinder, any SiteMinder responses gathered during the authentication and authorization process may be used by the SPS to determine the destination of a request.

For example, if a user directory contains information about the account type for a banking web site, the SPS can proxy users with different types of accounts to different destinations. This enables an enterprise to provide a higher quality of service to its best customers. Customers with standard accounts can be handled by one set of destination servers, while customers with premium accounts can be handled by a separate set of high performance destination servers.

# Modify Proxy Rules

To modify proxy rules you must edit the proxy rules XML configuration file using a text editor. Since proxy rules are XML files, your proxy rules configuration file must be well-formed and valid. Remember that the tags in a well-formed XML file must all consist of opening and closing tags. To be valid, the file must adhere to the guidelines laid out in the proxyrules.dtd.

Changes to the proxy rules XML configuration file are picked up automatically by the SPS. When the SPS receives a request, it checks whether or not the proxy rules have changed. If the file has changed, the rules are reloaded before fulfilling the request.

Changes to the proxy rules XML configuration file are picked up automatically by the SPS. When the SPS receives a request, it checks whether or not the proxy rules have changed. If the file has changed, the rules are reloaded before fulfilling the request.

**Note:** If you change the name of the proxy rules XML configuration file in the rules_file directive in the <ServiceDispatcher> element of the server.conf file, you must restart the SPS.

# Sample Proxy Rules Configuration Files

The SPS installs several examples of proxy rules configuration files. You can use these example XML files as the basis for your own proxy rules files.

You can find these example files in the directory *sps_home*\secure-proxy\proxy-engine\examples\proxyrules. We recommend you look at the example file as you are reading the descriptions in this guide.

You may copy and customize a file to suit your needs.

**To customize and deploy a proxy rules file**

1. Navigate to the directory *sps_home*\secure-proxy\proxy-engine\examples\proxyrules.
2. Make a copy of the example file you want to use.
3. Customize the content and save the new file under a unique name.
4. Copy the modified file to the directory *sps_home*/secure-proxy/proxy-engine/conf.
5. Open up the server.conf file to modify the proxy rules section of the file to point to the customized file.

## Proxy Rules Example—Routing Requests by Virtual Host

The example file proxyrules_example1.xml file routes requests based on the hostname specified in the request. The example file proxyrules_example10.xml file also routes SPS requests based on the hostname specified in the request, SPS uses the PID in the proxy rule to count of the number of times the proxy rule has been triggered. If you configured CA Wily Introscope to monitor SPS, the count is displayed in CA Wily Introscope data metrics.

In this file, a simple set of proxy rules routes user requests based on the virtual host specified in the requested resource. All requests to the bondtrading.company.com server are forwarded to server2, all requests to banking.company.com are forwarded to server1, and all other requests are forwarded to the companies home server, which is the default for requests that do not match the criteria in any other nete:cond element.

**Note:** The nete:case elements must specify a port, since the port number is considered as part of the virtual host requested by the user. Use the beginswith criteria to avoid needing port numbers.

The following table illustrates the results of requests using the proxy rules based on virtual hosts.

| **Requested URL** | **Forwarded URL** |
|---|---|
| http://banking.company.com/index.html | http://server1.company.com/index.html |
| http://bondtrading.company.com/index.html | http://server2.company.com/index.html |
| http://www.company.com/index.html | http://home.company.com/index.html |

## Proxy Rules Example—Routing Requests by Header Value

The example file proxyrules_example2.xml file routes SPS requests based on the value of an HTTP header. The HTTP header can be a standard header or one created using a SiteMinder response.

**Note:** For information about SiteMinder responses, see the *CA SiteMinder Policy Design.*

In this example, assume that the SPS routes requests made to a default virtual host of www.company.com.

In this file, the value of the HTTP header variable "HEADER" determines the destination for the request.

The following table illustrates the results of requests using the proxy rules based on an HTTP header.

| **Requested URL** | **Forwarded URL** |
|---|---|
| http://www.company.com/index.html<br>HTTP_HEADER has the following value:<br>HTTP_HEADER="value1" | http://server1.company.com/index.html |
| http://www.company.com/index.html<br>HTTP_HEADER has the following value:<br>HTTP_HEADER="value2" | http://server2.company.com/index.html |
| http://www.company.com/index.html<br>HTTP_HEADER has a value other than value1 or value2. | http://home.company.com/index.html |

**Note:** You do not need to include the HTTP_ of the header variable name in the nete:cond element. SPS assumes HTTP_ for header variable names.

Proxy rules that use header values are an excellent way to forward requests based on a desired level of service. For example, you can use the value of an HTTP header variable that contains a user account types to distribute requests to high performance servers for customers with premium accounts.

## Proxy Rules Example—Routing Requests by Device Type

The example file proxyrules_example3.xml file routes SPS requests based on the type of device used to access the resource.

**Note:** The user-agent HTTP header value is used to determine how to route requests.

In the file, users who access resources using a browser (user agent contains Mozilla for Web browsers) are forwarded to a Web server, while all other users are forwarded to a wireless server.

The following table illustrates the results of requests using the proxy rules based on a device type.

| Requested URL | Forwarded URL |
|---|---|
| http://www.company.com/index.html<br>User access resource via a Web browser. | http://home.company.com/index.html |
| http://www.company.com/index.wml<br>User access resource via a wireless device. | http://wireless.company.com/index.wml |

## Proxy Rules Example—Routing Requests with URIs

The example file proxyrules_example4.xml file routes SPS requests based on the URI specified in the user request.

The following table illustrates the results of requests using the proxy rules based on URIs.

| Requested URL | Forwarded URL |
|---|---|
| http://www.company.com/dir1/index.html | http://server1.company.com/index.html |
| http://www.company.com/dir2/index.html | http://server2.company.com/index.html |
| http://www.company.com/index.html | http://home.company.com/index.html |

## Proxy Rules Example—Routing Requests by File Extension

The example file proxyrules_example5.xml file routes SPS requests based on the file extension requested by the user. This is achieved by using the URI condition in combination with the endswith criteria.

In the file, the <nete:forward> and </nete:forward> tags appear on separate lines due to space constraints. However, in your proxy rules configuration files, the opening and closing tags for a <nete:forward> element must appear on the same line. If they do not, the SPS interprets the line break as part of the forward URL, which causes requests to be forwarded incorrectly.

In the previous example, users who access .jsp resources are forwarded to an application server, while wireless users are forwarded to the wireless server. All other users are forwarded to the home server.

The following table illustrates the results of requests using the proxy rules based on file extensions.

| **Requested URL** | **Forwarded URL** |
|---|---|
| http://www.company.com/app.jsp | http://application.company.com/app.jsp |
| http://www.company.com/index.wml | http://wireless.company.com/index.wml |
| http://www.company.com/index.html | http://home.company.com/index.html |

## Proxy Rules Example—Routing Requests with Nested Conditions

The example file proxyrules_example6.xml file routes SPS requests based on the hostname, certain headers, and device types. This file demonstrates how the SPS can handle complex relationships in a single configuration file.

In the file, the <nete:forward>*URL*</nete:forward> elements must be located on the same line. In the example, the </nete:forward> closing tags sometimes appear on a separate line due to space constraints, however, a line break in an actual proxy rules file causes an error. The SPS interprets line breaks before the </nete:forward> closing tag as characters that are part of the URL contained in the nete:forward element.

The following table illustrates the results of requests using proxy rules with nested conditions.

| Requested URL | Forwarded URL |
|---|---|
| http://banking.company.com/index.wml | http://wireless.company.com/banking/index.wml |
| http://banking.company.com/index.html | http://server1.company.com/banking/index.html |
| http://bondtrading.company.com/index.html<br>with a header value of GOLD_USER="yes" | http://fast.company.com/bondtrading/index.html |
| http://bondtrading.company.com/index.html<br>with a header value of GOLD_USER="no" | http://server2.company.com/bondtrading/index.html |
| http://www.company.com/index.wml<br>with a USER_AGENT header value that contains a wireless device name | http://home.company.com/<br>wireless/index.wml |
| http://www.company.com/index.html<br>with a USER_AGENT header value that does not contains a wireless device name | http://home.company.com/index.html |

## Proxy Rules Example—Using Regular Expression in Proxy Rules

The example file proxyrules_example7.xml file routes SPS requests based on nete:xprcond elements that contain regular expressions. Regular expressions are evaluated based on the URI and query string of a request.

In the file, the URI and query string of the request are evaluated against the three regular expressions defined in the nete:xpr elements. If a match is not found against the first nete:xpr element, the SPS tries to match it against the second, and finally the third regular expression. If no matches are found, the nete:xpr-default condition is used to handle the request.

The following table lists the results of requests using the regular expression proxy rules.

| Requested URL | Forwarded URL |
|---|---|
| http://server.company.com/realma/hr/index.html | http://server1.company.com/hr/index.html |

| Requested URL | Forwarded URL |
|---|---|
| http://server.company.com/GOTO=server2.company.com/index.html | http://server2.company.com/index.html |
| http://server.company.com/REDIR=server2.company.com/index.html | http://server2.company.com/index.html<br>User is redirected so that server2.company.com must directly fulfill the user's request. |
| http://server.company.com/index.html | http://www.company.com/index.html |

## Proxy Rules Example—Routing Requests by Cookie Existence

The example file proxyrules_example8.xml file routes SPS requests based on the existence of a cookie.

In this example, if a request contains a cookie header "mycookie", SPS routes the request to www.company.com.

## Proxy Rules Example—Routing Requests by Cookie Value

The example file proxyrules_example9.xml file routes SPS requests based on the value of a cookie.

In this example, if a request contains a cookie header "mycookie" and the request does not specify encoding mechanism, SPS routes the request to www.abcd.com. If a request contains a cookie header "mycookie" and the base64 encoding mechanism, SPS routes the request to www.xyz.com.

# Chapter 10: Deploying the SPS

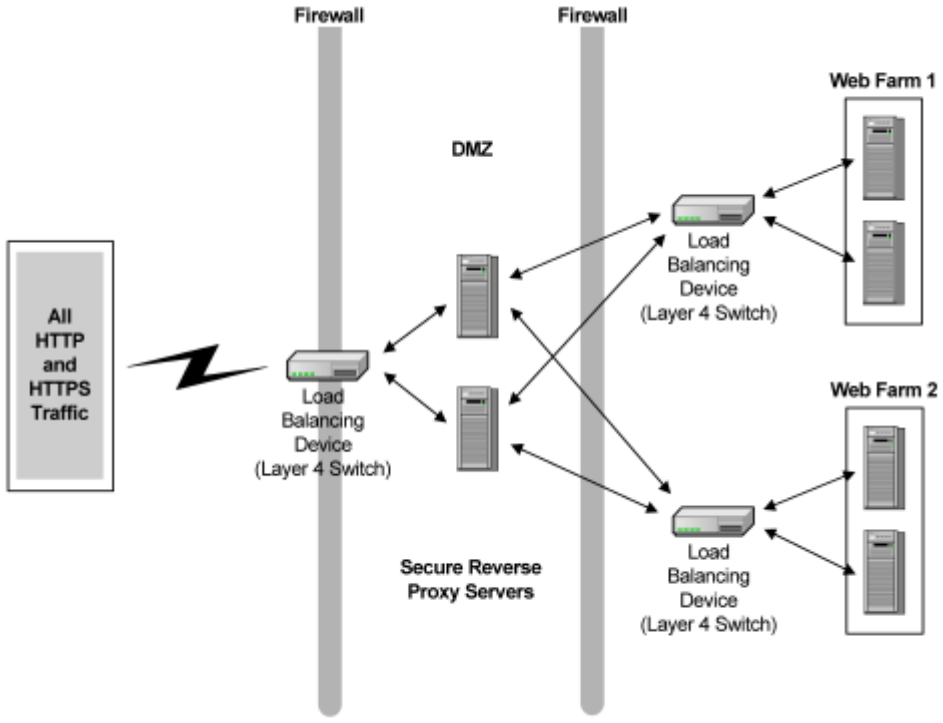
This section contains the following topics:

SPS Deployment in an Enterprise (see page 169)

## SPS Deployment in an Enterprise

The SPS uses reverse proxy architecture to enable access control, single sign-on, and SSL acceleration. It does not provide the content caching and some other features provided by traditional reverse proxy servers. The SPS is intended to be an addition to enterprise architecture, rather than a replacement for other proxy technologies. As such, the SPS can be placed in clusters with load balancing devices and caching devices on either side of the clusters.

The following illustration shows how the SPS can be inserted into a network to work in conjunction with load balancing devices.




**Note:** In addition to load balancing devices, caching devices can be placed on either side of the SPS cluster.

## Sticky-Bit Load Balancing

When using the cookieless session schemes supported by the SPS, session information for users who access resources through SPS is maintained in an in-memory session store. Because the session information is maintained at the SPS where a user is first authenticated, the same SPS should be used for all the user requests in a single session. When implemented in clusters, the SPS must be used in conjunction with sticky-bit load balancers to provide a consistent connection to the same SPS, enabling single sign-on when using session schemes other than the traditional SiteMinder cookie session scheme.

To deploy the SPS using cookieless session schemes the following must be considered:

- In most deployments, the SPS is deployed in clusters, with several servers sharing the load of incoming requests. The load balancing is handled by load balancer devices. These devices must have sticky bit capability to maintain single sign-on.

  Sticky bit load balancers ensure that once a user's session is established with a specific SPS in a cluster, that SPS services all of the user's requests. This capability is required because the SPS maintains session information for cookie-less sessions in active memory. If a user's request is not handled using sticky bit technology, the user will be charged for new credentials each time a request is fulfilled by a different SPS in the cluster of servers.
- When configuring the settings for the SPS, the default virtual host defined in the server.conf file of the SPS must be defined using the name and IP address of the load balancing device.
- The load balancing device must be configured as the point of entry to the SPS.
- The load balancing device must point to the cluster of SPSs.
- The httpd.conf file, located in the *sps_home*/secure-proxy/httpd/conf directory, must be modified so that the value of the ServerName directive is set to the name of the load balancing device, not the system on which you installed the SPS.
- If using SSL, a certificate must be issued to the load balancer, not the SPS.
- The system or systems on which you install the SPS must have approximately 1K of memory for each simultaneous user session that will be maintained in the in-memory session store. For example, if a single system must maintain 1000 concurrent sessions, the system must have 1 megabyte of RAM available for this purpose.

## Proxying to Trusted Sites vs. Non-Trusted Sites

The SPS is assumed to proxy for trusted sites within the enterprise. As part of a proxy transaction, SiteMinder generated HTTP header variables and any variables generated by SiteMinder responses are forwarded along with each HTTP and HTTPS request. These responses can be used by other enterprise applications.

**Important!** If you employ the SPS in transactions that proxy for content on non-trusted sites, the headers that accompany the transaction will also be forwarded to the non-trusted sites. We recommend using the SPS to proxy for destination servers trusted by your enterprise.

## Configuring Virtual Hosts for the SPS

You can configure the SPS with multiple hosts and act as a virtual host for one or more hostnames.

**To configure SPS as multiple hosts and act as a virtual host for one or more hostnames**

1. Edit the <VirtualHost> parameters of the server.conf file to configure the SPS to act as a virtual host for one or more hostnames.
2. Edit the configuration file for the embedded Apache Web server.

**More information:**

Virtual Host Names Configuration (see page 130)

### Edit the Apache Configuration File To Handle Multiple Virtual Hosts

When you are running multiple virtual hosts in the same operating environment with the SPS, and transactions run in this environment, update the Apache configuration file (httpd.conf). This file is located in the directory *sps_home*\secure-proxy\httpd\conf. If SSL is enabled for the web server, also make the same updates to the httpd-ssl.conf file, which is located in the *sps_home*\secure-proxy\httpd\conf\extra directory. The updates vary depending on whether your operating environment is based on IPv4 or IPv6.

**To update the httpd.conf file, and optionally the httpd-ssl.conf file, to handle multiple virtual hosts**

- For IPv4 environments, add the following LISTEN directive:

  LISTEN 127.0.0.1:<*_port*>

- For IPv6 environments, add the following LISTEN directive:

  LISTEN [::1]:<*_port*>

- For dual stack environments with IPv4 and IPv6 supports, add the following LISTEN directives:

  LISTEN 127.0.0.1:<*_port*>

  LISTEN [::1]:<*_port*>

In addition, update the loopback address entry in the hosts file so that the new host name is added, as follows:

- IPV4: 127.0.0.1
- IPV6: [::1]

The hosts file is usually located on Windows in C:\WINDOWS\system32\drivers\hosts. On UNIX, the hosts file is usually in /etc/hosts.

## Implementing Session Scheme Mappings for Multiple Virtual Hosts

If you want to configure the SPS to recognize multiple user agent types, assign different session scheme mappings for those user agents based on virtual hosts, you must follow these steps:

1. Configure session schemes, or verify the configuration of the schemes included with the SPS.
2. Define user agent types in the server.conf file.
3. Create a section for each virtual host in the server.conf file that defines any directives that differ from default settings (refer to Overriding Default Values for a Virtual Host).
4. Define session scheme mappings for each virtual host.

The following excerpts from a server.conf file provide an example where a user agent type has been defined for Internet Explorer (IE) browser users. IE users will be mapped to use session schemes other than the default session scheme defined for a virtual host. The following example shows the session schemes defined in the server.conf file.

```
#Session Schemes
<SessionScheme name="default">
        class="com.netegrity.proxy.session.SessionCookieScheme"
        accepts_smsession_cookies="true"
</SessionScheme>
<SessionScheme name="ssl_id">
        class="com.netegrity.proxy.session.SSLIdSessionScheme"
        accepts_smsession_cookies="false"
</SessionScheme>
<SessionScheme name="simple_url">
        class="com.netegrity.proxy.session.SimpleURLSessionScheme"
        accepts_smsession_cookies="false"
</SessionScheme>
<SessionScheme name="minicookie">
        class="com.netegrity.proxy.session.MiniCookieSessionScheme"
        accepts_smsession_cookies="false"
        cookie_name="MiniMe"
</SessionScheme>
```

The following example shows the definition of the IE user agent type. This user agent type will be referenced when defining session scheme mappings later in the server.conf file.

```
# TO-DO: Define Any User Agents, if you want to
# use a different session scheme based on
# the type of client accessing the server.
#
# NOTE:  UserAgent matching is done in the order
# in which the user agents are defined in this file.
 <UserAgent name="IE">
     User-Agent="MSIE"
 </UserAgent>
# <UserAgent name="NS">
#     User-Agent=some other regular expression
# </UserAgent>
```

The preceding example shows that the default session scheme specified in the defaultsessionscheme directive is mini-cookie. This session scheme will be used for all transactions unless another session scheme is explicitly included in a session scheme mapping, or another scheme overrides the default session scheme in the definition of a virtual host.

The <VirtualHostDefaults> directive shows the session scheme mapping for the IE user agent type that was defined in <UserAgent name="IE">. This mapping indicates that for all virtual hosts using default session scheme mappings, IE browser users' sessions will be maintained using the simple URL rewriting sessions scheme.

```
<VirtualHostDefaults>
        # Service Dispatcher
        <ServiceDispatcher>
                class="com.netegrity.proxy.service.SmProxyRules"
                rules_file="conf\proxyrules.xml"
        </ServiceDispatcher>
        # default session scheme
        defaultsessionscheme="minicookie"
        #TO-DO:  Define any session scheme mappings
        <SessionSchemeMappings>
       #    user_agent_name=session_scheme_name
           IE="simple_url"
       #    NS=simple_url
        </SessionSchemeMappings>
```

The Virtual Host directives show the server name and IP address for the default virtual host configured for the SPS.

```
# Default Virtual Host
<VirtualHost name="default">
        hostnames="server1, server1.company.com"
        addresses="192.168.1.10"

        #The defaults can be overriden
        #not only for the Virtual Host
        #but for the WebAgent for that
        #virtual host as well
        #<WebAgent>
        #</WebAgent>

</VirtualHost>
```

The Virtual Host directive for additional virtual host shows the specific default virtual host settings that will be overridden for the server2 virtual host. Notice that these overrides include new session scheme mappings. The default scheme for server2 is default. In Session Scheme directive the default is defined as the traditional SiteMinder cookies session scheme. Further, the session scheme mapping for IE users in Virtual Host directives is also mapped to the default scheme. Therefore, the SPS will use SiteMinder cookies session scheme to maintain sessions for all users who access server2.

```
# Additional Virtual Host
<VirtualHost name="host2">
       requestblocksize="4"
        responseblocksize="4"
        hostnames="server2, server2.company.com"
        #addresses="192.168.1.15"
        # default session scheme
        defaultsessionscheme="default"

        #TO-DO:  Define any session scheme mappings
        <SessionSchemeMappings>
       #user_agent_name=session_scheme_name
           IE="default"
        </SessionSchemeMappings>

        #<WebAgent>
        #</WebAgent>
</VirtualHost>
```

# Chapter 11: Integrating the SPS with SiteMinder

This section contains the following topics:


How the SPS Interacts with SiteMinder (see page 177)
SPS and SharePoint Resources (see page 183)
SPS and ERP Resources (see page 183)
Password Services for SPS (see page 184)
Configuring Managed Self Registration for the SPS (see page 186)
Firewall Considerations (see page 189)
Keep Alive and Connection Pooling (see page 189)
HTTP Header Configuration for Sun Java Web Servers (see page 189)
HTTP Header for SiteMinder Processing with SPS (see page 190)
Handling Encoded URLs (see page 190)


## How the SPS Interacts with SiteMinder

SiteMinder is a solution for securely managing e-business. SiteMinder consists of a Policy Server that allows you to specify policies for your enterprise, and Web Agents that are installed on web servers. The Web Agents communicate with the Policy Server to provide authentication, authorization, and other functions.

The SPS contains a Web Agent that is compatible with the SiteMinder Web Agent and Policy Server technology. As all SiteMinder Web Agents, the SPS must be configured as an object in SiteMinder. Policies must be created that determine authentication and authorization requirements for accessing destination servers.

SiteMinder objects are configured using SiteMinder <adminui>. You can configure the following objects:

**Agent**

Configure an agent object with settings for the Web Agent included in the SPS. Specify this Web Agent when creating realms.

**User Directories**

Configure connections to any user directories that authenticate and authorize users.

**Policy Domains**

Configure policy domains that contain realms, rules, and policies.

**Realms**

Configure realms that contain resources you want to protect with SiteMinder.

**Rules**

Configure rules that identify specific resources and actions that you want to protect with SiteMinder.

**Responses**

Configure any responses that can return information to applications, or to the SPS. Information returned to the SPS can determine how to route user requests.

**Policies**

Configure policies that bind users and groups to rules and responses.

**Note:** For complete information about how to configure SiteMinder objects, see the *CA SiteMinder Configuration Guide*.

## Authentication Scheme Considerations

SiteMinder enforces authentication schemes to protect resources. When users attempt to access protected resources through a SiteMinder Web Agent or the SPS, SiteMinder asks for credentials based on the authentication scheme protecting the resource.

SiteMinder also provides protection levels to each authentication scheme. The protection levels are enforced during single sign-on when the user tries to access resources protected by different authentication schemes. In such scenarios, the users can access resources protected by different authentication schemes without reauthentication if the protection levels for each of the authentication schemes are equal or lower. When moving from a lower protection level to a higher protection level, the user is challenged for authentication. When moving from a higher protection level to a lower protection level, the user is not challenged for reauthentication.

When the SPS is integrated with SiteMinder, the SPS behaves similar to a SiteMinder Web Agent. However, SPS using basics authentication behaves similar to a Web Agent only if the SPS is configured to use default SessionCookieScheme scheme to track user sessions. If the SPS is configured to use any of the other advanced or cookieless session schemes, the user has to reauthenticate. Single sign-on does not work.

For example, a basic authentication scheme with a protection level of 5 protects two resources, resource1 and resource2. The SPS is configured to use a mini-cookie session scheme (or any other cookieless session scheme) to track user sessions. When a user tries to access resource1, the SPS forwards the request to SiteMinder. SiteMinder verifies the authentication scheme for resource1 and challenges the user for credentials.

The SPS collects the credentials from the user and after successful authentication by SiteMinder, allows the user to access resource1. If the user then tries to access resource2, the SPS forwards the request to SiteMinder. SiteMinder verifies the authentication scheme for resource2 and challenges the user for credentials. Because the SPS is configured to use mini-cookie session scheme, the SPS requests the user to reauthenticate. If the SPS is configured to use the default SiteMinder cookie session scheme, then the user need not reauthenticate to access resource2.

**Note:** For more information about authentication schemes and their protection levels, see the *CA SiteMinder Policy Configuration Guide.*

## Proxy-Specific WebAgent.conf Settings

A number of settings in the WebAgent.conf configuration files for Web Agents installed behind the DMZ in an enterprise have specific implications for the SPS.

The settings that must be modified in the destination server WebAgent.conf files include:

**proxytrust**

As an optimization, the proxytrust directive can be set on the destination server Web Agent sitting behind the SPS. Enter one of the following settings:

**yes**

The destination server web agent automatically trusts the authorizations made by the SPS.

**no**

The destination server web agent requests authentication every time. (Default)

**proxytimeout**

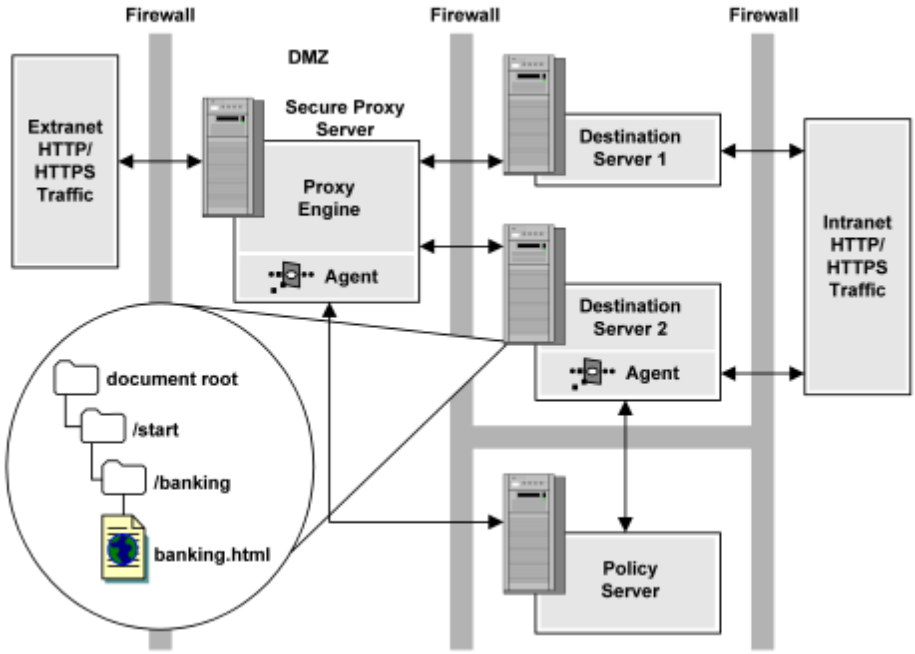
Instructs the Web Agent on the destination server to time out the single sign-on token used in requests made by the SPS. Enter a value in seconds.

**Default:** 120 seconds.

## Avoiding Policy Conflicts with Destination Server Web Agents

In some deployments, when the SPS is running in proxy trust mode, the SPS can protect resources from one set of users, while a Web Agent on a destination server protects the same resources from another set of users.

In the following illustration, Destination Server 2 has its own Web Agent. Extranet users are authenticated and authorized at the SPS, while Intranet users are authenticated and authorized through the Web Agent on the destination server. In such situations policies must exist in SiteMinder policy store for the embedded SPS Web Agent and the Web Agent on the destination server.



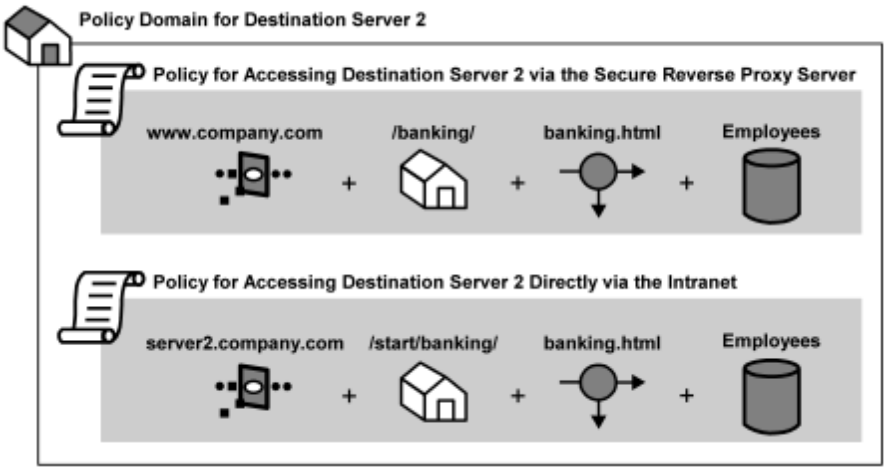

**Note:** When creating policies, administrators must be sure that the policies do not conflict with each other. If policies contradict one another, it is possible that SiteMinder may allow unwanted or unexpected behavior.

To correctly create policies and other required SiteMinder objects for the resources contained on Destination Server 2, the following objects could be created in SiteMinder:

- SPS Web Agent
- Destination server 2 Web Agent
- Realm using the SPS Web Agent
- Realm using the destination server 2 Web Agent
- Rule for resources accessed through the SPS Web Agent
- Rule for resources accessed through the destination server 2 Web Agent

- Policy for SPS web agent resources
- Policy for destination server 2 Web Agent resources

The following illustration shows how two policies must be created to protect a single resource when Compatibility Mode is used in environments that include both the SPS and Web Agents.




In the preceding illustration, the rules and realms for the same resources may have different paths based on the location of the resources on the destination server and the proxy rules used to forward requests.

For example, using the sample configuration in the preceding illustration, a resource called banking.html may be located on Destination Server 2 in the server2.company.com/start/banking/ directory, but the SPS may have proxy rules that forward all requests for www.company.com/banking/banking.html to the same destination on Server 2. Therefore, the same resource can have two different SiteMinder rules that represent the same resource. One rule authorizes access to the resource directly for employees on the intranet, and the other authorizes employees on the road who want to access the same resource from the extranet.

## Configuring SiteMinder Rules that Redirect Users

SiteMinder provides the ability to create response objects that redirect a request under certain conditions. For example, you can create a response that redirects a request to a custom error page after a failed authentication (OnRejectRedirect). By default for cookieless session schemes that rewrite a requested URL (Simple URL Rewriting), the SPS recognizes a user session information after a redirection.

To terminate a user session after a redirection, create a response attribute in SiteMinder for the relevant policy that modifies the value of the SiteMinder SM_REWRITE_URL header. This HTTP header must be set to NO to force a new session after a redirection.

For example, if you have a resource in realmA that is protected by an authentication scheme with a protection level of 5, and a second resource in realmB protected by authentication scheme with a protection level of 10, a user who successfully authenticates in realmA will be challenged for credentials when moving to realmB (due to the higher protection level).

If an OnRejectRedirect response is associated with realmB and the user fails to authenticate when challenged for credentials in realmB, the default behavior of the SPS maintains the user's original session information even after the user is redirected to a custom error page.

To terminate the user's session after the redirect and force an entirely fresh session on the next login attempt, you must create a response attribute that sets the SM_REWRITE_URL=NO, and associate the response with the appropriate policy.

# SPS and SharePoint Resources

If you want to use the SPS to secure resources managed by Microsoft SharePoint, make the configuration changes listed following.

- In the SPS Agent Configuration Object, set the following parameters:
  - SPClientIntegration = *server_name:port*

    The server name must match the value set for the ServerName field in the httpd.conf file. Most often, the ServerName is a fully qualified host name, but the value can be an IP address.
  - ProxyAgent = Yes

  **Note**: These parameters can also be added to the SPS LocalConfig.conf file.
- In the SPS WebAgent.conf file, add a LoadPlugin parameter that points to the location SharePoint plugin (SPPlugin.dll on WIndows, libSPPlugin.so on Solaris).
- In the server.conf file, add a VirtualHost element with the following parameters:

```
<VirtualHost name="VHName">
hostnames="host_name, host_name"
enableredirectwrite="yes"
redirectwritablehostnames="server1.company.com, domain1.com"
</VirtualHost>
```

**Note**: For more information, see the *CA SiteMinder Agent for SharePoint Guide*.

# SPS and ERP Resources

You can use the SPS to secure resources managed by an ERP system. The SPS can function as a proxy in front of ERP agents protecting the following ERP systems:

- Siebel Application Server
- PeopleSoft Application Server
- SAP AS Web Application Server
- SAP ITS Application Server

While the ERP agent must be installed on the ERP server, the SPS secures the ERP resources at the Policy Server.

**Note**: For information about the Policy Server settings required to support the ERP server, see the appropriate CA ERP agent guide.

**To configure the SPS as a reverse proxy for an ERP agent**

1. Specify the ERP server and appropriate port number for the <_nete:forward> element in the proxyRules.xml file.
2. Specify the following values in the server.conf file:
   - Set the value of the enableredirectrewrite parameter to "yes".
   - Set the value of the redirectrewritablehostnames parameter to the host name of the system on which the ERP server is running. For example:

```
<VirtualHost name="sales">
    hostnames="sales, sales.company.com"
    enableredirectrewrite="yes"
    redirectrewritablehostnames="server1.company.com,domain1.com"
</VirtualHost>
```

   - Set the value of the addquotestocookie parameter in the <_Sever> section of the server.conf to "no". For example:

```
addquotestocookie="no"
```

   **Note**: For information about any required settings on the ERP Server side, refer to the documentation for your ERP server.

The SPS is configured as a proxy for the ERP agent.

# Password Services for SPS

Password services are a SiteMinder feature that provides an additional layer of security to protected resources by allowing a SiteMinder administrator to manage user passwords. Password services allow an administrator to create password policies that define rules and restrictions governing password expiration, composition, and usage.

When configuring password services in SiteMinder, a password policy is associated with a directory. All users contained in the directory, or some part of the directory identified by an LDAP search expression, must adhere to the password policy. Password services are processed from inside the Apache Web server rather than from a back-end Web server hosting an agent.

**Note:** For more information about password services, see the *Policy Design Guide.*

## Configure a Password Policy for SPS

For SiteMinder to implement Password Services in a SPS deployment, the redirection URL specified in the Policy Server User Interface must refer to the SPS server, with the addition of a specific virtual directory path.

**To configure a password policy for SPS**

1. Log in to the Policy Server User Interface.
2. Select the Systems tab in the Policy Server User Interface.
3. Click the User Directories object.
4. In the User Directory List select the user directory you want to protect with Password Services.
5. Right click and select Properties of User Directory.

   The User Directory Properties dialog appears.
6. In the Credentials and Connection tab, select Require Credentials.
7. Enter the administrator's credentials, including the Username and Password.
8. In the User Attributes tab of the same dialog, enter names for the following directory user profile attributes:
   - Universal ID (example: uid)
   - Disabled Flag (example: carLicense)
   - Password Attribute (example: userPassword)
   - Password Data (example: audio)

   **Note:** For more information on the User Directory Properties dialog, see the Policy Server User Interface help.
9. Click OK.
10. In the System tab, select the Password Policies object.
11. Right click on the Password Policies object and select Create Password Policy.

    The Password Policy Properties dialog appears.
12. In the General tab, select the name of the user directory for which you made the settings for Password Services.
13. In the General tab, specify a Redirection URL as follows:

    `/siteminderagent/pw/smpwservicescgi.exe`
14. Click OK.

    The configuration is complete.

## Verify Password Services for SPS

After you have configured Password Services for the SPS, you can perform a simple test to see whether Password Services are in effect.

**To verify whether Password Services is working**

1. Select the password-protected directory from the User Directory list.
2. Select Manage User Accounts from the Tools menu.

   The User Management dialog appears.
3. Select a user.
4. Select "User must change password at next login."
5. Click OK.

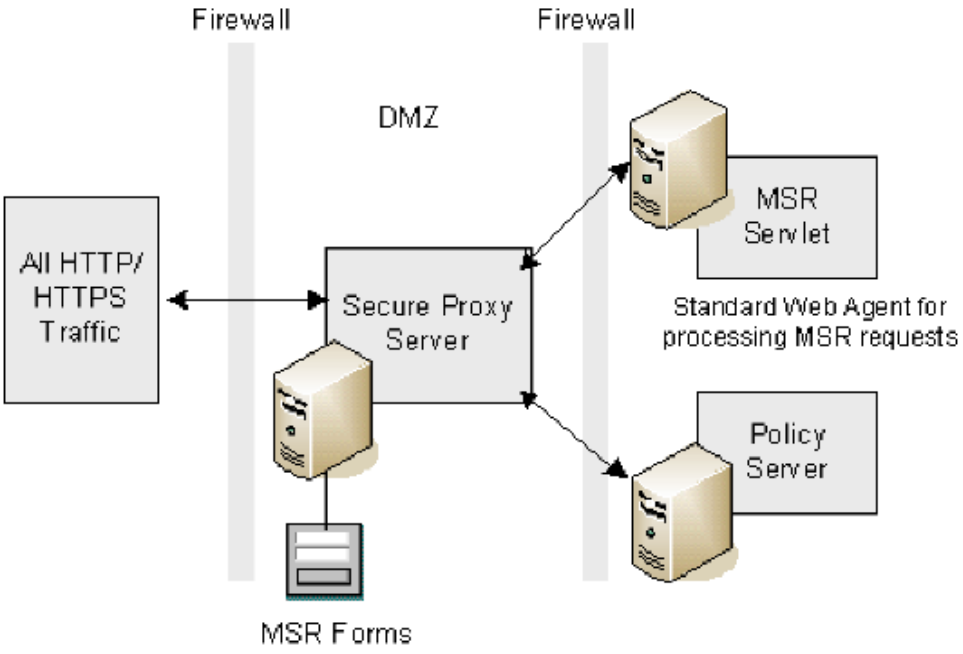
When you next request a protected page in SPS and are challenged, enter the credentials for the specified user. The Password Change screen appears, indicating the Password Services is working.

# Configuring Managed Self Registration for the SPS

Managed Self-Registration (MSR) is a SiteMinder feature that allows users to login to a web site and establish a new user account. New users can access the web site, provide personal information, and receive an account on the site. In addition, users can also define a hint that may be used to retrieve a forgotten password.

When configuring MSR in SiteMinder, a registration scheme must be configured. This registration scheme can be specified in the realm that contains the registration URL. The registration realm requires an HTML forms-based authentication scheme that points to an appropriate template for MSR services.

The following illustration shows the user of MSR with SPS.




The additional web server, which should reside outside of the DMZ , must contain a SiteMinder Web Agent installation. This web agent installation provides the processing required by MSR, and ensures that the processing of sensitive data occurs behind the DMZ.

## Install a Web Agent for MSR

To use MSR with the SPS, you must install a SiteMinder Web Agent on a Web server behind the DMZ. For information about installing a Web Agent, see the *CA SiteMinder Web Agent Installation Guide*.

After the Web Agent is installed, note the following:

- The Web Agent does not need to be enabled.
- The Web Agent must be configured as an agent object in SiteMinder.
- The Web Agent must be a version 6.x Web Agent; r12 Web Agetnts do not support MSR.

The SPS uses the MSR servlet configured for the web agent. The web agent is not used for authentication or authorization.

## Configure the Policy Server User Interface for MSR

For SiteMinder to properly implement MSR in a SPS deployment, the template path specified in the Registration Properties dialog must be defined as follows.

**To configure the policy server user interface for MSR**

1. In the SiteMinder Policy Server User Interface, select the System tab.
2. Click the Registration Schemes object.
3. Select Edit, Create Registration Scheme.

   The Registration Properties dialog opens.
4. Configure a password policy as described in the *Policy Design Guide*.
5. In the Template Path field, specify a path as follows:

   `/siteminderagent/selfreg`

## Proxy Rules for an MSR Request

The SPS supports the Managed Self Registration services through proxyrules.xml for forwarding the request to the Web Agent (6.x) hosting MSR Servlet. The forwarding is based upon the URI of the incoming request. For example, if the URI begins with /siteminderagent/selfreg, the request is forwarded to the Web Agent hosting MSR Servlet; otherwise, the request is forwarded to the backend server.

An example of a proxy rule for forwarding the password services request is following.

```
<nete:cond type="uri" criteria="beginswith">
<nete:case value="/siteminderagent/selfreg">
<nete:forward>http://MSR_server.company.com$0</nete:forward>
</nete:case>
<nete:default>
<nete:forward>http://default_backendserver.company.com$0</nete:forward>
</nete:default>
</nete:cond>
```

MSR_server.company.com stands for the server behind the DMZ on which the Web Agent hosting MSR Servlet is installed, and default_backendserver.company.com stands for destination server.

# Firewall Considerations

When configuring firewalls for the DMZ that will contain the SPS, the SPS uses ports 8007 and 8009 for internal communication. These ports should be protected from access by entities outside of the DMZ.

**Note:** You can change the ports used by the SPS by altering the appropriate directives in the server.conf file.

# Keep Alive and Connection Pooling

The SPS is designed to use a connection pool to provide better performance by spreading out the workload generated from initiating server connections. It is recommended for performance reasons that KEEP ALIVE settings should be turned on for destination servers.

All destination server products have individual methods and attributes for managing keep alive settings. These settings should be reviewed and understood when configuring SPS.

# HTTP Header Configuration for Sun Java Web Servers

By default, some web servers, such as Sun Java Web servers limit the number of header variables that can accompany a request. You might have to increase this upper limit to accommodate transactions that contain many custom headers.

The server typically returns 413 Request Entity Too Large error is # of headers is greater than allowable maximum. For more information refer to your destination server's administration guide.

**To change the maximum number of headers**

1. Locate the magnus.conf file for the back-end Sun Java Web server and open it in a text editor.
2. Add or modify the following entry in magnus.conf :

   *MaxRqHeaders 50*

   Be sure to set the maximum value at a level above the number of headers created by your SPS transactions.
3. Restart the Sun Java Web server so that the changes will take affect.

# HTTP Header for SiteMinder Processing with SPS

The SPS introduces an additional layer in the traditional SiteMinder architecture. This layer forwards or redirects all requests to destination servers in the enterprise. When the SPS processes a request, the URL requested by the user is preserved in an HTTP header variable called SM_PROXYREQUEST. This header may be used by other applications that require the original URL requested by a user before the SPS proxied the request.

The value of the ProxyAgent parameter in the Agent Configuration object must be set to YES to enable sending the SM_PROXYREQUEST HTTP header to the backend.

**Note:** This header is only added when a request is made for a protected resource.

# Handling Encoded URLs

Web servers can process both encoded and decoded normalized or unescaped URLs. How a Web server handles encoded URLs differs based on the type of server. For security reasons, and to provide consistent behavior, the SPS always decodes or normalizes a URI before processing. This provides a universal representation for a single URL, and protects against attempts to exploit the SPS using encoded strings.

# Chapter 12: Configure SPS to Support the SessionLinker

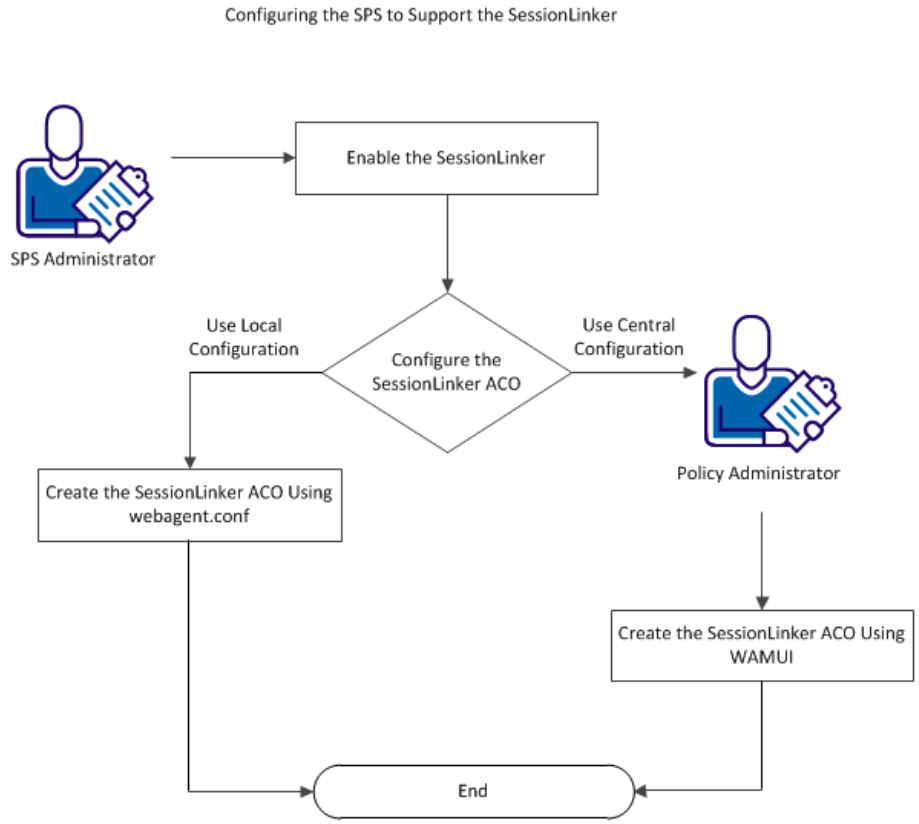
This section contains the following topics:

Configure the SPS to Support the SessionLinker (see page 191)

## Configure the SPS to Support the SessionLinker

The SessionLinker synchronizes the CA SiteMinder session with the third-party application session for better security. By default, SPS installs SessionLinker in a disabled mode.

The following diagram describes how the SPS administrator and policy administrator can configure SPS to support the SessionLinker:



**More information:**

Enable the SessionLinker (see page 194)
Create the NPS_Session_Linker ACO (see page 195)

## How the SessionLinker Works

The SessionLinker synchronizes a SiteMinder session with a third-party application session for better security. If a user logs out of SiteMinder, the SessionLinker invalidates the related session of the third-party application.

When a user authenticates, SiteMinder assigns a unique session identifier to that user session. The session identifier, called the SiteMinder Session ID, remains constant for that user for the life of the user session. If the user logs out of SiteMinder through the Logout URL, SiteMinder deletes the SMSESSION cookie that SiteMinder uses to track the SiteMinder Session ID.

The SessionLinker module takes application session cookies and associates them, one by one, with a SiteMinder session. Once associated, the application cookie (referred to here as the foreign cookie) can only be used in conjunction with that particular SiteMinder session. The SessionLinker prevents attempts by other SiteMinder sessions to use the same foreign session.

To understand the SessionLinker operation, associate the SiteMinder session and corresponding foreign cookies that SiteMinder tracks together in a table, as shown in the following example:

| SiteMinder Session ID | Foreign Cookie |
|---|---|
| ONE | ABCD |
| TWO | LMNO |
| THREE | PQRST |
| FOUR | VWXY |

The SessionLinker uses the following process:

1. The SessionLinker receives a request from a web server.
2. The SessionLinker extracts the SiteMinder Session ID from the HTTP headers and the Foreign Cookie from all the incoming HTTP cookies.
3. The SessionLinker compares the values that are presented from the web server against the contents of the table to determine whether the request must be allowed, as shown in the following examples:
   a. If the Session ID is FIVE and the Foreign Cookie is RSTU, SessionLinker inserts these values into the table.
   b. If the Session ID is SIX and the Foreign Cookie is ABCD, SessionLinker blocks the request because the Foreign Cookie ABCD is already associated with Session ONE.
   c. If the Session ID is ONE and the Foreign Cookie is HIJK, the old session is orphaned and SessionLinker updates the table to associate Session ID ONE with HIJK. When a session is orphaned, the Foreign Cookie can no longer be presented by anyone. This feature allows the SessionLinker to support applications that update the cookie during the user session.

The entire process is repeated for each Foreign Cookie. The resulting table may appear as follows:

| SiteMinder Session ID | Foreign Cookie |
|---|---|
| ***Orphaned*** | ABCD |
| ONE | HIJK |
| TWO | LMNO |
| THREE | PQRST |
| FOUR | VWXY |
| FIVE | RSTU |

## What the SessionLinker Does Not Support

The SessionLinker does *not* do any of the following tasks:

- Track cookies issued to the user throughout the CA SiteMinder environment. Doing so would require a persistent data store that could be read from and written to by every web server employing SessionLinker. The massive number of reads and writes necessary to support this tracking would require substantial processing power and bandwidth, and is thus unmanageable.

- Destroy the cookies of an existing user when the user logs out of CA SiteMinder. Because the cookies are not being tracked centrally, no mechanism knows which cookies to destroy. In addition, because of the way different web browsers handle cookies, the logout page cannot always determine which cookies the user has received. Finally, SessionLinker does not actually integrate with the CA SiteMinder logout process.
- Terminate the session of an underlying application. To support this function, the SessionLinker would need to know how to terminate sessions in each of the applications – many of which do not have an exposed API to manage sessions. Because applications can be configured to terminate sessions after some amount of idle time, and there is little the overhead in leaving a session active, this function has not been implemented.

SessionLinker accomplishes the linking by preventing the user from presenting an invalid Foreign Session cookie.

## Enable the SessionLinker

Enable the SessionLinker to synchronize the CA SiteMinder session with the third-party application session. You can configure the SessionLinker ACO after you enable the feature.

**Follow these steps:**

1. Navigate to Virtual Hosts, Virtual Hosts, Add/Edit Virtual Host, Web Agent Configuration.
2. Select the Enable Session Linker option.
3. Click Save.
4. Restart the SPS server.

## Create the NPS_Session_Linker ACO

SPS manages the SessionLinker configuration through an ACO. The SessionLinker ACO has the following syntax:

`SessionLinker=Cookie=`*`cookie_value`*`;BLOT|NOBLOT;Orphantimeout=`*`timeout_value`*`;`

Where

**COOKIE=*cookie name***

Specifies the name of the cookie holding the foreign session. If cookie names may change, use an asterisk as a wildcard character.

**BLOT|NOBLOT**

(Optional) Specifies how the SessionLinker responds to invalid sessions. If you set the value of this parameter to BLOT, the user is granted access but the foreign session cookie is not passed through the web server to the target page. If you set the value of this parameter to NOBLOT, the foreign cookie is deleted from the request and the user is then redirected to the URL specified in the URL parameter. If you do not specify the URL in the URL parameter, the internal server error is displayed.

**Default**: BLOT

**ORPHANTIMEOUT=*seconds***

Specifies the number of seconds that the SessionLinker maintains orphaned sessions.

**Default**: 86400 (24 hours)

**Limits**: Cannot be *less* than the *maximum number of seconds* that cookies from the third party (foreign) application are accepted.

To configure SPS to support SessionLinker, create the ACO named SessionLinker in *one* of the following ways:

- Using the webagent.conf file
- Using the WAMU

## Create the NPS_Session_Linker ACO Using webagent.conf

If you want to use the local configuration to create the SessionLinker ACO, create the ACO using the webagent.conf file.

**Follow these steps:**

1. Open the localconfigfile specified in the webagent.conf file.
2. Add the following command to the file:

   `SessionLinker= Cookie=`*`cookie_value;`*`BLOT|NOBLOT;Orphantimeout=`*`timeout_value`*`;`
3. Save the changes.

The SessionLinker ACO is created. The SPS is configured to support the SessionLinker.

If you want to configure SessionLinker to work with cookies, see Working with Cookies. If you want to troubleshoot any errors that are caused by the SessionLinker, see Troubleshooting.

## Create the NPS_Session_Linker ACO Using WAMUI

If you want to use the central configuration to create the SessionLinker ACO, the policy administrator must create the ACO through the WAMUI.

**Follow these steps:**

1. Open the WAMUI.
2. Add an ACO with the following details:

   **Name**: SessionLinker

   **Value**: Cookie=*cookie_name;*BLOT|NOBLOT;Orphantimeout=*timeout_value*;
3. Click Save.

The SessionLinker ACO is created. The SPS is configured to support the SessionLinker.

If you want to configure SessionLinker to work with cookies, see Working with Cookies. If you want to troubleshoot any errors that are caused by the SessionLinker, see Troubleshooting.

# Appendix A: Working with Cookies

This section contains the following topics:

Single Session Cookie Enforcement (see page 197)
Enable Wildcard Cookie Names (see page 198)
Determine Cookie Settings (see page 198)

## Single Session Cookie Enforcement

In most cases, an application has a specific name that is always used for an associated session cookie. In other cases, the name of the cookie begins with a known string, such as ASPSESSIONID or MYAPPSESSION, and ends with a random or unpredictable suffix. In such cases, the SessionLinker prevents users from presenting more than one of these cookies and enforces the expected session linking.

If the SessionLinker detects multiple potential session cookies, it performs the following steps:

1. Blocks access to sessions
2. Destroys all the cookies
3. Redirects the user to a URL that you specify. If you do not specify a URL, the internal server error is displayed.

## Enable Wildcard Cookie Names

You can add the following parameters of the ACO configured on the Policy Server to the configuration settings already selected:

**COOKIE**

Specifies that a cookies beginning with the specified name must be considered as a potential foreign session cookie. The cookie value may end in an asterisk (*). If you specify a cookie value other than a wildcard syntax, you must specify COOKIEPATH and COOKIEDOMAIN values that determine how to destroy the incoming cookies.

**COOKIEPATH**

Specifies the cookie path. If you specified a wildcard syntax for the COOKIE parameter, do not specify this parameter. The COOKIEPATH value depends on the session cookie, and has the following format:

`COOKIEPATH=<Path for outbound cookies or cookies>`

**Default Value**: /

**Example**: COOKIEPATH=/

**COOKIEDOMAIN**

Specifies the cookie domain. If you specified a wildcard syntax for the COOKIE parameter, you can specify this value in the following format:

`COOKIEDOMAIN=<domain name for outbound cookie or cookies>`

**Default Value**: Blank

**Example**: COOKIEDOMAIN=.ca.com

## Determine Cookie Settings

To determine the cookie name settings, perform the following steps:

1. Access the application multiple times.
2. Note the cookies sent by the application.
3. Enable the warning prompts for cookies in a web browser.
4. Examine the warnings that appear.

## COOKIE Setting

If the name of a session cookie for an application starts with the same text string but ends differently, specify a cookie name in the following format:

`COOKIE=cookiename*`

## COOKIEDOMAIN Setting

The domain name of a cookie consists of any of the following items:

- The domain name of the web server prefixed by a leading period (.)
- The fully qualified computer name of the web server (myserver.example.com)
- A blank

The fully qualified computer name or a blank name are equivalent.

**Note**: Internet Explorer deletes the leading period before displaying a domain. So, we recommend that you test various configurations in a staging environment to determine the correct settings.

## COOKIEPATH Setting

The path associated with a cookie is typically a directory, but could be a file or another string. Examine the path shown in the cookie warning dialog of your web browser. If the path shown is *not* a forward slash (/), enter the correct value for the COOKIEPATH setting.

## Maintain Links to Multiple Cookies

Some web applications use more than one cookie simultaneously within the same area of the site. You can configure the SessionLinker to maintain links from a single CA SiteMinder session to a number of cookies. A maximum of ten foreign session cookies can be linked to a single CA SiteMinder session.

**Follow these steps:**

1. Determine the correct configuration string for each cookie.

   **Note**: Each configuration string requires at least a COOKIE directive, but any of the directives can be combined.

2. Assign an integer from 0 through 9 to each cookie.
3. Append the selected number to the directive name.

   **Note**: You can use any number for each set of directives but the settings for a single cookie require the same number.

4. Concatenate the separate configuration strings into a single string.

## Troubleshooting

If an error occurs, consider the following possibilities to troubleshoot the error:

- Verify that the valid SMSESSION cookie and FOREIGN SESSION cookie are set at the user and are passed to the SPS.
- If you enabled the SessionLinker using the webagent.conf file, verify that the web agent is enabled.
- Verify that the SessionLinker ACO syntax is correct.
- If agent tracing is enabled in the SessionLinker ACO, verify the logs and trace messages in the agent logs and trace.
- Verify that the SPS loaded the SessionLinker plug-in binary properly. Check the agents.log file for log messages. If there are any errors, check if any dependent libraries exist for the SessionLinker plug-in library on the SPS.
- If a request is rejected, verify that the session identifiers on CA SiteMinder Policy Server (SMSESSION) and application web server (FOREIGN SESSION) are linked are the same user.

# Chapter 13: SSL and the Secure Proxy Server

This section contains the following topics:

Keys and Server Certificates Management (see page 201)
Configure SSL for CA SiteMinder SPS (see page 206)
Enable SSL for Virtual Hosts (see page 207)

## Keys and Server Certificates Management

The SPS fully supports the Secure Sockets Layer (SSL) protocol. SSL provides secure communication between client and server, enabling mutual authentication (using certificates) and private encrypted messages (using keys).

The SPS uses the OpenSSL cryptography toolkit, which implements the SSL v2/v3 and Transport Layer Security (TLS v1) network protocols and related cryptography standards required by these protocols. The OpenSSL toolkit includes the openssl command line tool for generating keys and certificates. The openssl executable image and supporting libraries are located in the <install dir>\SSL\bin folder or corresponding UNIX directory.

**Note:** To enable SSL on Solaris, you must have patch 127127-11 installed on the same system as the Secure Proxy Server.

To run the openssl command line tool, connect to the appropriate folder or directory. Open a command line Window or UNIX shell. Use the following syntax as a guideline for entering openssl commands:

**openssl** *command* [ *command_opts* ] [ *command_args* ]

The **openssl** tool provides a large number of commands (*command* in the synopsis above); each one can include numerous options and arguments (*command_opts* and *command_args* in the synopsis). You can find complete documentation for openssl at the following URL:

http://www.openssl.org/docs/apps/openssl.html

**Important**! When you issue the openssl command for any propose be sure to specify a valid path to the openssl configuration file (openssl.conf) using the -config parameter in the command line.

The commands you are most likely to use to perform fundamental SSL tasks are as follows:

- Generating a private key
- Generating a Certificate Signing Request (CSR)
- Generating a certificate by self signing the CSR
- Having a certificate signed by a Certificate Authority (CA)
- Installing a signed certificate
- Decrypting an RSA key
- Encrypting an RSA key
- Changing the password of an RSA key

Before you proceed, review the following important information about private keys and server certificates:

- The server certificate and private key work together. You must use the server certificate with the corresponding private key.
- The server certificate should be digitally signed by a Certificate Authority (CA) or self-signed with your own private key (recommended for sites intended exclusively for internal use).
- The SSLCertificateFile and SSLCertificateKeyFile directives in the SSL.conf file must point to the corresponding certificate and key files.
- If you are using Apache's virtual host feature, each virtual host you want to secure must have its own private key and server certificate.

## Generate a Private RSA Key

SSL uses keys to encrypt and decrypt messages. Keys generally come in pairs: one public key, and one private key. With OpenSSL, the private key contains the public key information, so you do not generate a public key separately.

Keys use various cryptographic algorithms and key exchange methods. For generating private keys for use with certificates, you most commonly will use the RSA key exchange method with the Date Encryption Standard (DES) cryptographic algorithm in an openssl command (on UNIX in this example) as follows:

openssl genrsa -des3 -out server.key

The key output file will be in encrypted ASCII PEM (from "Privacy Enhanced Mail") format.

Because the file is encrypted, you will be prompted for a pass-phrase to protect it and decrypt it later if you want. If you do not want your key to be protected, do not use the -des3 argument in the command line.

**Important!** Do not use the -des3 option if you are running on Windows. The Secure Proxy Server does not start if there is a prompt for a pass-phrase.

To view the details of this RSA key, enter the following command:

openssl rsa -noout -text -in server.key

## RSA Key Decryption

**To remove the encryption from a private key**

1. Make a copy of the encrypted key as a backup, for example:
   cup server.key server.key.org
2. Enter this command:
   openssl rsa -in server.key.org -out server.key

Specifying the output file without a preceding encryption option (that is, -des, -des3, or -idea), the file is written out in plain text, and there is no prompt for a pass-phrase.

**Important!** The availability of an unencrypted key on your system makes your system vulnerable to impersonation on the Internet. Make sure that this file has the appropriate permissions, that is, readable only by root on UNIX, or Administrator on Windows.

## RSA Key Encryption

To encrypt an unencrypted RSA key, enter the following command:

openssl dsa -in server.key -des3 -out server.key.new

Do not user this command on Windows, because the Web server will not start if there is a prompt for a pass-phrase.

## Modify the Passphrase for an RSA Key

To change the password on an existing RSA key, enter the following command:

openssl rsa -des3 -in server.key -out server.key.new

You are prompted for both the old pass-phrase and a new pass-phrase. You then rename your newly created key to the old key name.

## Create Certificate Signing Request

Certificates are created for authentication. They associate a public key with the identity of a user or server. Because OpenSSL uses private keys to generate public keys, the first step for creating a certificate is to generate a private key, as described in the previous sections.

The next step is to generate a certificate request, or Certificate Signing Request (CSR), using the private key. You can send the CSR to a Certificate Authority for signing into a certificate, or you can create a self-signed certificate (not recommended, except for testing or other internal use).

To create a CSR with the RSA server private key, enter the following command:

```
openssl req -config openssl.cnf -new -key server.key -out server.csr
```

You are prompted for several answers to identify the request.

**Note:** This command presupposes the existence of an openssl configuration file in the present working directory. The file is located at <install dir>\SSL\bin\openssl.cnf. If you change the name, or move it to another location, you must supply the correct location of openssl.cnf in the command line.

The CSR output file will be in ASCII PEM Privacy Enhanced Mail (PEM) format. You can specify a different format with the -outform option. See the online guide for a list of supported formats.

To view details about the CSR, use the following command:

```
openssl req -noout -text -in server.csr
```

## Create a Self-Signed Certificate

To create a certificate for testing or other internal purposes, use the following command:

```
openssl -req -new -x509 -key server.key -out cert_name.crt
```

To set an expiration time, you can use the -days flag. For example, -days 365 will force the certificate to expire in one year.

Place the output file in the following directory:

*sps_home*\SSL\certs

You must restart the SPS to enable the certificate.

## Obtain Certificate Signed by a CA

To have a certificate signed by a Certificate Authority, go to the CA's web site and complete the online submission form. You will probably also have to pay a fee. For more information about commercial CAs, you can visit one of these web sites:

- Verisign
  http://digitalid.verisign.com/server/apacheNotice.htm
- Thawte
  http://www.thawte.com/certs/server/request.html

Allow 5-10 working days for the CA to process your request.

## Install a Signed Certificate

You install a CA-signed certificate by editing the ssl.conf file. Be sure that the SSLCertificateFile and SSLCertificateKeyFile directives are pointing to the key file and the certificate file you previously created. The csr file is no longer required.

# Configure SSL for CA SiteMinder SPS

Configure CA SiteMinder SPS to support SSL.

**Follow these steps:**

1. Generate a server key with a minimum key size of 1024 KB and a FIPS-compliant algorithm.

   Example:

   ```
   openssl genrsa -des3 -out server.key 1024
   ```

2. Generate a Certificate Signing Request (CSR).

   Example:

   ```
   openssl req -config openssl.cnf -new -key server.key -out server.csr
   ```

3. Sign the certificate by a Certificate Authority (CA).
4. Install the signed certificate.
5. Open the httpd-ssl conf file.

   **Default Path**: *sps_home*\httpd\conf\extra\httpd-ssl.conf

6. Verify that the directives of the server key and certs are correct.
7. Verify that the value of the SSLPassPhraseDialog variable is custom. If not, set the value to custom.
8. Verify that the value of the SSLCustomPropertiesFile variable is <*sps_home*>\httpd\conf\spsapachessl.properties. If not, set the value.
9. Perform *one* of the following steps:
   - If you are configuring SSL on Windows, perform the following steps:
     a. Execute the following command from the command prompt:

        ```
        sps_home\httpd\bin\configssl.bat –enable passphrase
        ```

        **Note**: The passphrase value must match the passphrase value of the server key.

        The passphrase is encrypted and is stored in the spsapachessl.properties file.

     b. Restart the Secure Proxy Service.
   - If you are configuring SSL on UNIX, perform the following steps:
     a. Execute the following command:

        ```
        sps_home/secure-proxy/proxy-engine/configssl.sh passphrase
        ```

        **Note**: The passphrase value must match the passphrase value of the server key.

The passphrase is encrypted and is stored in the spsapachessl.properties file.

b. Execute the following command:

```
sps_home/secure-proxy/proxy-engine/sps-ctl startssl
```

SSL is enabled and configured.

**Note**: If you want to run SPS without SSL, execute the sps_home\httpd\bin\configssl.bat -disable command to disable SSL on Windows or execute the sps_home/secure-proxy/proxy-engine/sps-ctl start command to disable SSL on UNIX.

# Enable SSL for Virtual Hosts

The Apache server supports virtual hosts, which are multiple Web hosts that are run from a single Apache binary. Apache virtual hosts can be name-based or IP-based. Name-based virtual hosts can share a single IP address, while IP-based virtual hosts require a different IP address for each virtual host.

Apache virtual hosts using the SSL protocol:

- Must be IP-based due to limitations in the protocol.
- Must have virtual host containers in the Apache configuration file for both secure (HTTPS) and not secure (HTTP) requests.

The following is an example of a secure (HTTPS) virtual host:

```
<VirtualHost 10.0.0.1:443>
DocumentRoot ".../htdocs/site1"
ServerName www.site1.net
ServerAdmin webmaster@site1.net
ErrorLog logs/covalent_error_log_site1
TransferLog logs/...
SSLEngine on
SSLCertificateFile /www.site1.net.cert
SSLCertificateKeyFile /www.site1.net.key
CustomLog logs/cipher_log_site1 \
          "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"
</VirtualHost>
```

# Chapter 14: Configure SPS to Support Integrated Windows Authentication

This section contains the following topics:

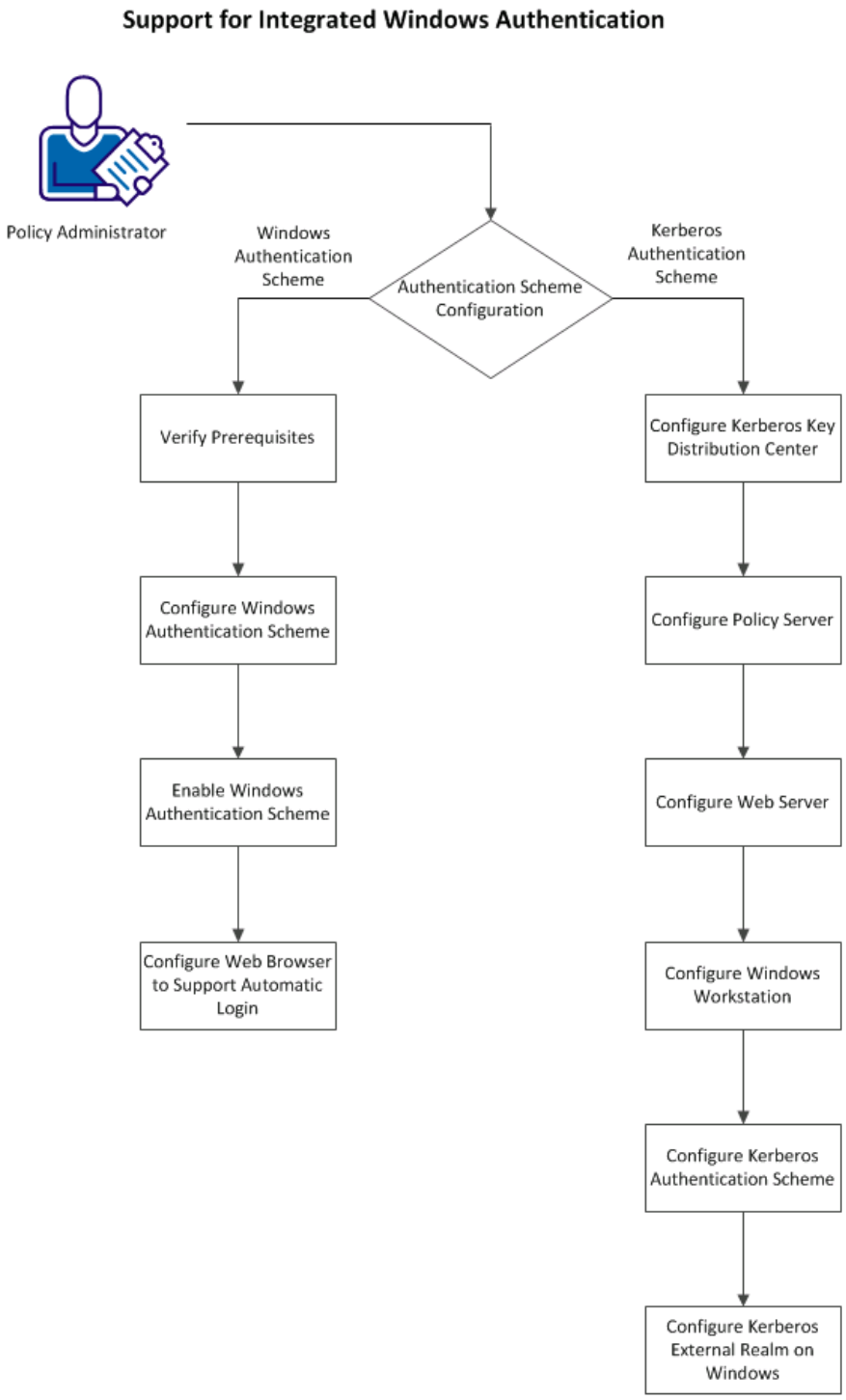
Configure SPS to Support Integrated Windows Authentication (see page 209)

## Configure SPS to Support Integrated Windows Authentication

Integrated Windows Authentication (IWA) uses the security features of Windows clients and servers. IWA enforces Single Sign-On by allowing Windows to gather user credentials during the initial interactive desktop login process and then transmitting that information to the security layer.

SiteMinder, using the Windows Authentication scheme, secures resources by processing user credentials that are obtained by the Microsoft Integrated Windows Authentication infrastructure.

The following diagram describes how you can configure SPS to support IWA:



You can configure *one* of the following authentication schemes with SPS:

- Windows authentication scheme
- Kerberos authentication scheme

**More information:**

Configure Windows Authentication (see page 211)
Configure Kerberos Authentication (see page 216)

## Windows Authentication Schemes

The Windows authentication scheme allows SiteMinder to provide access control in deployments with Active Directories running in native mode and Active Directories configured to support NTLM authentication. Windows authentication scheme uses SPNEGO to allow initiators and acceptors to negotiate with either Kerberos or NTLMSSP.

### Configure Windows Authentication

Perform the following steps to support Windows authentication:

1. Verify the prerequisites.
2. Configure Windows authentication scheme.
3. Enable Windows authentication scheme.
4. Configure web browser to support an automatic login.

### Verify the Prerequisites

Verify that you perform the following tasks before you configure SPS to support IWA:

1. Configure a Windows domain controller.
2. Add SPS host as a member of domain host for the Windows domain controller.
3. For legacy WinNT directories or Active Directory in mixed mode:
   - The user directory connection that you create in the Administrative UI specifies the WinNT namespace.
   - The requested resources can be located in any type of web server.
4. For Active Directories running in native mode:
   - User data resides in an Active Directory.
   - User directory connections must specify either an LDAP or AD namespace.

- The requested resources can be located in any type of web server.
- Client and server accounts are enabled for delegation.

## Configure a Windows Authentication Scheme

You can use a Windows authentication scheme to authenticate users in a Windows environment.

**Note**: The following procedure assumes that you are creating an object. You can also copy the properties of an existing object to create an object. For more information, see Duplicate Policy Server Objects.

**Follow these steps:**

1. Click Infrastructure, Authentication.
2. Click Authentication Schemes.

   The Authentication Schemes page appears.
3. Click Create Authentication Scheme.

   Verify that the Create a new object of type Authentication Scheme is selected.
4. Click OK

   The Create Authentication Scheme page appears.

   **Note**: Click Help for descriptions of settings and controls, including their respective requirements and limits.
5. Enter a name and protection level.
6. Select Windows Authentication Template from the Authentication Scheme Type list.

   Scheme-specific settings appear.

7. Enter Server Name, Target, and User DN information. If your environment requires NT Challenge/Response authentication, obtain the following values from the agent owner:

   **Server Name**

   The fully qualified domain name of SPS, for example:

   server1.myorg.com

   **Target**

   /siteminderagent/ntlm/smntlm.ntc

   **Note**: The directory must correspond to the virtual directory already configured by the installation. The target file, smntlm.ntc, does not need to exist and can be any name that ends in .ntc or the custom MIME type that you use in place of the default.

   **Library**

   smauthntlm

8. Click Submit.

   The authentication scheme is saved and can be assigned to a realm.

## Support for the Integrated Windows Authentication

The SPS now supports Windows authentication scheme that is configured on the Policy Server. To let the SPS support the Windows authentication scheme, perform the following steps:

1. Create an ACO WindowsNativeAuthentication in the Policy Server.
2. Set the value of WindowsNativeAuthentication to no.

**Note**: If you do not define or set the value of the WindowsNativeAuthentication ACO, the SPS does not support the Windows authentication.

### Configure Web Browser to Support Automatic Login

To configure automatic logon in Internet Explorer 5.x and 6.x Browsers, perform the following steps:

1. From the menu bar in Internet Explorer, select Tools, Internet Options.

   The Internet Options dialog opens.
2. Click the Security tab to bring it to the front.
3. Select your Internet zone and click Custom Level.

   The Security Settings dialog opens.
4. Scroll down to User Authentication, Logon.
5. Select the Automatic logon with current username and password radio button.
6. Click OK.

To configure automatic logon in Internet Explorer 4.x Browsers, perform the following steps:

1. From the menu bar in Internet Explorer, select View, Internet Options.

   The Internet Options dialog opens.
2. Click the Security tab to bring it to the front.
3. Select your Internet zone from the drop down list.
4. In the Internet zone group box, select the and click Custom radio button and click Settings.

   The Security Settings dialog opens.
5. Scroll down to User Authentication, Logon.
6. Select the Automatic logon with current username and password radio button.
7. Click OK.

SPS is configured to support Windows authentication

## Kerberos Authentication Schemes

Kerberos is a standard protocol, designed at MIT, to provide a means of authentication between a client and a server on an open network. The Kerberos protocol protects messages from eavesdropping and replay attacks. Kerberos uses shared secrets, symmetric keys, and Kerberos services. Microsoft Windows operating environments use Kerberos V5 as the default authentication package. Solaris 10 also includes Kerberos V5.

In a Kerberos environment, user accounts and service accounts are named principals. Kerberos uses a trusted third party (the Key Distribution Center, or KDC) to mediate message exchanges between principals. The purpose of the Key Distribution Center is to reduce the risks inherent in exchanging keys.

Kerberos authentication is based on messages that request and deliver tickets. The Key Distribution Center processes two types of tickets:

- Ticket-Granting Ticket (TGT) — used internally by KDC to transport a requestor's credentials to the ticket-granting service (TGS).
- Session Ticket — used by the ticket-granting service (TGS) to transport the requestor's credentials to the target server or service.

Kerberos uses keytab files for logging in to KDC. Keytab files consist of pairs of Kerberos principals and encrypted keys derived from a Kerberos password.

The Kerberos protocol message exchange can be summarized in a simplified way as follows:

1. When a user logs in, the client contacts KDC Authentication Service, requesting a short-lived message (the ticket-granting ticket) containing the user identity information.
2. KDC authentication service generates the TGT and creates a session key that the client can use to encrypt communication with the ticket-granting service.
3. When a user requests access to local or network resources, the client presents the ticket-granting ticket (TGT), an authenticator, and the Service Principal Name (SPN) of the target server to KDC.
4. The ticket-granting service examines the ticket-granting ticket and the authenticator. If these credentials are acceptable, the ticket-granting service creates a service ticket, which includes the user identity information copied from the TGT. The service ticket is sent back to the client.

   **Note**: The ticket-granting service cannot determine whether the user is granted access to the target resource. The ticket-granting service only authenticates the user and returns the session ticket.
5. After the client has the session ticket, the client sends the session ticket and a new authenticator to the target server, requesting access to a resource.
6. The server decrypts the ticket, validates the authenticator, and grants the user access to the resource.

## Configure Kerberos Authentication

Perform the following steps to configure Kerberos Authentication:

1. Configure Kerberos Key Distribution Center (KDC).
2. Configure Policy Server.
3. Configure Web Server.
4. Configure Kerberos authentication scheme.
5. Configure a Kerberos External Realm on Windows.

## Configure Kerberos Key Distribution Center

When using Kerberos, the domain controller is the Kerberos Key Distribution Center (KDC) for the Kerberos Realm. In a pure Windows environment, a Kerberos Realm is equivalent to a Windows Domain. The domain controller host provides storage for the user, service accounts, credentials, the Kerberos ticketing services, and Windows Domain services.

Kerberos authentication requires a keytab file, which lets users authenticate with KDC without being prompted for a password. On Windows, use the ktpass command tool utility to create the keytab file and on UNIX, use the ktadd utility creates the keytab file.

Perform the following steps to configure KDC:

1. Create a user account to log in to the workstation.
2. Create a service account for the web server to log in to the web server host.
3. Create a service account for the Policy Server to log in to the Policy Server host.
4. Associate the web server account with a web server principal name.
5. Create a keytab file, which is transferred to the web server host.
6. Associate the Policy Server account with a Policy Server principal name.
7. Create another keytab file and transfer the new keytab file to the Policy Server host.
8. Verify that the web server and Policy Server accounts are Trusted for Delegation.

**Important!** For any service to use Kerberos protocol, ensure that you create the Service Principal Name (SPN) in the service/fqdn_host@REALM_NAME format.

## Configure Policy Server

Perform the following steps in addition to the standard Policy Server configuration:

1. Open the ACO of the agent you want to configure and perform the following steps:
   a. Add the value .kcc to the KCCExt parameter.
   b. Add the value web server principal name to the HttpServicePrincipal parameter.

      Example: HTTP/win2k8sps.test.com@TEST.COM
   c. Add the Policy Server principal name to the SmpsServicePrincipal parameter.

      Example: smps@winps.test.com
2. Configure a Kerberos configuration file, krb5.ini and perform one of the following steps:
   - On Windows, place the krb5.ini file in the system root path.
   - On UNIX, place the krb5.ini file in the /etc/krb5/ path.
3. Deploy the keytab file created on KDC that contains the Policy Server principal credentials to a secure location on the Policy Server.

**Important**! If the Policy Server is installed on Windows and KDC is deployed on UNIX, ensure that you perform the additional configuration on the Policy Server host using the Ksetup utility.

## Configure Web Server

Perform the following steps to configure a web server:

1. Install a SiteMinder Web Agent with SiteMinder Kerberos Authentication Scheme support.
2. Register a trusted host with the Policy Server and configure the Web Agent.
3. Configure a Kerberos configuration file, krb5.ini, and perform the following steps:
   a. Configure KDC for the Kerberos realm (domain).
   b. Configure krb5.ini to use the keytab file containing the credentials of the web server principal.
   c. Place krb5.ini in the system root path on Windows and in /etc/krb5/ on UNIX.
4. Deploy the keytab file created on KDC that contains the web server credentials to a secure location on the web server.

**Important!** If the web server is installed on Windows and KDC is deployed on UNIX, ensure that you perform additional configuration on the web server using the Ksetup utility.

## Configure Windows Workstation

Perform the following steps to configure the Windows workstation:

**Important!** If KDC is deployed on UNIX, be sure to perform the additional required configuration on the workstation using Ksetup utility.

1. Add the host for the Windows workstation to KDC domain.
2. Log in to the host using user account created on KDC.
3. Configure Internet Explorer to pass credentials automatically:
    a. Initiate an instance of the IE web browser.
    b. Select the Internet options menu.
    c. Select the Security tab.
    d. Select Local intranet tab.
    e. Click Sites and select all three checkboxes.
    f. Select the Advanced tab and add http://*.*domain.com* to local intranet zone.
    g. Select the Custom level tab under security settings and select Automatic logon only in intranet zone under the User Authentication tab.
    h. Select the Advanced tab from Internet options and select the Enable Integrated Windows authentication (requires restart) option.
    i. Close the browser.

## Configure a Kerberos Authentication Scheme

A custom authentication scheme is required to support Kerberos authentication in the SIteMinder environment. Associate this authentication scheme with any realm whose protected resources use Kerberos authentication.

**Follow these steps:**

1. Log in to the Administrative UI.

    **Note:** When you create or modify a Policy Server object in the [set the ufi variable for your book], use ASCII characters. Object creation or modification with non-ASCII characters is not supported.
2. Select Infrastructure, Authentication, Authentication Schemes.
3. Click Create Authentication Scheme.
4. Select Custom Template from the Authentication Scheme Type list.

    Custom Template settings appear.
5. Enter **smauthkerberos** in the Library field.

6. Enter the following values in the Parameter field. Enter the values in the order in the following list, delimited by semicolons:

   a. The name of the host web server and target fields

   b. The Policy Server principal name from the Kerberos domain

   c. The mapping between user principal and the user store search filter

   **LDAP Example 1:**
   http://win2k8sps.test.com/siteminderagent/Kerberos/creds.kcc;smps/winps.test.com@TEST.COM;(uid=%{UID})

   **LDAP Example 2:**
   http:/win2k8sps.test.com/siteminderagent/Kerberos/creds.kcc;smps/winps.test.com@TEST.COM;(uid=%{UID})

   **AD Example 1:**
   http://win2k8sps.test.com/siteminderagent/Kerberos/creds.kcc;smps/winps.test.com@TEST.COM;(cn=%{UID})

   **AD Example 2:**
   http://win2k8sps.test.com/siteminderagent/Kerberos/creds.kcc;smps/winps.test.com@TEST.COM;(cn=%{UID})

   **ODBC Example 1:**
   http://win2k8sps.test.com/siteminderagent/Kerberos/creds.kcc;smps/winps.test.com@TEST.COM;%{UID}

   **ODBC Example 2**:
   http://win2k8sps.test.com/siteminderagent/Kerberos/creds.kcc;smps/winps.test.com@TEST.COM;%{UID}

7. Click OK.

   The Kerberos Authentication scheme is saved and appears in the Authentication Scheme List.

## Configure a Kerberos External Realm on Windows

For the Windows workstation to use a Kerberos KDC deployed on UNIX, configure both the Kerberos KDC server and the workstation.

In the Kerberos realm, create a host principal for the Windows host. Use the following command:

```
kadmin.local: addprinc host/machine-name.dns-domain_name.
```

For example, if the Windows workstation name is W2KW and the Kerberos realm name is EXAMPLE.COM, the principal name is host/w2kw.example.com.

A Kerberos realm is not a Windows domain, perform the following procedure to configure KDC operating environment as a member of a workgroup:

1. Remove the host from the Windows domain.
2. Add the test user, for example, testkrb, to the local user database.
3. Add the Kerberos Realm:

   ```
   ksetup /SetRealm EXAMPLE.COM
   ```

4. Restart the host.
5. Add KDC:

   ```
   ksetup /addkdc EXAMPLE.COM rhasmit
   ```

6. Set a new password:

   ```
   ksetup /setmachpassword password
   ```

   **Note**: The password used here is same as the one used while creating the host principal account in the MIT KDC.

7. Restart the host.

   **Note**: Whenever changes are made to the external KDC and realm configuration, a restart is required.

8. Set the Realm Flag

   ```
   ksetup /SetRealmFlags EXAMPLE.COM delegate
   ```

9. Run AddKpasswd:

   ```
   ksetup /AddKpasswd EXAMPLE.COM rhasmit
   ```

10. Use Ksetup to configure single sign on to local workstation accounts by defining the account mappings between the Windows host accounts to Kerberos principals. For example:

    ```
    ksetup /mapuser testkrb@EXAMPLE.COM testkrb
    ksetup /mapuser * *
    ```

    The second command maps clients to local accounts of the same name. Use Ksetup with no arguments to see the current settings.

SPS is configured to support Kerberos authentication.

## Kerberos Configuration Examples

The configurations that follow include examples of the specifics, such as keytab file creation, required to implement Kerberos authentication in a SiteMinder environment. Note that additional configuration is required when KDC is deployed in a UNIX operating environment and the Policy Server, or web server, or workstation is in a Windows operating environment.

## KDC Configuration on Windows 2008 Example

The steps listed following exemplify how to configure a Windows domain controller to support SiteMinder Kerberos authentication.

1. Promote a Windows Server to a domain controller (named test.com in this example) using Windows dcpromo utility.
2. Raise the domain functional level:
    1. Open the Active Directory users and computers dialog from Administrative tools.
    2. Right-click the test.com drop-down on the left side of dialog.
    3. Click Raise domain functional level.
    4. Raise the domain functional level of Active directory.

       **Important!** This step is irreversible.
3. Create a user account (for example, testkrb). Provide a password for this account. Clear the option, User must change password at next logon. Add this account to the domain administrators group so that the user has permissions to login. The Windows workstation uses this account to log in to test.com.
4. Create a service account for the web server (for example, wasrvwin2k8sps). Create a password for this account. Clear the option, User must change password at next logon. Add this account to the domain administrators group so that the user has permissions to login. SPS uses this account to log in to test.com.
5. Create a service account for the Policy Server (polsrvwinps). Provide a password for this account. Clear the option, User must change password at next logon. Add this account to the domain administrators group so that the user has permissions to login. The Policy Server host (winps) uses this account to log in to test.com.
6. Join the web server (win2k8sps) and the Policy Server (winps) hosts to the test.com domain using their service accounts created in Steps 4 and 5.
7. Associate the web server account (wasrvwin2k8sps) with a web server principal name (HTTP/win2k8sps.test.com@TEST.COM) and create a keytab file using the Ktpass utility. The syntax differs depending on whether the Policy Server is on Windows or on UNIX.

   **Note**: The Ktpass command tool utility is a Windows support tool. You can install it from MSDN download or an installation CD. Always verify the version of support tools. The default encryption type must always be RC4-HMAC. The encryption type can be confirmed by running ktpass /? at the command prompt.

When the Policy Server is on Windows:

```
ktpass -out c:\wasrvwin2k8sps.keytab -princ HTTP/win2k8sps.test.com@TEST.COM
-ptype KRB5_NT_PRINCIPAL -mapuser wasrvwin2k8sps -pass <<password>>

Targeting domain controller: winkdc.Test.com
Using legacy password setting method
Successfully mapped HTTP/win2k8sps.test.com to wasrvwin2k8sps.
Key created.
Output keytab to c:\wasrvwin2k8sps.keytab:
Keytab version: 0x502
keysize 67 HTTP/win2k8sps.test.com@TEST.COM ptype 1 (KRB5_NT_PRINCIPAL) vno 2
etype 0x17 (RC4-HMAC) keylength 16 (0xfd77a26f1f5d61d1fafd67a2d88784c7)
```

The password is the same as the one used for creating the service account for the web server.

When the Policy Server is on UNIX:

```
ktpass -out d:\sol10sunone_host.keytab -princ
host/sol10sunone.test.com@TEST.COM -pass <<password>> -mapuser sol10sunone
–crypto DES-CBC-MD5 +DesOnly -ptype KRB5_NT_PRINCIPAL –kvno 3

Targeting domain controller: winkdc.test.com
Successfully mapped host/sol10sunone.test.com to sol10sunone.
Key created.
Output keytab to d:\sol10sunone_host.keytab:
Keytab version: 0x502
keysize 52 host/sol10sunone.test.com@TEST ptype 1 (KRB5_NT_PRINCIPAL) vno 3 etype
0x3 (DES-CBC-MD5) keylength 8 (0xb5a87ab5070e7f4a)
Account sol10sunone has been set for DES-only encryption.
```

8. Associate the Policy Server account (polsrvwinps) with a Policy Server principal name (smps/winps.test.com@TEST.COM) and create another keytab file destined for the Policy Server host (winps).

   When the Policy Server is on Windows:

   ```
   Ktpass -out c:\polsrvwinps.keytab –princ smps/winps.test.com@TEST.COM -ptype
   KRB5_NT_PRINCIPAL -mapuser polsrvwinps -pass <<password>>
   Targeting domain controller: winkdc.Test.com
   Using legacy password setting method
   Successfully mapped smps/winps.test.com to polsrvwinps.
   Key created.
   Output keytab to c:\polsrvwinps.keytab:
   Keytab version: 0x502
   keysize 72 smps/winps.test.com@TEST.COM ptype 1 (KRB5_NT_PRINCIPAL) vno 2 etype
   0x17 (RC4-HMAC) keylength 16 (0xfd77a26f1f5d61d1fafd67a2d88784c7)
   ```

   The password is same as the one used for creating the service account for Policy Server.

   When the Policy Server is on UNIX:

   ```
   ktpass -out d:\sol10polsrv.keytab -princ host/sol10sunone.test.com@TEST.COM
   -pass <<password>> -mapuser sol10sunone –crypto DES-CBC-MD5 +DesOnly -ptype
   KRB5_NT_PRINCIPAL –kvno 3

   Targeting domain controller: winkdc.test.com
   Successfully mapped host/sol10sunone.test.com to sol10sunone.
   Key created.
   Output keytab to d:\sol10polserv.keytab:
   Keytab version: 0x502
   keysize 52 host/sol10sunone.test.com@TEST ptype 1 (KRB5_NT_PRINCIPAL) vno 3 etype
   0x3 (DES-CBC-MD5) keylength 8 (0xb5a87ab5070e7f4a)
   Account sol10sunone has been set for DES-only encryption.
   ```

9. Specify that the web server and Policy Server service accounts are Trusted for Delegation as follows:

   1. Right-click the service account (polsrvwinps/wasrvwin2k8sps) properties.
   2. Select the Delegation tab.
   3. Select the second option, Trust this user for delegation to any service (Kerberos only)

      Or, select the third option, Trust this user for delegation to specified service. Select the Use Kerberos only option button, and add the corresponding service principal name.

The domain controller is ready for SiteMinder Kerberos authentication.

## KDC Configuration on UNIX Example

The process listed following exemplifies how to configure a KDC Kerberos Realm on a UNIX host to support SiteMinder Kerberos authentication.

1. Install MIT Kerberos, if necessary.
2. Use the kdb5_util command to create the Kerberos database and an optional stash file. The stash file is used to authenticate KDC to itself automatically before starting the kadmind and krb5kdc daemons as part of the host auto-boot sequence.

   Both the stash file and the keytab file are potential point-of-entry for a break-in. If you install a stash file, it must be readable only by root, must not be backed up, and must exist only on KDC local disk. If you do not want a stash file, run the kdb5_util without the -s option.

   This example generates the following five database files in the directory specified in kdc.conf file:

   - Two Kerberos database files: principal.db and principal.ok
   - One Kerberos administrative database file: principal.kadm5
   - One administrative database lock file: principal.kadm5.lock
   - One stash file: .k5stash

   ```
   [root@rhasmit init.d]# kdb5_util create -r EXAMPLE.COM –s
   Initializing database '/var/kerberos/krb5kdc/principal' for realm
   'EXAMPLE.COM',
   master key name 'K/M@EXAMPLE.COM'
   You will be prompted for the database Master Password.
   It is important that you NOT FORGET this password.
   Enter KDC database master key:
   Re-enter KDC database master key to verify:
   ```
3. Create a user principal (testkrb).
4. Create a user principal (for example, testwakrb), a host principal (host/win2k8sps.example.com@EXAMPLE.COM, and a service principal (HTTP/win2k8sps.example.com@EXAMPLE.COM) for the web server host. The password used for creating host account must be same as the password specified when using the ksetup utility on the web server host.
5. Create a user principal (testpskrb), host principal (host/winps.example.com@EXAMPLE.COM) and service principal (smps/winps.example.com@EXAMPLE.COM) for the Policy Server host. The password used for creating host account must be same as the password specified when using the ksetup utility on the Policy Server host.

6. Create a keytab file for the web server service principal as follows:

   ```
   ktadd -k /tmp/win2k8sps.keytab HTTP/win2k8sps.example.com
   ```

7. Create keytab for Policy Server service principal as follows:

   ```
   ktadd -k /tmp/winps.keytab smps/winps.example.com
   ```

The Kerberos Realm is configured for SiteMInder on a UNIX host.

## Kerberos Configuration at the Policy Server on UNIX Example

The following procedure shows an example of how to configure a Policy Server on a UNIX host to support CA SiteMinder Kerberos authentication.

**Follow these steps:**

1. Create a user, for example, sol10psuser, with the same password used for creating a service account for the Policy Server host (sol10ps) in Active Directory.
2. Add the host to the test.com domain and login to host with user sol10psuser.
3. Install and configure CA SiteMinder Policy Server.
4. Install and configure policy store directory services.
5. Add a Host Configuration Object referencing the Solaris Policy Server.
6. Add an Agent Configuration Object and add the following three new parameters:

| Parameter | Value |
|---|---|
| KCCExt | .kcc |
| HttpServicePrincipal | Specify the web server principal name.<br>**Example:** HTTP/win2k8sps.test.com@TEST.COM |
| SmpsServicePrincipal | Specify the Policy Server principal name.<br>**Example:** smps@winps.test.com |

7. Create a user directory.
8. Create a user, for example, testkrb, in the user directory.

9. Configure a new Authentication Scheme using the CA SiteMinder Admin UI:
    1. Create the scheme using the custom template.
    2. Specify the CA SiteMinder Kerberos Authentication Scheme library.
    3. Select the parameter field and specify the following three semicolon-delimited values in the specified order:
        - Server name and target fields.
        - Policy Server principal name from the Windows 2003 Kerberos realm.
        - Mapping between the user principal and an LDAP search filter.

        Sample parameter field:

        ```
        http://sol10sunone.test.com/siteminderagent/Kerberos/creds.kcc;smps/sol10ps.test.com@TEST.COM;(uid=%{UID})
        ```
10. Configure a policy domain.
11. Add a realm to protect a resource using the Authentication Scheme.
12. Add Rules and Policies to allow access for the user, testkrb.
13. Configure a Kerberos configuration file (krb5.ini) and place krb5.ini in the /etc/krb5 system path.
    - Configure KDC for the Windows 2003 Kerberos realm (domain) to use the Windows 2003 domain controller.
    - Configure krb5.ini to use the Windows 2003 KDC keytab file containing the Policy Server principal credentials.

      See the following sample krb5.ini:

```
[libdefaults]
 ticket_lifetime = 24000
 default_realm=TEST.COM
 default_tgs_enctypes = des-cbc-md5
 default_tkt_enctypes = des-cbc-md5
 default_keytab_name = FILE:/etc/krb5.keytab
 dns_lookup_realm = false
 dns_lookup_kdc = false
 forwardable = true
 proxiable = true
[realms]
 TEST.COM = {
  kdc = winkdc.test.com:88
  admin_server = winkdc.test.com:749
  default_domain = test.com
 }
[domain_realm]
 .test.com=TEST.COM
  test.com=TEST.COM
```

14. Use the ktutil utility to merge the keytab files (sol10ps_smps.keytab & sol10ps_host.keytab) containing the host principal and service principal names for the Policy Server host in the /etc/krb5.keytab file:

```
ktutil: rkt sol10ps_host.keytab
ktutil: wkt /etc/krb5.keytab
ktutil: q
ktutil: rkt sol10ps_smps.keytab
ktutil: wkt /etc/krb5.keytab
ktutil: q
```

15. Verify the created krb5.keytab as follows:

```
klist -k
Keytab name: FILE:/etc/krb5.keytab
KVNO Principal
----
--------------------------------------------------------------------------
   3 host/sol10ps.test.com@TEST.COM
   3 smps/sol10ps.test.com@TEST.COM
```

16. Deploy the Windows 2003 KDC keytab file containing the host and Policy Server principal credentials to a secure location on the Policy Server.

17. Verify that the following environment variable is set before starting the Policy Server:

```
KRB5_CONFIG=/etc/krb5/krb5.conf
```

The Policy Server on a UNIX host is configured for Kerberos authentication.

## Kerberos Configuration at the Policy Server on Windows Example

The following procedure shows an example of how to configure a Policy Server on Windows to support CA SiteMinder Kerberos authentication.

**Note**: If the Policy Server is installed on Windows and KDC is deployed on UNIX, be sure to perform additional required configuration on the Policy Server host using the Ksetup utility.

**Follow these steps:**

1. Install and configure the CA SiteMinder Policy Server.
2. Install and configure policy store directory services.
3. Log in to the Policy Server host with the service account (for example, polsrvwinps) created in Active Directory on the Windows domain controller.
4. Add a Host Configuration Object referencing the Policy Server.

5. Create an Agent Configuration Object and add these three new parameters:

| Parameter | Value |
|---|---|
| KCCExt | .kcc |
| HttpServicePrincipal | Specifies the web server principal name.<br>**Example:** HTTP/win2k8sps.test.com@TEST.COM |
| SmpsServicePrincipal | Specifies the Policy Server principal name.<br>**Example:** smps@winps.test.com |

6. Create a user directory.
7. Create a user, for example, testkrb, in the user directory.
8. Configure a new Authentication Scheme using the CA SiteMinder Admin UI:
    1. Create the scheme using the custom template.
    2. Specify the CA SiteMinder Kerberos Authentication Scheme library.
    3. Select the parameter field and specify the following three semicolon-delimited values in the specified order:
        - Server name and target fields.
        - Policy Server principal name from the Windows 2003 Kerberos realm.
        - Mapping between the user principal and an LDAP search filter.

        Sample parameter field:

        ```
        http://win2k8sps.test.com/siteminderagent/Kerberos/creds.kcc;smps/winps.test.com@TES.COM;(uid=%{UID})
        ```
9. Configure a policy domain.
10. Add a realm to protect a resource using the Authentication Scheme.
11. Add Rules and Policies to allow access for the user, testkrb.
12. Configure a Kerberos configuration file (krb5.ini) and place krb5.ini in the Windows system root path:
    - Configure KDC for the Windows 2003 Kerberos realm (domain) to use the Windows 2003 domain controller.
    - Configure krb5.ini to use the Windows 2003 KDC keytab file containing the Policy Server principal credentials.

See the following sample krb5.ini:

```
[libdefaults]
default_realm = TEST.COM
default_keytab_name = C:\WINDOWS\krb5.keytab
default_tkt_enctypes = rc4-hmac des-cbc-md5
default_tgs_enctypes = rc4-hmac des-cbc-md5
[realms]
TEST.COM = {
kdc = winkdc.test.com:88
default_domain = test.com
}
[domain_realm]
.test.com = TEST.COM
```

13. Deploy the Windows KDC keytab file containing the Policy Server principal credentials to a secure location on the Policy Server.

The Policy Server on a Windows host is configured for Kerberos authentication.

# Chapter 15: Data Monitoring Using CA Wily Introscope

This section contains the following topics:

Data Monitoring Using CA Wily Introscope (see page 231)
Monitor Web Agents with the OneView Monitor (see page 233)

## Data Monitoring Using CA Wily Introscope

If you are already using CA Wily Introscope in your organization, you can monitor the health of the SPS. You can use the Wily EPAgent to monitor the following statistics of a Tomcat server:

- The average response time of each of the following SPS component:
  - Session Discovery
  - Java Web Agent
  - Post Agent Session Writer
  - Proxy Rules Filter
  - Noodle Servlet
  - HTTP Client
- The average response time of each backend server
- The SPS request wait time
- Number of hits for each proxy rule
- The SPS Agent framework instances health data

The data monitoring section has the following format:

```
<Server>
.
.
monitor_data_buffer_size="1000"
.
.
</Server>
    <metric-reporter name="Wily Metric Reporter">
    enabled="yes"
    class="com.ca.proxy.monitor.wily.WilyMetricReporter"
    endpoint="tcp://localhost:8886"
</metric-reporter>
```

***monitor_data_buffer_size***

Specifies the size of the buffer the SPS maintains to store the collected statistics before sending them/statistics to Wily.

**Default Value**: 1000

***metric-reporter name***

Specifies the name of the metric reporter. You can use this name to debug any metric errors.

***enabled***

Specifies the state of the data monitoring feature. If you want to monitor the health of SPS, set the value to yes. If you do not want to monitor the health of the SPS, set the value to no.

**Default Value**: no

***endpoint***

Specifies the configuration details of the EPAgent. Start the EPAgent before you communicate with the SPS. The endpoint parameter has the following format:

*protocol*://*hostname_of_EPAgent*:*port*

***protocol***

Specifies the communication protocol that the EPAgent uses.

***hostname_EPAgent***

Specifies the hostname of the machine where the EPAgent is installed.

**Default Value**: localhost

***port***

Specifies the port number the EPAgent uses to communicate with the SPS. If you use TCP protocol, enter the network data port that is configured in EPAgent. If you use HTTP protocol, enter the HTTP port number that is configured in EPAgent.

## Enable Data Monitoring

Perform the following steps to enable data monitoring:

**Note**: For information about configuring the CA Wily EPAgent, see the *CA Wily Introscope Environment Performance Agent Guide*.

1. Configure the CA Wily EPAgent to collect metrics from the SPS.
2. Configure the server.conf file by performing the following steps:
   a. Open the server.conf file and navigate to the metric-reporter section.
   b. Set the following value:

      ```
      enabled=yes
      ```

   c. (Optional) To monitor the agent instance metrics configured with SPS, set the following value:

      ```
      enablemonitoring=yes
      ```

   d. Save the changes.
3. Configure the ACO for each Web Agent that is configured with the SPS.
4. Start the CA Wily EPAgent.
5. Restart the SPS.

# Monitor Web Agents with the OneView Monitor

The CA SiteMinder OneView monitor reports cache statistics and other information to the Policy Server, which administrators can use to analyze and fine-tune the Web Agent. You control the CA SiteMinder OneView monitor with the following parameter:

**EnableMonitoring**

Specifies whether the agent sends monitoring information to the Policy Server.

**Default:** Yes.

To have the Web Agent use the CA SiteMinder OneView Monitor, set the EnableMonitoring parameter to yes.

**Note:** For more information, see the Policy Server documentation.

# Chapter 16: Operating System Tuning

This section contains the following topics:

Tune the Shared Memory Segments (see page 235)
How to Tune the Solaris 10 Resource Controls (see page 237)

## Tune the Shared Memory Segments

If you install an Apache-based agent on Solaris systems, tune the shared memory settings of the operating environment for the Agent to function correctly. By increasing the shared memory segments or your operating environment, you improve the performance of the Agent. The variables that control shared memory segments are defined in the specification file of your operating environment.

**Note:** Sometimes Linux operating environments require tuning the shared memory segments. For more information about the shared memory segments and how to tune them, see the documentation for your particular operating environment.

**Follow these steps:**

1. Follow the appropriate procedure for your operating environment:
   - Solaris**:** Open the /etc/system file, using the editor of your choice.
2. Modify the shared memory variables using *one* of the following methods:
   - Solaris: Add the variables shown in the following list and configure them using the recommended settings shown in the examples. Use the following syntax:

     ```
     set shmsys:shminfo_shmmax=33554432
     ```

     **shmsys:shminfo_shmmax**

     Specifies the maximum shared memory segment size. Controls the maximum size of the Agent resource and session cache.

     **Note:** To estimate the amount of memory segments that are required, allocate 4 KBs per entry in each cache, or view cache usage statistics in the OneView Monitor. See the *Web Agent Configuration* Guide for more information about using the OneView Monitor.

     **Example:** 33554432 (32 MB) for busy sites that require large caches.

     **shmsys:shminfo_shmmin**

     (Not required for Solaris) Minimum shared memory segment size. Controls the minimum size of the Agent resource and session cache.

**shmsys:shminfo_shmmni**

Specifies the maximum number of shared memory segments that can exist simultaneously, systemwide.

**Example:** (except Solaris 9) N/A

**Example:** (Solaris 9) 200

**shmsys:shminfo_shmseg**

(Not required for Solaris 9) Specifies the maximum number of shared memory segments per process.

**Example:** 24

**semsys:seminfo_semmni**

Specifies the number of semaphore identifiers. Use 11 for every instance of the Agent that you run on the system.

**Example:** (except Solaris 9) 100

**Example:** (Solaris 9) 200

**semsys:seminfo_semmns**

Specifies the number of semaphores in the system. Use 10 for every instance of the Agent that you run on the system.

**Example:** (Solaris 9) 100

**Example:** (Solaris 9) 400

**semsys:seminfo_semmnu**

Specifies the number of processes using the undo facility. For optimal performance, set the semmnu value so it exceeds the number of Apache child processes running on the system at any one time. For Apache–based servers, use a value exceeding the maxclients setting by 200 or more.

**Example:** (Solaris 9) 200

3. Save your changes then exit the file or the utility.
4. Reboot the system.
5. Verify your changes by executing the following command:
   ```
   $ sysdef -i
   ```

# How to Tune the Solaris 10 Resource Controls

Tune the resource controls at the project level to improve the performance of the agent.

**Note:** See your Solaris documentation for more information.

Tuning the resource controls on Solaris 10 uses the following process:

1. Determine the project that is associated with the user account under which the Web Agent runs.
2. Increase the settings for any of the following resource controls of that project:

   **project.max-shm-ids**

   Specifies the maximum shared memory IDs for a project.

   **project.max-sem-ids**

   Specifies the maximum number of semaphore IDs for a project.

   **project.max-msg-ids**

   Specifies the maximum number of message queue IDs for a project.

   **project.max-shm-memory**

   Specifies the total amount of shared memory allowed for a project.

   **process.max-sem-nsems**

   Specifies the maximum number of semaphores allowed per semaphore set.

   **process.max-sem-ops**

   Specifies the maximum number of semaphore operations allowed per semop.

   **process.max-msg-messages**

   Specifies the maximum number of messages on a message queue.

   **process.max-msg-qbytes**

   Specifies the maximum number of bytes of messages on a message queue.

# Chapter 17: SPS APIs

This section contains the following topics:

Session Scheme API (see page 239)
Filter API Overview (see page 246)

## Session Scheme API

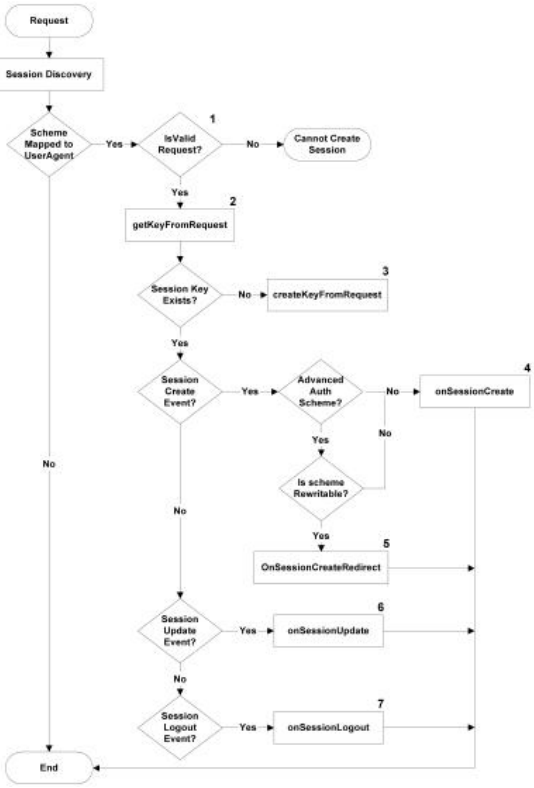
The SPS supports a Java API that allows you to develop custom session schemes. These schemes can be assigned to user agent types for each virtual host configured on the SPS.

### Overview of Session Scheme API Processing

The SPS processes a number of methods to establish, maintain, and end a typical user session. One of the steps during session processing is to determine whether a scheme is rewritable. Rewritable schemes provide the ability to modify the URL. The simple URL rewriting session scheme is an example of a rewritable scheme, since part of the processing of a request includes rewriting the requested URL to include a token.

To implement a rewritable session scheme, you must implement the rewritable interface, which is described in Rewritable Session Schemes (see page 244).

The following illustration shows the process flow for the session scheme API methods.




The methods identified in the illustrated are:

1. **isValidRequest**—This method must be implemented in a custom session scheme to determine and verify the conditions that make up a valid request.
2. **getKeyFromRequest**—This method must be implemented to extract a key from a valid request. If no key is present, the createKeyFromRequest method is called.
3. **createKeyFromRequest**—This method must be implemented to trigger the creation of a key for a new session.
4. **onSessionCreate**—On the event of session creation, if the session scheme in use is not rewritable, this method is called. This method may be implemented with any code that should be triggered at the inception of a new session.
5. **onSessionCreateRedirect**—On the event of session creation, if the scheme is rewritable, this method is called. This method may be implemented with any code that should be called at the inception of a new session for a rewritable session scheme.

6. **onSessionUpdate**—A session is updated for each new request made during the session. This method is called during each session update. It may be implemented by adding any code that should be triggered during a session update.
7. **onSessionLogout**—This method is called when a session is terminated. It may be implemented with any code that should be executed when a user session is terminated.

## Session Scheme API Class Files

The SPS session scheme API makes use of the session scheme abstract class contained in *sps_home*/Tomcat/server/lib/proxycore.jar.

## Constructor for Session Scheme API

The constructor for a custom session scheme must consist of the following:

```
public IPAddrSessionScheme(String name, boolean
                                acceptFlag, Hashtable props) {
               // Always call the parent constructor for proper
               // initialization of the scheme

               super(name,acceptFlag,props);
```

The settings are as follows:

**name**

String that is populated by the name in the server.conf file associated with your custom session scheme class.

**acceptFlag**

Boolean value that determines whether or not the custom session scheme accepts SiteMinder's SMSESSION cookies.

**props**

List of name/value pairs for any other properties required by the session scheme as specified in the server.conf file.

## Session Scheme API Methods

The session scheme API class consists of the following methods:

| Return Value | Method | Notes |
|---|---|---|
| Boolean | acceptsCookies() | Retrieves the value of the acceptFlag from the accepts_smsession_cookies parameter of the session scheme in the server.conf file and returns a value indicating whether this scheme supports SiteMinder SMSESSION cookies. |
| abstract java.lang.String | createKeyFromRequest(HttpServletRequest req) | Executes code to retrieve values needed to create a new session key from the request. |
| abstract java.lang.String | getKeyFromRequest(HttpServletRequest req) | Retrieves the session key from a request. |
| java.lang.String | getName() | Retrieves the name of the custom session scheme as defined in the server.conf file. |
| abstract Boolean | isValidRequest(HttpServletRequest req) | Evaluates if the request for this session scheme is valid. |
| abstract int | onSessionCreate(java.lang.String id, HttpServletRequest req, HttpServletResponse resp) | Hook that is available on the event of session creation. |
| java.lang.String | onSessionCreateRedirect(java.lang.String id, java.lang.String url, HttpServletRequest req, HttpServletResponse resp) | Hook that is available on the event of session creation for rewritable schemes. |
| abstract void | onSessionLogOut(HttpServletRequest req, HttpServletResponse resp) | Hook that is available at the event of a session termination (logout). |
| abstract void | onSessionUpdate(HttpServletRequest req, HttpServletResponse resp) | Hook that is available on the event of session updates. This method is only for internal use. |
| static Boolean | usingDefaultSessionScheme(HttpServlet Request req) | Helper method for recognizing that a request is using the default session scheme. |

## Implement a Custom Session Scheme

Use the following procedure to implement a custom session scheme.

**Follow these steps:**

1. Review the sample code for the IP address session scheme in IP Address Session Scheme.
2. Write source code for your session scheme.
3. If you are creating a rewritable session scheme, implement the rewritable interface.
4. Ensure that your system CLASSPATH includes the following content:
   - proxycore.jar which contains the session scheme API
   - JDK version 1.6.0_30 or higher jar files
   - *sps_home*/Tomcat/server/lib jar files
5. Compile the session scheme.
6. Do *one* of the following steps:
   - Create a .jar file that contains your custom session schemes then copy the .jar file to the directory *sps_home/*Tomcat/server/lib.
   - Add the class files for your custom session scheme to the *sps_home/*Tomcat/server/classes directory then configure scheme in the SPS server.conf file.
7. Restart the SPS.

### Configure Custom Session Scheme in the server.conf File

When you compile the code for a custom session scheme you must configure the session scheme in the SPS server.conf file. To configure the session scheme, add a SessionScheme element to the file. For example:

```
<SessionScheme name="custom_scheme">
                class="com.netegrity.proxy.session.CustomScheme"
                accepts_smsession_cookies="false"
                property1="value1"
                property2="value2"
</SessionScheme>
```

In addition, if you have configured user agent types, you can map the session scheme to any appropriate user agents types.

**More information:**

Session Scheme Mapping for the Default Virtual Host (see page 127)

# Configure Rewritable Session Schemes

If a session scheme must have the ability to modify the URL requested by a user, you must implement the rewritable interface. For example, this interface is used by the simple URL rewriting scheme to enable the session scheme to append a token to the end of URL requests.

## Implement the Rewritable Interface

When implementing the rewritable interface, the following methods are available:

| Return Value | Method | Description |
|---|---|---|
| public string | rewrite(String url, String id, HttpServletRequest req) | Rewrites a requested URL to contain a session identifier. |
| public string | onSessionCreateRedirect(String id, String url, HttpServletRequest req, HttpServletResponse resp) | Provides a callback for the event of session creation by redirection. It is typically used in conjunction with forms-based authentication, where the target URL is different from the requested URL. For example, the authentication scheme may modify the URL or add a cookie. |

In addition to the rewritable interface, the methods must be implemented in the custom session scheme.

| Return Value | Method | Description |
|---|---|---|
| protected void | setRequestURI(HttpServletRequest req, String uri) | Allows the scheme to modify the request URI. |
| protected void | setRequestPathInfo(HttpServletRequest req, String pathInfo) | Allows the scheme to modify the path information of the request. |

## Use an IP Address Session Scheme

The default SPS installation includes an IP address session scheme. This scheme maps a session using the IP address of the client. When a user makes a request, the SPS retrieves the client's IP address from the HTTP headers and uses this to generate the session key for the client's session.

The IP address session scheme was created using the session scheme API. The source code for this scheme can be found in the directory *sps_home*\secure-proxy\proxy-engine\examples\sessionschemes.

**Note:** In the sample session scheme file, a backslash (\) character indicates that the line should continue, but must be interrupted due to space constraints in this document.

**To implement an IP address session scheme**

1. Add a <SessionScheme> section to your server.conf file like the following:

```
<SessionScheme name="ip_address">
    class="com.netegrity.proxy.session.IPAddrSessionScheme"
    accepts_smsession_cookies="false"
    allowed_proxied_addresses="true"
</SessionScheme>
```

The directives are:

**class**

This directive specifies the Java class that handles IP address session schemes. This value should not be modified if you want to use the default IP address session scheme installed with the SPS.

**Default:** com.netegrity.proxy.session.IPAddrSessionScheme

**accepts_smsession_cookies**

Indicates that SiteMinder smsession cookies are not supported by this session scheme. To ensure a cookieless session using the IP address scheme, the value of this directive should not be changed.

**Default:** false

**allowed_proxied_addresses**

Indicates whether or not requests will be validated using the SessionScheme.isValidRequest call. Set the value to true to allow the use of proxied addresses. Accept the default, false to use the isValidRequest method for determining if the VIA HTTP header variable is present. If this variable is present, the SPS determines that the address is proxied and blocks the request.

**Default:** true

2. Map the session scheme to one or more user agents for a virtual host in the server.conf file.
3. Restart the SPS.

## Session Storage API

The SPS stores mappings from a session token to a SiteMinder session. This information is accessed using the SessionStorageAPI.

The SessionStorageAPI provides the following capabilities:

**Session creation**

Allows the creation of a new session.

**Session update or synchronization**

Allows updates to SiteMinder session information.

**Session retrieval**

Allows the retrieval of session information when provided with the correct session key.

**Explicit session removal**

Allows the removal of a session using a specific session key.

**Session expiration**

Allows the removal of all expired sessions.

# Filter API Overview

Custom filters are filters defined by customer's needs. SPS uses custom filters to manipulate a request before forwarding the request to a backend server, and also to manipulate the responses sent by the backend server to the user client.

The SPS can process a single custom filter or a group of custom filters for each request. When you create a custom filter group, the SPS processes all the filters that are part of the custom filter group in a chain.

You can look at the source code for a pre-processing filter and a post-processing filter produced with the filter API. These samples may be found in the following directory:

*sps_home*/proxy-engine/examples/filters

**Note:** In the code samples, a backslash (\) character indicates that the line should continue, but must be interrupted due to space constraints in this document.

**More Information**

Associate Custom Filters to Proxy Rules (see page 247)

## How SPS Processes Custom Filters

The SPS includes an API for inserting pre-processing and post-processing into the proxy stage of a request.

In a standard SPS transaction, the following process occurs:

1. User requests a resource.
2. SPS examines its proxy rules and determines where to direct the request (after successful authentication and authorization).
3. Destination server sends the requested resource to the SPS, which passes the resource to the user.

The Filter API provides a method for developers to insert processing before a request is passed to a destination server, as described in step 2 of the preceding process, or after the response from the destination server is returned to the SPS as described in step 3 of the preceding process, but before the resource is passed to the user.

## Associate Custom Filters to Proxy Rules

When the SPS receives a request or a response, the SPS reads the proxy rules and processes the associated filters. The custom filters or custom group filters that are declared in the server.conf file must be associated with proxy rules. To associate custom filters or custom group filter to proxy rules, open the proxyrules.xml located in *<install dir>*/secure-proxy/proxy-engine/conf, edit the proxyrules.xml file for the rule that is expected to run the filter.

For example:

<nete:forward filter="*your filter name or your groupfiltername*">http://FQDN$0</nete:forward>

## Filter API Class File

The SPS Filter API makes use the proxy filter classes contained in *sps_home*/Tomcat/server/lib/proxyrt.jar.

## ProxyFilter Interface

The ProxyFilter interface defines the interface implemented by a proxy filter. However, it is recommended that you extend the BaseProxyFilter Abstract Implementation, rather than implementing the ProxyFilter interface.

The ProxyFilter interface consists of the following methods:

| Return Value | Method |
|---|---|
| void | doFilter(ProxyRequest prequest, ProxyResponse presponse)<br>Performs the filtering.<br>Parameters:<br>request - the proxy request data<br>response - the proxy response data<br>Throws:<br>ProxyFilterException - thrown if failure processing filtering. |
| ProxyFilterConfig | getFilterConfig()<br>Returns this filter's ProxyFilterConfig object. (ProxyFilterConfig object that initialized this filter). |
| void | init(ProxyFilterConfig config)<br>Called when the filter is created to perform any required initialization.<br>Parameters:<br>config - a ProxyFilterConfig object containing the filters's configuration and initialization parameters<br>Throws:<br>ProxyFilterException - thrown if failure initializing this filter. |

## BaseProxyFilter Abstract Implementation

The Filter API includes BaseProxyFilter, an abstract implementation of a proxy filter that can be implemented as a subclass to create ProxyFilters.

**Note:** It is recommended that you extend the BaseProxyFilter Abstract Implementation, rather than implementing the ProxyFilter interface.

A subclass of BaseProxyFilter must override at least one of the following methods:

- doPreFilter
- doPostFilter
- doFilter (not recommended)

The BaseProxyFilter includes filter initialization and separates pre-processing and post-processing hooks for inserting your own filters into SPS transactions, as listed in the following table.

| Return Value | Method |
|---|---|
| Void | doFilter(ProxyRequest prequest, ProxyResponse presponse) throws ProxyFilterException<br>This implementation determines the state of the request processing and calls doPreFilter if it's in an inbound state otherwise it calls doPostFilter for outbound state. At the time the filters get called processing can only be in one of these states.<br>Specified by:<br>doFilter in interface ProxyFilter<br>Parameters:<br>request - the proxy request data<br>response - the proxy response data<br>Throws:<br>ProxyFilterException - thrown if failure processing filtering |
| Void | doPreFilter(ProxyRequest prequest, ProxyResponse presponse) throws ProxyFilterException<br>Performs pre-filtering. Override this method to perform filtering tasks before the request is sent to the target server.<br>Parameters:<br>request - the proxy request data<br>response - the proxy response data<br>Throws:<br>ProxyFilterException - thrown if failure processing filtering |
| Void | doPostFilter(ProxyRequest prequest, ProxyResponse presponse) throws ProxyFilterException<br>Performs post-filtering. Override this method to perform filtering tasks after the response is received from the target server.<br>Parameters:<br>request - the proxy request data<br>response - the proxy response data<br>Throws:<br>ProxyFilterException - thrown if failure processing filtering |

| Return Value | Method |
|---|---|
| ProxyFilterConfig | getFilterConfig()<br>Returns this filter's ProxyFilterConfig object.<br>Specified by:<br>getFilterConfig in interface ProxyFilter<br>Returns:<br>ProxyFilterConfig the ProxyFilterConfig object that initialized this filter. |
| Void | init(ProxyFilterConfig config) throws ProxyFilterException<br>Called when the filter is created to perform any required initialization.<br>**Note:** When overriding this method, the first statement should call the parent init method "super.init(config);".<br>Specified by:<br>init in interface ProxyFilter<br>Parameters:<br>config - a ProxyFilterConfig object containing the filters's configuration and initialization parameters<br>Throws:<br>ProxyFilterException - thrown if failure initializing this filter. |

## ProxyFilterConfig Interface

Defines the interface to the configuration data available to a filter. The interface consists of the following methods:

| Return Value | Method |
|---|---|
| java.lang.String | getFilterName()<br>Returns the name of this filter. |

| Return Value | Method |
|---|---|
| java.lang.String | getInitParameter(java.lang.String name)<br>Returns a String containing the value of the named initialization parameter, or null if the parameter does not exist.<br>Parameters:<br>name - a String specifying the name of the initialization parameter |
| java.util.Enumeration | getInitParameterNames()<br>Returns the names of the filter's initialization parameters as an Enumeration of String objects, or an empty Enumeration if the filter has no initialization parameters. |

## ProxyResponse Interface

Defines the interface that provides access to HTTP response information to be returned to the proxy client. The interface consists of the following methods:

| Return Value | Method |
|---|---|
| void | addHeader(java.lang.String name, java.lang.String value)<br>Adds a header with the specified name and value. This method allows response headers to have multiple values.<br>Parameters:<br>name - a String specifying the header name<br>value - a String specifying the header value |
| byte[] | getContent()<br>Returns a byte array of the content of the response to the proxy request. This is the content to be returned to the proxy client. |
| java.lang.String | getHeader(java.lang.String name)<br>Returns the value of the specified header as a String. If the header does not exist, this method returns null. The header name is not case sensitive.<br>Parameters:<br>name - a String specifying the header name |

| Return Value | Method |
|---|---|
| java.util.Enumeration | getHeaderNames()<br>Returns an Enumeration of all the header names. If no headers exist, this method returns an empty Enumeration. |
| int | getStatusCode()<br>Returns the HTTP response status code of the response to the proxy request. |
| java.lang.String | removeHeader(java.lang.String name)<br>Removes the specified header. Returns the value of the removed header as a String. If the header does not exist, this method returns null. The header name is not case sensitive.<br>Parameters:<br>name - a String specifying the header name |
| void | setContent(byte[] content)<br>Sets the content of the response to the proxy request. This overwrites the content to be returned to the proxy client.<br>Parameters:<br>content - the byte array containing the content |
| void | setHeader(java.lang.String name, java.lang.String value)<br>Sets a header with the specified name and value. If a header with the same name exists it will be overwritten.<br>Parameters:<br>name - a String specifying the header name<br>value - a String specifying the header value |

## ProxyFilterException Class

The ProxyFilterException class defines a general exception that a filter can throw when it encounters difficulty.

| Constructor Signature | Description |
|---|---|
| ProxyFilterException() | Constructs a new ProxyFilterException. |

| Constructor Signature | Description |
|---|---|
| ProxyFilterException(java.lang.String message) | Constructs a new ProxyFilterException with the specified message.<br>Parameters:<br>message - Message of exception |
| ProxyFilterException(java.lang.String message, java.lang.Throwable rootCause) | Constructs a new ProxyFilterException with the specified message and root cause.<br>Parameters:<br>message - Message of exception<br>rootCause - Exception that caused this exception to be raised |
| ProxyFilterException(java.lang.Throwable rootCause) | Constructs a new ProxyFilterException with the specified message and root cause.<br>Parameters:<br>rootCause - Exception that caused this exception to be raised |

## ProxyRequest Interface

Defines the interface that provides access to HTTP request information to be sent by the proxy. The interface consists of the following methods:

| Return Value | Method |
|---|---|
| java.lang.String | getHeader(java.lang.String name)<br>Returns the value of the specified header as a String. If the header does not exist, this method returns null. The header name is not case sensitive.<br>Parameters:<br>name - a String specifying the header name |
| java.util.Enumeration | getHeaderNames()<br>Returns an Enumeration of all the header names. If no headers exist, this method returns an empty Enumeration. |
| javax.servlet.http.HttpServletRequest | getOriginalRequest()<br>Returns the original HttpServletRequest made to the proxy. |

| Return Value | Method |
|---|---|
| java.lang.String | getSessionKey()<br>Returns the value of the session key as a String. If the key is not available, this method returns null. The key may be used to rewrite URL's in the content when using cookieless schemes.<br>**Note:** The SessionScheme is responsible for creating the key and storing it in an attribute named SessionScheme.DEFAULT_SESSION_KEY_NAME |
| java.lang.String | getTargetQueryString()<br>Returns the query string the proxy will use with the target URL. The query string may be from the original request or a new one defined through the proxy rules. This method returns null if the URL does not have a query string. |
| java.lang.String | getTargetURL()<br>Returns the URL the proxy will use to make the request as defined by the proxy rules. The URL does not include query string parameters. |
| boolean | isInbound()<br>Returns a boolean value indicating the state of the request processing. If the request has not been forwarded to the target server true is returned. If the request was sent and the response received false is returned. |
| boolean | isOutbound()<br>Returns a boolean value indicating the state of the request processing. If the request has not been forwarded to the target server false is returned. If the request was sent and the response received true is returned. |
| java.lang.String | removeHeader(java.lang.String name)<br>Removes and returns the value of the specified header as a String. If the header does not exist, this method returns null. The header name is not case sensitive.<br>Parameters:<br>name - a String specifying the header name |

| Return Value | Method |
|---|---|
| void | setHeader(java.lang.String name, java.lang.String value)<br>Sets a header with the specified name and value. If a header with the same name exists it will be overwritten.<br>Parameters:<br>name - a String specifying the header name<br>value - a String specifying the header value |
| byte[] | getContent()<br>Returns a byte array of the content of the Request POST data. This is the content which is sent to the backend server. |
| void | setContent(byte[] content)<br>Sets the content of the POST data to the proxy request.<br>This overwrites the content to be sent to the backend server.<br>Parameters:<br>content - the byte array containing the request POST data |

## Implement a Filter

Filters that use the session key depend on the session scheme to define the key. To make the session key available to a filter, an attribute keyed by SessionScheme.DEFAULT_SESSION_KEY_NAME must be set to hold the value of the key when it is created by the createKeyFromRequest(..) callback and retrieved on subsequent requests by the getKeyFromRequest(..) callback of the Session Scheme API.

Out-of-the-box session schemes that generate the session key are:

- Mini-cookies
- Simple URL rewriting

**Follow these steps:**

1. Review the sample code for filters in Filter Examples.
2. Write source code for your filter.

3. Ensure that your system CLASSPATH includes the following content:
   - proxyrt.jar which contains the filter API
   - JDK version 1.6.0_30 or higher jar files
   - *sps_home*/Tomcat/server/lib jar files
4. Compile the filter.
5. Do *one* of the following steps:
   - Create a .jar file that contains your filter and copy the file to the directory *sps_home/*Tomcat/server/lib directory.
   - Add the class files for your filter to the *sps_home/*Tomcat/server/classes directory, in a subdirectory that corresponds to the package name.
6. Configure the SPS server.conf file.
7. Edit the proxyrules.xml file for the rule that is expected to implement the filter. For example:
   <nete:forward filter="your filter name">http://FQDN$0</nete:forward>
8. Restart the SPS.

## Filter API Example

The SPS installation includes sample source files for a preprocessing filter and a post-processing filter. Both of these samples use the BaseProxyFilter Abstract Implementation. For a complete description of the example filters, see Filter Examples.

## Using a Filter to Rewrite Absolute Links in a Requested Page

One of the most common uses of the Filter API is to support the rewriting of absolute links in pages requested by a user through the SPS. For absolute links to be handled properly by the SPS, you must use the Filter API to perform the appropriate substitution for any absolute links contained in resources returned to the user based on a SPS request.

# Chapter 18: Troubleshooting

This section contains the following topics:

Unable to Start Apache on UNIX systems (see page 257)
Non-english Input Characters Contain Junk Characters (see page 258)
Unable to Log Federation Web Services Errors (see page 258)
DNS Caching in the SPS (see page 259)
Resource Request Fails (see page 259)
No Root Permissions (see page 263)
Cannot Start the SPS Server (see page 263)
Cannot Access the SPS with a Browser (see page 264)
Unknown Server Name (see page 264)
Issues Configuring Virtual Hosts (see page 265)
Command not found Error Received (see page 265)
SPS Not Forwarding Requests (see page 265)
SPS and SharePoint (see page 266)

## Unable to Start Apache on UNIX systems

**Symptom:**

When running the SPS on a UNIX system, the Apache server fails to start. In the Apache log file, the following error message appears:

```
Invalid argument: setgid: unable to set group id to ...
```

**Solution:**

This error occurs when the group for the Run-As-User on UNIX systems does not correspond to the group specified in the Apache configuration file (httpd.conf). If you see this error, edit the Group directive in the Apache httpd.conf file.

**To edit the Group directive**

1. Remove the comment sign (#) before the Group directive
2. Specify the group to which the Run-As-User belongs.
3. Run the SPS startup command again (sps-ctl start or startssl).

# Non-english Input Characters Contain Junk Characters

**Symptom:**

When I install or configure SiteMinder components in the console mode on UNIX machines, few non-English input characters are not displayed correctly in the console window.

**Solution:**

Verify terminal settings of your console window and confirm that the console does not clear high (8th) bit of input characters by executing the following command:

```
stty –istrip
```

# Unable to Log Federation Web Services Errors

**Symptom:**

The Federation Web Services errors are not logged.

**Solution:**

To log the errors in Federation Web Services, enable the AffWebServices and FWSTrace logs parameters in the LoggerConfig.properties file.

**Follow these steps:**

1. Open the LoggerConfig.properties file.

   **Default Path**:
   *sps_home*/secure-proxy/Tomcat/webapps/affwebservices/WEB-INF/classes/LoggerConfig.properties

2. Configure the following parameters:

   ```
   LoggingOn=Y
   ```

   ```
   TracingOn=Y
   ```

3. Save the changes.

# DNS Caching in the SPS

**Symptom:**

I do not want the SPS to cache the DNS name look-up settings of the server.

**Solution:**

The SPS is configured by default to cache the DNS settings of the server. To change this default behavior, adjust the networkaddress.ttl setting in the java.security file.

**Follow these steps:**

1. Navigate to the directory *sps_home*\secure-proxy\JDK\1.6.0\jre\lib\security.
2. Open the java.security file.
3. Set the networkaddress.cache.ttl parameter to a positive integer. For example, networkaddress.cache.ttl=2

   **networkaddress.ttl**

   Specifies the duration, in seconds, for which the SPS caches the successful DNS name look-ups. Enter a positive integer. If you enter a negative value, the SPS caches the DNS settings.

   **Default:** -1

# Resource Request Fails

**Symptom:**

SPS failed to serve a resource request.

**Solution:**

To troubleshoot an error, verify the following log files for the error details:

- spsagent and spsagenttrace logs
- Apache access and error logs
- httpclient.log
- server.log
- mod_jk.log

If the log files do not contain logs, ensure that you have enabled logging in the log files.

**More information:**

Configure spsagent Logs (see page 260)
Configure SPSAgentTrace Logs (see page 261)
Configure the mod_jk.log File (see page 262)
Configure the httpclient.log File (see page 262)

## Configure spsagent Logs

CA SiteMinder SPS logs errors that are related to the proxy engine in the spsagent logs. A local configuration file or an ACO in the Policy Server contains the parameters that enable error logging and determine logging options.

**Follow these steps:**

1. Open the ACO of CA SiteMinder SPS in the Policy Server.
2. Set the value of the LogFile parameter to yes.

   **Note**: Setting the value of this parameter to yes in a local configuration file overrides any of the logging settings that are defined on the Policy Server.
3. Complete the following parameters:

   **LogFileName**

   Specifies the full path including the file name, to the log file.

   **LogAppend**

   Adds new log information to the end of an existing log file. When this parameter is set to no, the entire log file is rewritten each time logging is invoked.

   **LogFileSize**

   Specifies the size limit of the log file in megabytes. When the current log file reaches this limit, a new log file is created.

   **LogLocalTime**

   Specifies whether the logs use Greenwich Mean Time (GMT) or local time. To use GMT, change this setting to no. If this parameter does not exist, the default setting is used.

   The error logs are configured.

## Configure SPSAgentTrace Logs

You can configure the trace logs to control the size and format of the file. After trace logging is configured, you determine the content of the trace log file separately. This lets you change the types of information contained in your trace log at any time, without changing the parameters of the trace log file itself.

**Follow these steps:**

1. Locate the SecureProxyTrace.conf file, and duplicate the file.
2. Open your Agent Configuration Object or local configuration file.
3. Set the TraceFile parameter to yes.

   **Note**: Setting the value of this parameter to yes in a local configuration file overrides any of the logging settings that are defined on the Policy Server.
4. Configure the following parameters:

   **TraceFileName**

   Specifies the full path to the trace log file.

   **TraceConfigFile**

   Specifies the location of the SecureProxyTrace.conf configuration file that determines which components and events to monitor.

   **TraceAppend**

   Specifies if the new logging information must be added to the end of an existing log file instead of rewriting the entire file each time logging is invoked.

   **TraceFormat**

   Specifies how the trace file displays the messages.

   **TraceDelimiter**

   Specifies a custom character that separates the fields in the trace file.

   **TraceFileSize**

   Specifies the maximum size of a trace file in megabytes. CA SiteMinder SPS creates a new file when this limit is reached.

   **LogLocalTime**

   Specifies whether the logs use Greenwich Mean Time (GMT) or local time. To use GMT, change this setting to no. If this parameter does not exist, the default setting is used.
5. Restart CA SiteMinder SPS.

## Configure the mod_jk.log File

CA SiteMinder SPS logs all the communication messages between Apache and the proxy engine in the mod_jk.log file. By default, logging is enabled in this file and the log file is located in *sps_home*\secure-proxy\httpd\logs\mod_jk.log.

**Follow these steps:**

1. Open the httpd.conf file.

   **Default Path**: *sps_home*\secure-proxy\httpd\conf

2. Modify the available parameters as required.

   **Note**: For information about configuring the httpd.conf file and the mod_jk.log file, see the Apache documentation set.

3. Ensure that JkRequestLogFormat is set in the %w %V %T %m %h %p %U %s format.
4. Save the changes.
5. Restart CA SiteMinder SPS.

## Configure the httpclient.log File

For debug purposes only, you can enable the httpclient.log. By default, the httpclient.log file is located in *sps_home*\secure-proxy\proxy-engine\logs.

**Follow these steps:**

1. Open the server.conf file
2. Ensure that httpclientlog is set to yes.
3. Open the httpclientlogging.properties file.

   **Default Path**: *sps_home*\Tomcat\properties directory

4. Modify the available parameters as required.

   **Note**: For information about configuring the httpclientlogging.properties file, see the Apache documentation set.

5. Save the changes.

# No Root Permissions

**Symptom:**

Configuring the SPS without root permissions.

**Solution:**

Use the following information to troubleshoot:

- You can still install SPS; however, the automatic process cannot complete all of the installation steps. The Installation Program displays warnings to help you determine which files you need to edit by hand.
- Non-root users are generally not allowed to bind to ports 80 and 443. Therefore, the default ports for HTTP and HTTPS traffic are set to 8080 and 8443 for non-root installations. This enables you to test your server without editing the httpd.conf file. Test with:
  - http://yourserver.com:8080
  - https://yourserver.com:8443

  **Note:** Non-root installations are not recommended for SSL-enabled servers. A non-root installation is less secure because it allows an additional person with root permissions access to your keys and certificates.

# Cannot Start the SPS Server

**Symptom:**

SPS server fails to start.

**Solution:**

Use the following information if you cannot start your server:

- Verify that the ServerName directive in *sps_home*/secure-proxy/httpd/conf/httpd.conf corresponds to the name of your server.
- Verify that the server is not already running by executing one of the following:
  - ps -ax|grep http on BSD compatible systems
  - ps -elf|grep http on System V release 4 compatible systems

  If this results in a list of processes, stop the running server before starting your new server.
- Check the log files in the directory *sps_home*/secure-proxy/httpd/logs

- Verify that the SSLCertificateFile and the SSLCertificateKeyFile directives in the httpd.conf file point to your certificate and key files. The file is in the directory *sps_home*/secure-proxy/httpd/conf
- Determine whether you are using non-IP-based virtual hosts. SSL requires IP-based virtual hosts.
- Verify that no other server is running on the default port for the SPS. The default port is specified in the httpd.conf file.
- If you using SSL, be sure you have generated a key and certificate before starting the server, otherwise you will get an error.

# Cannot Access the SPS with a Browser

**Symptom:**

Difficulty accessing the SPS using a browser.

**Solution:**

To access the SPS using a browser:

- Verify that DNS is aware of your servername with the command nslookup servername or try to 'ping' your server with the ping servername command.
- Run the server without SSL and access your web site to verify whether the problem is with the key or certificate files. To start the server without SSL, execute ./sps-ctl start in the directory *sps_home*\secure-proxy\proxy engine directory.
- Try to make a telnet connection to ports 80 and 443 of your Web server (or the non-default ports you specified). If you installed as a non-root user, try to connect to ports 8080 and 8443.

# Unknown Server Name

**Symptom:**

Determining the server name.

**Solution:**

Do any one of the following to determine your server name:

- Use the hostname command.
- Enter echo $DISPLAY at a command prompt. If the result is servername:0.0, your servername is the portion that precedes the colon.
- Consult the guide for your operating system.

# Issues Configuring Virtual Hosts

**Symptom:**

Difficulty configuring virtual hosts.

**Solution:**

Refer to the information about configuring virtual hosts at:

http://httpd.apache.org/docs-2.0/vhosts/

# Command not found Error Received

**Symptom:**

Command not found error,

**Solution:**

- Use the ls command to verify that the command exists in the directory you are in and that it is spelled correctly.
- Add ./ before the command, for example,./setup.
- If you are using Solaris and encounter problems running the SPS, verify that you have installed all recommended operating system patches for your version of Solaris. For more information, see the release notes for the SPS.

# SPS Not Forwarding Requests

**Symptom:**

404 File Not Found browser error.

**Solution:**

If you receive a 404 File Not Found browser error and there is no action in the Web Agent log for the SPS Web Agent, verify the name an IP address of the virtual host in the server.conf file.

# SPS and SharePoint

**Symptom:**

When accessing a SharePoint page through the SPS, the SPS always displays the Alternate Access Mapping Connection parameter, no matter how the mode (Forward or Redirect) is set in the proxyrule.xml file.

**Solution:**

Follow these steps for a solution to this problem:

1. On the SharePoint server, go to Central Administration, Operation, Alternate Access Mapping. Notice that the Alternate Access Mapping includes a Default Zon Internal URL and a Public URL.
2. Add one Internal URL with the Public URL set as http://<SPS Host>:port and Default Zone.
3. Add one more Internal URL Public URL set as http://<SharePoint Host>:port and Default Zone.
4. Edit the entry for the Intranet zone created in step 3 and specify the Public URL as http://<SPS Host>:port
5. In the SPS proxyrule.xml file, the backend is an internal URL with a public URL pointing to the SPS host. For example:

```
<!--Proxy Rules-->
<nete:proxyrules xmlns:nete="http://ww.ca.com/">
      <nete:forward>http://SharePointServer with public URL as SPS
           host:port$0</nete:forward>
</nete:proxyrules>
```

# Index


## A

acceptFlag • 241
acceptsCookies • 242
access control • 169
addHeader • 251
Agent • 177, 187
Agents • 177
Apache • 81, 201
API
    filter • 256

## B

Basic authentication schemes • 178
beginswith • 148

## C

cache
    session • 101
    session store • 101
caching • 169
Certificate Authority • 201
clean_up_frequency • 101
clusters • 169, 170
comparison type attribute • 147
compatibility • 180
connection pool • 107
constructor
    session scheme API • 241
containers, virtual host • 207
cookie-less
    session schemes • 170, 182
createKeyFromRequest • 239, 242

## D

destination servers • 129
device ID session scheme • 118
DMZ • 129, 179, 184, 186, 189
Documentation Type Definition
    proxy rules • 145
doFilter • 248
doPostFilter • 248
doPreFilter • 248
DTD • 145
dynamic header value • 161

## E

enableredirectrewrite • 129
error
    413 Request Entity Too Large • 189
extranet • 180

## F

federation gateway • 71
filter • 153
filter API • 256
forward • 103, 108, 148

## G

getContent • 251
getFilterConfig • 248
getFilterName • 250
getHeader • 251, 253
getHeaderNames • 251, 253
getInitParameter • 250
getInitParameterNames • 250
getKeyFromRequest • 239, 242
getName • 242
getOriginalRequest • 253
getSessionKey • 253
getStatusCode • 251
getTargetQueryString • 253
getTargetURL • 253

## H

header • 164
headers • 171
hostname • 141, 166
HTML forms authentication schemes • 178
HTTP • 171
    header • 164
HTTP header
    header • 171
HTTPS • 171
httpsd.conf • 81, 170

## I

init • 248
in-memory session store • 170

intranet • 180
IP address
  session scheme • 114
IP-based virtual hosts, requirement for SSL • 207
isInbound • 253
isOutbound • 253
isValidRequest • 239, 242

## K

keep-alives • 107

## L

LDAP search expression • 184
load balancing
  devices • 169

## M

Managed Self-Registration • 186
max_size • 101
mini-cookies
  session scheme • 114
MSR • 186, 187

## N

name-based virtual hosts • 207
nete
  case • 147, 151
  cond • 145, 147, 151
  default • 151
  description • 146
  forward • 145, 147, 152, 153, 160, 161
  local • 145, 147
  proxyrules • 145
    debug • 146
  redirect • 145, 147, 153, 160, 161
  result • 154, 159, 160, 161
  rule • 159
  xpr • 154, 156, 159
  xprcond • 147, 154, 156, 159
  xpr-default • 154, 156

## O

OnRejectRedirect • 182
onSessionCreate • 239, 242
onSessionCreateRedirect • 239, 242
onSessionLogout • 239, 242
onSessionUpdate • 239, 242

## P

password policies • 184
password services • 184
policies • 177, 180
policy domains • 177
Policy Server • 177, 187, 188
post-processing • 256
pre-processing • 256
private key • 201
props • 241
protection level • 178, 182
Proxy Engine • 81, 156
proxy rules • 162
  DTD • 145
  hostname • 166
  regular expressions • 167
  URI • 165
Proxy Trust Mode • 180
ProxyFilterException • 252
proxyrules.dtd • 162
proxytimeout • 179
proxytrust • 179

## Q

query string • 156

## R

realms • 177, 180
redirect • 103, 108, 148, 160, 182
redirect URL • 129
redirectrewritablehostnames • 129
regular expression • 154, 156, 159, 167
removeHeader • 251, 253
responses • 162, 177, 182
  SiteMinder • 162
rules • 177, 180
rules_file • 162

## S

server certificate
  See also certificate • 201
server configuration • 81
server.conf • 81, 84, 101, 103, 162, 170, 172, 189
services configuration • 81
servlet • 187
session scheme • 170
  device ID • 118

IP address • 114
mini-cookies • 114
SSL ID • 113
session scheme API
acceptFlag • 241
acceptsCookies • 242
createKeyFromRequest • 239, 242
getKeyFromRequest • 239, 242
getName • 242
isValidRequest • 239, 242
name • 241
onSessionCreate • 239
onSessionCreateRedirect • 239, 242
onSessionLogout • 239, 242
onSessionUpdate • 239, 242
props • 241
usingDefaultSessionScheme • 242
session schemes • 81
session store • 170
session store cache • 101
sessions scheme API
onSessionCreate • 242
SessionStore • 101
setContent • 251
setHeader • 251, 253
single sign-on • 169, 170
SiteMinder • 177, 180, 182, 186
cookie • 170
objects • 177
responses • 162
SiteMinder responses • 162
SM_PROXYREQUEST • 190
SM_REWTRITE_URL • 182
SSL • 170
SSL ID session scheme • 113


## T


Template Path • 188
timeout • 107


## U


URI • 148, 156, 159, 165
matching • 147
substring • 148
URL • 148
user agent • 81
user directories • 177
usingDefaultSessionScheme • 242


## V


virtual host configuration • 129
virtual hosts • 81, 201


## W


Web Agent • 184, 186
Web Agents • 177, 179
Web servers • 177
WebAgent.conf • 179