CA SiteMinder® Secure Proxy Server

Administration Guide

12.52 SP1

ca technologies

CA Technologies Product References

This document references the following CA Technologies products:

- CA SiteMinder® SPS
- 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

The following updates have been made to the 12.52 SP1 documentation, as a result of issues found in previous releases:

Proxy Service Configuration—Updated the topic to describe how http_connection_timeout works when -Dhttp_connection_timeout is configured. The change resolves CQ 184414, 21695829 and STAR 21695829.

Authentication REST Interface (see page 146)—Updated the topic to correct the URI format. The change resolves CQ 184721 and STAR 21755360-1.

Prerequisites (see page 27)—Updated the topic to note the requirement of the ncurses package and to add that CA SiteMinder® SPS must be installed on a separate computer. The change resolves CQ 184218 and 184222. The topic also includes the requirement of opening the port 7658 and verifying that Policy Server and CA RiskMinder are running.

Implement a Filter (see page 211)—Updated the topic to correct the path to the lib directory. The change resolves CQ 183762 and STAR 21730064-1.

SPS as a Web Agent Replacement (see page 124)—Updated the topic to clarify the requirements of the Web Agent Option Pack.

Prerequisites for Using CA SiteMinder® SPS as a Web Agent Replacement (see page 125)—Deleted the note on the requirement of Web Agent Option pack installation. The change resolves CQ 181448 and STAR 21657979-1.

Generate a Self-signed Certificate (see page 165)—Updated the topic to correct the command to generate a self-signed certificate. The change resolves CQ 172433.

Download and Install the Certificates from the Certificate Authority (see page 166)—Updated the topic to add steps on how to add RootCA or self-signed certificate to ca-bundle.cert. The change resolves CQ 172433.

Debug Attribute—Corrected the example value of xmlns:nete. The change resolves CQ 184884 and STAR 21754567-02.

Configuring the CA SiteMinder® SPS Server Settings—Updated the chapter to correct the default values of enablecachepostdata, worker.ajp13.max_threads, http_connection_pool_max_size, http_connection_timeout, http_connection_stalecheck, http_connection_pool_min_size, and http_connection_pool_incremental_factor. The change resolves CQ 182068 and STAR 21652689-1.

Bookshelf—Updated the bookshelf with the following changes:

- The Administration Guide content is structured based on tasks such as Installing, Configuring.
- To facilitate better user experience on our new platform, the Online Help content of administrating CA SiteMinder® SPS using Administrative UI is merged with the Administration Guide content of administrating CA SiteMinder® SPS using backend files. The new content is available in Administration Guide, Configuration Methods of CA SiteMinder® SPS. The following changes are done as a result of this task:
  - The Online Help content that is embedded in Administrative UI is not updated with the new content and is still accessible through the Help option in Administrative UI. It contains content that describes how to administer CA SiteMinder® SPS using Administrative UI. However, the Online Help content is no longer delivered through the bookshelf.
  - The Configure the Proxy Rules and Create Virtual Hosts scenarios are now delivered as task-based content. You can find the content in Configuration Methods of CA SiteMinder® SPS.

The following updates have been made to the 12.52 documentation, as a result of issues found in previous releases

- Required patches for the Java Cryptographic Extension (JCE)—This topic details the files that require updates to use the cryptographic algorithms provided by Java. Resolves CQ 174929.
- Integrated Windows Authentication (see page 169)—This chapter details the operating systems on which Integrated Windows Authentication is supported and the ACO parameter that is required to enable Windows authentication scheme. Resolves CQs 172603 and 172605.
- Authentication and Authorization—This chapter details the AgentName format, and the authentication and authorization request formats. Resolves the CQs 177424, 173173, 172762, 172758, 172760, and 172764.

Contents

Chapter 1: Introduction to the CA SiteMinder® SPS Architecture

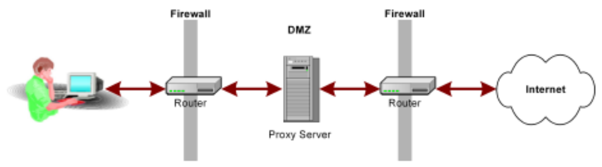
CA SiteMinder® for Secure Proxy Server (CA SiteMinder® SPS) is a stand-alone server that provides a proxy-based solution for access control. CA SiteMinder® SPS employs a proxy engine that provides a network gateway for the enterprise and supports multiple session schemes that do not rely on traditional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

This section contains the following topics:

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

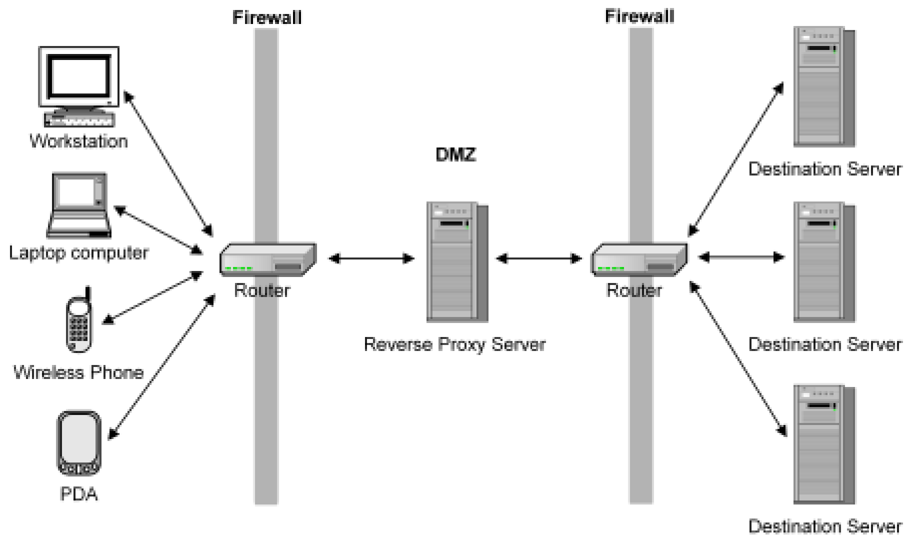
Rather than requesting a resource directly from a destination server, the reverse proxy server caches much of the content from the destination servers, providing ready access for users. This type of proxy server deployment is considered a reverse proxy, because the proxy is transparent to the user and 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

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

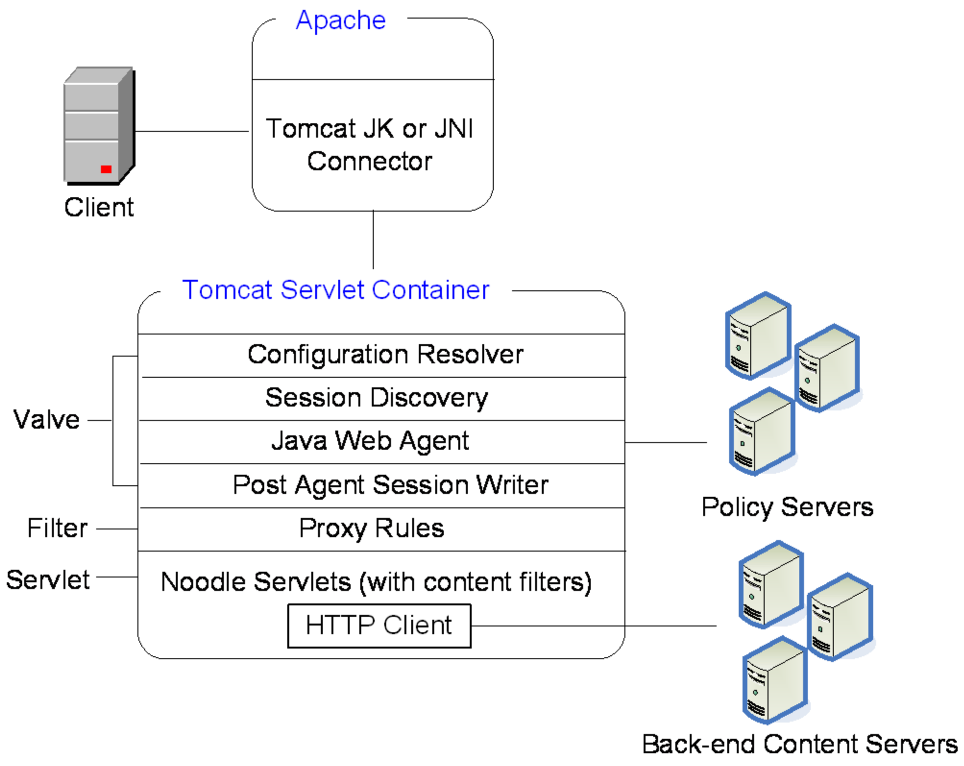
A set of configurable proxy rules determines how CA SiteMinder® SPS handles a user request. Users can access resources through multiple session schemes based on mapping between user agent types and virtual hosts. Requests can be routed to different destination servers based on the type of device being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

The following illustration shows a configuration of CA SiteMinder® SPS. Users access CA SiteMinder® SPS using various devices. CA SiteMinder® 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.

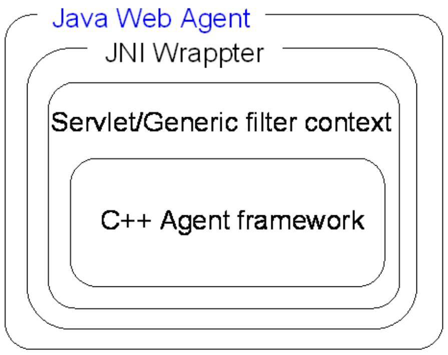


Components

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



Java Web Agent Components



CA SiteMinder® SPS architecture includes the following components:

Apache

CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® SPS include the following:

Configuration Resolver ProxyBootstrap

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

Session Discovery

The session discovery component analyzes the incoming requests for extracting CA SiteMinder® 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 CA SiteMinder® SPS session information.

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

Product Features

CA SiteMinder® SPS offers the following features:

Access Control for HTTP and HTTPS Requests

CA SiteMinder® 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, CA SiteMinder® SPS is fully integrated with SiteMinder to manage e-business transactions.

Single Sign-on

The embedded web agent in CA SiteMinder® 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. CA SiteMinder® 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

CA SiteMinder® SPS is equipped with an in-memory session store. The session store maintains session information. CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® 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, CA SiteMinder® SPS separates the corporate network and centralizes access control.

Enterprise Class Architecture

CA SiteMinder® SPS is designed to be scalable, manageable, and extensible.

Product Limitations

The following conditions apply to CA SiteMinder® SPS:

- CA SiteMinder® SPS is not a plug-in to another Web server. CA SiteMinder® SPS is a fully supported, stand-alone server.
- CA SiteMinder® SPS does not support local content. The ability to place content on CA SiteMinder® SPS is not exposed, and CA SiteMinder® SPS does not support proxy rules for providing access to local content.
- CA SiteMinder® SPS does not support having the Web server on the same system as CA SiteMinder® 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.
- CA SiteMinder® 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 CA SiteMinder® SPS, destination servers can also have SiteMinder web agents or application server agents installed. When policies for CA SiteMinder® SPS agent are inconsistent with policies for the agent installed on the destination server, CA SiteMinder® SPS can pass responses back to the invoking client. Because CA SiteMinder® SPS does not provide warnings about inconsistencies in processing such policies, use care when setting up SiteMinder policies in such environments.
- CA SiteMinder® 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, CA SiteMinder® 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.

CA SiteMinder® 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. CA SiteMinder® 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

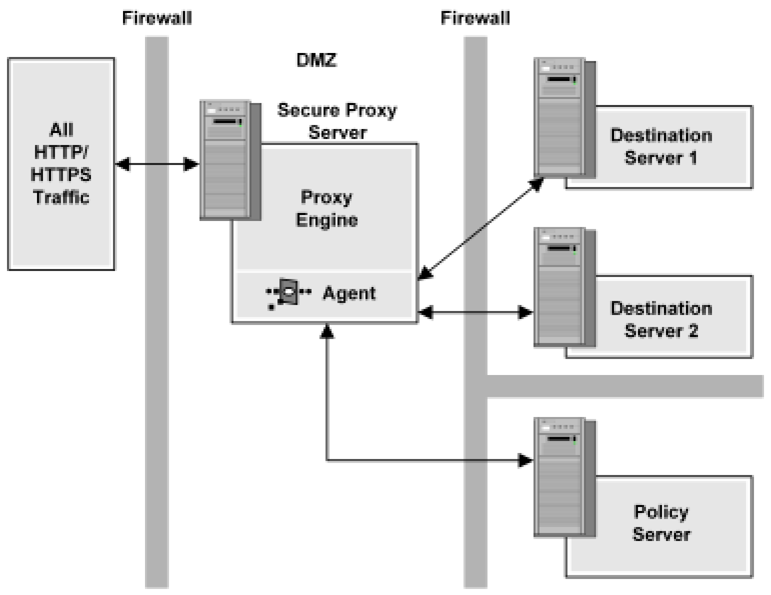
You can deploy CA SiteMinder® SPS in an enterprise to serve the following functions:

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

CA SiteMinder® SPS as a Centralized Access Control Filter

To limit access to destination servers and provide a central entry point to the network, CA SiteMinder® 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 CA SiteMinder® SPS, and forwarded to the appropriate destination server for fulfillment.

The following illustration shows how CA SiteMinder® 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 CA SiteMinder® SPS. All users must be authenticated and authorized by SiteMinder residing behind the second firewall. The destination servers provide content after SiteMinder and CA SiteMinder® SPS verify user entitlements.

This deployment provides the following benefits:

- Centralizes configuration through proxy rules

  CA SiteMinder® SPS uses proxy rules defined in XML configuration files to establish how CA SiteMinder® 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 CA SiteMinder® SPS. Based on the proxy rules established for CA SiteMinder® SPS, requests are forwarded to the appropriate destination servers for fulfillment.

- Verifies user identities and establishes entitlements

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

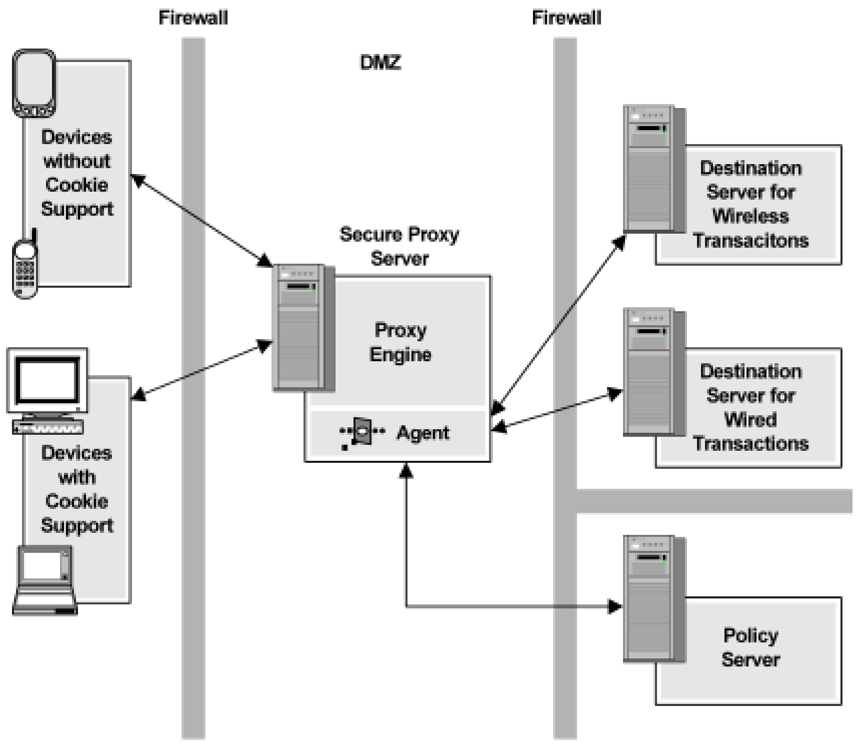
CA SiteMinder® SPS Support for Cookieless Sessions

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.

CA SiteMinder® 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 CA SiteMinder® SPS provides a combination of standard sessions using cookies and sessions without cookies:


Firewall
Firewall
DMZ
Devices without Cookie Support
Devices with Cookie Support
Secure Proxy Server
Proxy Engine
Agent
Destination Server for Wireless Transacitons
Destination Server for Wired Transactions
Policy Server

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

- Supports multiple device types

  Through a set of proxy rules, CA SiteMinder® SPS forwards, or redirects, requests based on the type of device issuing the requests. For example, all initial requests can be directed at CA SiteMinder® SPS, which forwards requests to destination servers based on device types. Browser requests can be redirected to destination servers, and CA SiteMinder® 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, CA SiteMinder® SPS provides alternatives to cookie-based sessions. CA SiteMinder® SPS maintains session information in the session store and returns a token. This token is exchanged with all transactions, allowing CA SiteMinder® 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

CA SiteMinder® 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 CA SiteMinder® SPS in a SiteMinder federation security services environment, the following process is enforced when a user request is received:

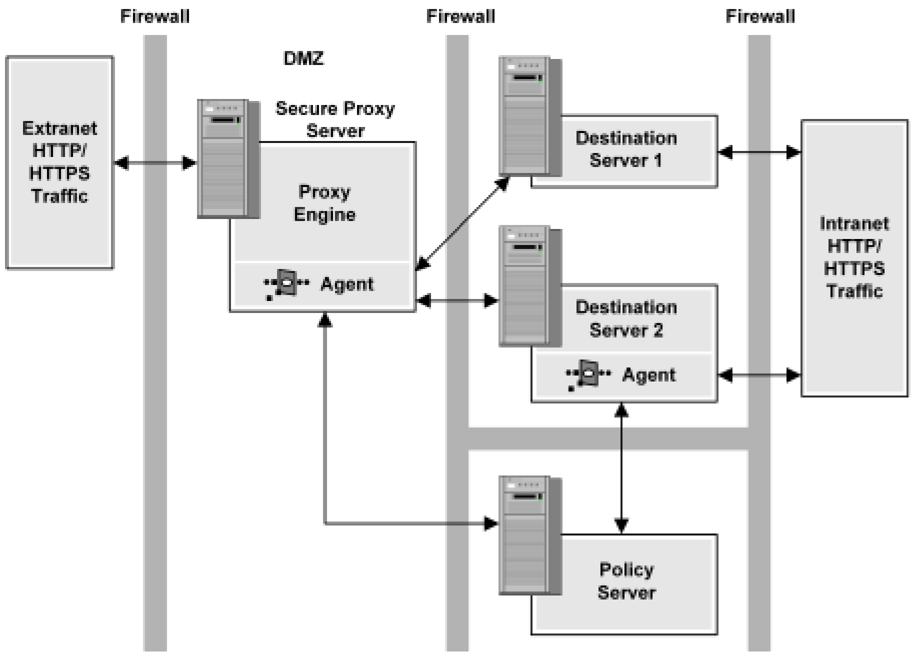
1. CA SiteMinder® SPS receives a request for a federated resource. The request is directed to the Federation Web Services (FWS) application at the site producing assertions.
2. CA SiteMinder® SPS verifies if cookieless federation is enabled for the virtual host requesting the redirect.
3. If a cookieless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

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

CA SiteMinder® SPS Support for Extranet Access Control

Another deployment of CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® 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

  CA SiteMinder® 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: Installing CA SiteMinder® SPS

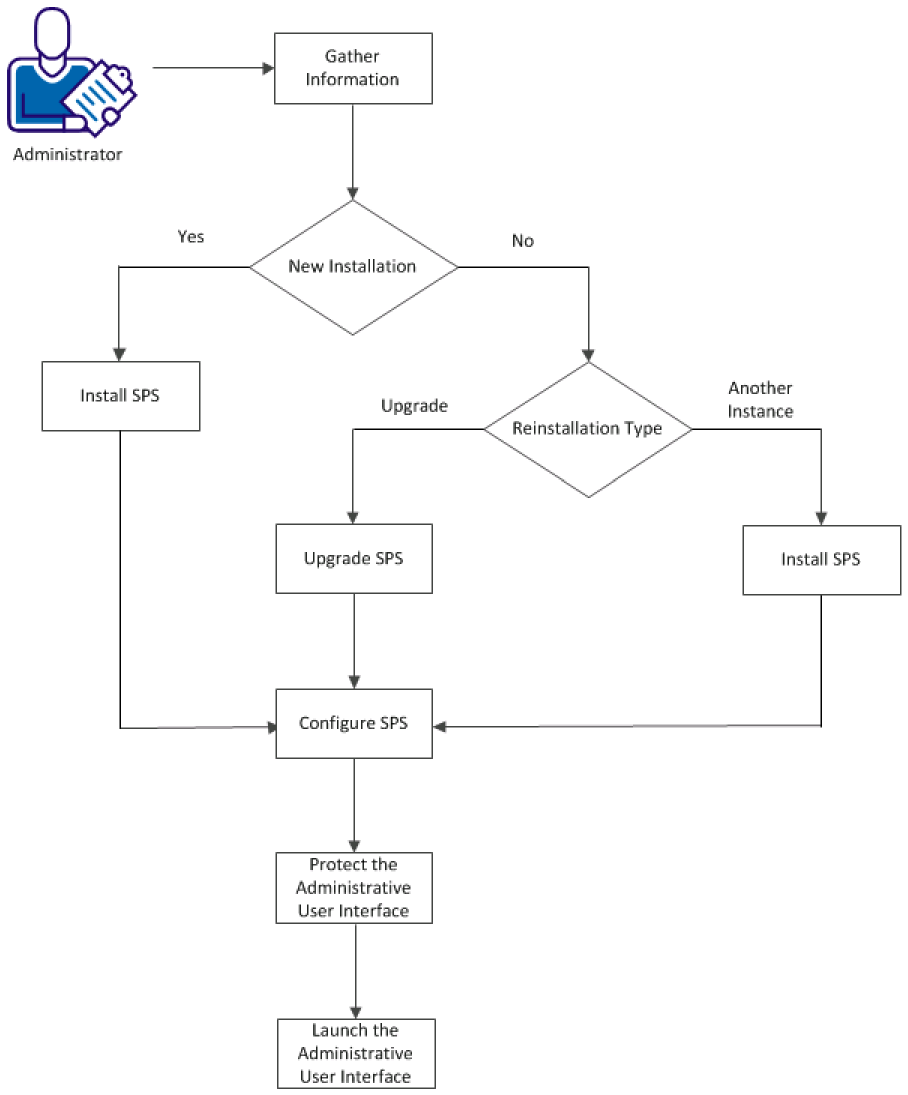
This section contains the following topics:

Install, Upgrade, and Configure SPS Manually

The CA SiteMinder for Secure Proxy Server is a stand-alone server that provides a proxy-based solution for access control. CA SiteMinder® 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 CA SiteMinder® SPS:

Install, Upgrade, and Configure SPS



Prerequisites

Before you install or upgrade CA SiteMinder® SPS, verify the following prerequisites:

- CA SiteMinder® SPS must not be installed on the computer where Policy Server is installed.
- Ensure that Policy Server is running.
- On Linux, ensure that you have write permission on the /opt/etc/CA directory.
- On Linux, the folder where you install CA SiteMinder® SPS must have sufficient permissions (755). Do not install CA SiteMinder® SPS under the /root folder, because its default permissions (750) are insufficient.
- On Solaris, CA SiteMinder® SPS runs as the "nobody" user. If you prefer not to run CA SiteMinder® SPS as this user, create an alternate user and assign the necessary permissions.
- If you are installing CA SiteMinder® SPS on RHEL, verify that you installed the ncurses package.
- If you are installing CA SiteMinder® SPS on an RHEL 5 or RHEL 6 64-bit computer, verify that you installed the following libraries on the computer:
  - libstdc++.so.6
  - libexpat.so.0
  - libuuid.so.01
  - libkeyutils.so.1

  Note: These libraries must be 32-bit binaries rather than 64-bit binaries.
- If you are installing CA SiteMinder® SPS on an RHEL 5.5 computer, verify that you installed the Legacy Software Development package on the computer.
- If you are installing or upgrading CA SiteMinder® SPS on Linux, verify that you installed the keyutils-libs-1.4-4.el6.i686.rpm package.

- **JCE**—The current Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction patches are required to use the Java cryptographic algorithms. To locate the JCE package for your operating platform, go to the Oracle website.

  Apply the patches to the following files on your system:

  - local_policy.jar
  - US_export_policy.jar

  These files are in the following directories:

  Windows: *jre_home*\lib\security

  UNIX: *jre_home*/lib/security

  jre_home

  This variable specifies the location of the Java Runtime Environment installation.

- Open port number 7658 between CA SiteMinder® SPS and Policy Server.
- Ensure that the CA RiskMinder service is running. To check the status, perform the following steps:

  Windows

  a. Open the Task Manager and verify that the arrfserver process is running.
  b. Navigate to <PolicyServer_InstallationPath>\aas\logs.
  c. Open the cariskminderstartup.log file and verify that the folllowing line exists at the end of the file:

  ```
  CA RiskMinder Service READY
  ```

  UNIX

  d. Run the ps command and verify that the arrfserver and arrfwatchdog processes are running.
  e. Navigate to <PolicyServer_InstallationPath>/aas/logs.
  f. Open the cariskminderstartup.log file and verify that the folllowing line exists at the end of the file:

  ```
  CA RiskMinder Service READY
  ```

Installation Worksheet

CA SiteMinder® 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 CA SiteMinder® SPS, verify that you gathered the following information required for host registration, embedded Apache web server and Tomcat server:

Parameter	**Description**
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.
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 Administrative UI.
Agent Configuration Object	Name of an existing Agent Configuration Object defined in 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.
Agent Name	Name of the default agent or an agent defined in the ACO.
Master Key	Identifies the master encryption key for the advanced authentication server. Enter the same value that you configured in the Policy Server.
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 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

Parameter	Description
Port number for Apache SSL requests	The port listening for SSL requests from Apache. Default: 443
Port number for Tomcat HTTP requests	The port listening for HTTP requests from Tomcat. Default: 8080
Port number for Tomcat SSL requests	The port listening for SSL requests from Tomcat. Default: 543
Port number for Tomcat shutdown requests	The port listening for shutdown requests from Tomcat. Default: 8005
Port number of AJP	The port number of AJP. Default: 8009

Install CA SiteMinder® SPS

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

Install CA SiteMinder® 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, CA SiteMinder® 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 CA SiteMinder® SPS instance.
5. Restart your system after the installation completes.

Install CA SiteMinder® SPS on Linux or Solaris

CA SiteMinder® SPS supports installations on Linux and Solaris.

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 CA SiteMinder® SPS Installation

You can check the InstallLog file to verify that CA SiteMinder® 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 CA SiteMinder® SPS

You can install multiple CA SiteMinder® SPS instances on the same computer. Each CA SiteMinder® 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 CA SiteMinder® SPS

You can run the installation program to upgrade from a previous version of CA SiteMinder® 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/ 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 CA SiteMinder® SPS version.
3. Follow the instructions from the installation wizard.
4. Restart your system after the installation completes.

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 CA SiteMinder® 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 CA SiteMinder® SPS from a release earlier to CA SiteMinder® 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.

Configure CA SiteMinder® SPS

After you install CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® SPS to act as a Federation Gateway.
11. Review the Configuration Summary
12. Click Install.

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

httpd.conf

Contains the settings for the Apache web server.

server.conf

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

logger.properties

Contains the settings that determine CA SiteMinder® SPS logging behavior.

proxyrules.xml

Contains the rules that determine how CA SiteMinder® SPS handles incoming requests.

Manage Session Assurance

By default, CA SiteMinder® SPS enables Session Assurance. If you want to disable the feature, perform the following steps:

1. Open the server.conf file.
2. Navigate to the <Context name="AALoginService"> section and set the value of enable to no.
3. Navigate to the <Context name="Advanced Auth Application"> section and set the value of enable to no.
4. Navigate to the <Context name="UI Application"> section and set the value of enable to no.
5. Save the changes.

Modify the Default Location of the SiteMinder Forms

Beginning with CA SiteMinder® 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 CA SiteMinder® 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.

Protect the Administrative User Interface

By default, the installer creates a protection policy to protect the Administrative User Interface. The installer uses the defined Agent Name to create the protection policy with the following details:

- Domain: DOMAIN-SPSPADMINUI
- Policy: POLICY-SPSADMINUI

The protection policy does not contain the user directory information. Perform the following steps to log in to the Administrative User Interface:

1. Update DOMAIN-SPSPADMINUI with the user directory information.
2. Update POLICY-SPSADMINUI with user information.

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`*`:`*`Tomcat_port`*`/proxyui/`

CA SiteMinder® 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 CA SiteMinder® SPS, see Uninstall CA SiteMinder® SPS. If you want to start CA SiteMinder® SPS in various modes, see Start CA SiteMinder® 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.

Install and Configure SPS Silently

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

After installation, CA SiteMinder® 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 CA SiteMinder® SPS

To uninstall from Windows, perform the following steps:

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

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

To uninstall from UNIX, perform the following steps:

1. Open a console window and navigate to the root installation directory.
2. Run the following program:

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

Note: If you have modified any files such as server.conf, the uninstall program does not remove these files or their parent folders automatically. You must delete the files and folders manually.

Chapter 3: Administrating CA SiteMinder® SPS

This section contains the following topics:

Configuration Methods of CA SiteMinder® SPS

You can configure CA SiteMinder® SPS settings in the following ways:

- Manually using the server.conf file.

  The server.conf file contains settings in groups of name/value pairs that CA SiteMinder® SPS reads at startup. It is located at the following location:

  ```
  installation_home/secure-proxy/proxy-engine/conf
  ```

- Administrative UI

  Administrative UI displays the most commonly used fields of each setting. To fine tune a setting in detail, use the server.conf file.

The following sections describe how to configure each setting manually and using Administrative UI.

Configure the Apache Settings

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

You can configure the settings manually or using Administrative UI.

To configure manually, perform the following steps:

1. Navigate to *installation_home*/secure-proxy/httpd/conf/.
2. Open the httpd.conf file and make the necessary changes.
3. Save the changes.
4. Restart CA SiteMinder® SPS.

To configure using Administrative UI, perform the following steps:

1. Navigate to Proxy Service, Advanced Settings.
2. Complete the following fields in the Apache Settings and httpd-mpm sections:

   JK Log Level

   Specifies what detail of logging must occur in the log file defined by JkLogFile.

   Log Level

   Specifies what detail of logging must occur in the log file defined by ErrorLog.

   Listen Port

   Defines tthe port Apache binds to for communication.

   JK Log File Path

   Defines the the path and file name of the mod_jk log file.

   Server Name

   Defines the name and port of the server.

   Error Log File Path

   Defines the location of the error log file.

3. Click Save.

Configure the Federation Settings

The federation settings enable CA SiteMinder® SPS to act as a federation gateway within a CA SiteMinder® federation network.

You can configure the settings manually or using Administrative UI.

To configure manually, perform the following steps:

1. Open the server.conf file.
2. Navigate to the federation section.
3. Configure the following parameters:

   enablefederationgateway

   Enables CA SiteMinder® SPS to act as a federation gateway proxy server.

   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.

4. Save the changes.

To configure using Administrative UI, perform the following steps:

1. Navigate to Proxy Configuration, Federation.
2. Configure the following fields:

   Enable Federation Gateway

   Enables CA SiteMinder® SPS to act as a federation gateway proxy server.

   Authentication URL

   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.

Configure the Custom Error Messages Settings

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

You can configure the settings manually or using Administrative UI.

To configure manually, perform the following steps:

1. Open the server.conf file.
2. Navigate to the <customerrorpages> section.
3. Configure the following parameters:

   enable

   Specifies how CA SiteMinder® 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. CA SiteMinder® SPS reads the value at the startup.

   locale_type

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

   showcallstack

   Specifies if the call stack must be retrieved to track exceptions that you want to debug. If the value is set to true, CA SiteMinder® 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.

   Important! For security reasons, we 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.

To configure using Administrative UI, perform the following steps:

1. Navigate to Administration, Custom Error Pages.
2. Select Enable Custom Error Pages.
3. Set the following locale settings:

   Locale Type

   Specifies the locale of the custom error pages. If you select Server, the CA SiteMinder® SPSses the locale settings of the CA SiteMinder® SPS server to display the custom error pages. If you select Web browser, CA SiteMinder® SPS uses the locale settings of your browser to display the custom error messages.

LocaleCountry

Specifies the locale country.

Locale Language

Specifies the language.

4. Click Save.

Modify the Default Custom Error Pages

By default, CA SiteMinder® SPS provides a list of possible request errors that are received from CA SiteMinder® 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 CA SiteMinder® SPS or web server returns when a user request fails.

Limits: 100 characters

error message|URL

Defines the error message CA SiteMinder® 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

Modify the Default CA SiteMinder® SPS Error Pages

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

The following table lists the default errors that CA SiteMinder® SPS displays due to inappropriate configuration settings of the CA SiteMinder® 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 SPS
SessionUpdateError	The session store might have been corrupted during update. Restart the SPS CA SiteMinder® SPS
WebAgentException	CA SiteMinder® SPS might not be communicating with the Web agent. For more information about the error, see CA SiteMinder® SPS logs
Noodle_GenericException	There might be an error during processing a request at noodle stage. For more information about the error, see CA SiteMinder® 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

Error Code	Error Message
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	CA SiteMinder® SPS does not support the method
Noodle_ProxytoProxyNotSupported	CA SiteMinder® 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
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 CA SiteMinder® SPS logs

You can modify the error message of an error page manually or using Administrative UI.

To modify manually, perform the following 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 changes.
4. Save the changes.

To modify using Administrative UI, perform the following steps:

1. Navigate to Administration, Custom Error Messages.
2. Edit the error message of the error code in CA SiteMinder® SPS Custom Error Messages.
3. Click Save.

Modify the Web Server Error Pages

By default, CA SiteMinder® 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.

You can add, modify, or delete an error code or error message of a web server error page manually or using Administrative UI. If you delete the error message of an error code, the CA SiteMinder® SPS displays a page with the following message:

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

To manage manually, perform the following 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 changes.
4. Save the changes.

To manage using Administrative UI, perform the following steps:

1. Navigate to Administration, Custom Error Messages.
2. To edit an error message, perform the following steps:
   a. Click the Edit icon of the error message in Webserver Custom Error Messages.
   b. Modify the message.
   c. Click OK and Save.
3. To delete an error code, error code, perform the following steps:
   a. Click Actions, Delete of the error code you want to delete in Webserver Custom Error Messages.
   b. Click Save.
4. To add an error message to the default webserver error codes, perform the following steps:
   a. Click Add, and enter the error message details.
   b. Click Add.

Configure the Proxy Service Settings

The following proxy services are pre-defined in CA SiteMinder® SPS

forward

Forwards requests to destination servers according to the conditions and cases defined in the proxy rules XML configuration file. CA SiteMinder® SPS handles the subsequent requests.

You can configure the forward proxy service to manage the connection pool of CA SiteMinder® SPS. You can improve server performance by maintaining connections and alleviating the overhead of establishing a new connection for each request to a destination server. You can define proxy filters to process tasks before a request is forwarded to a destination server, and after the destination server returns data to CA SiteMinder® SPS.

redirect

Redirects requests to destination servers. The destination server handles the subsequent requests. You cannot configure the redirect proxy service.

To configure the forward proxy service manually, perform the following steps:

1. Open the server.conf file.
2. Navigate to the <Service name="forward"> section.
3. Configure the following parameters:

   class

   Specifies the implementation that provides forwarding services for the CA SiteMinder® 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.

   protocol.multiple

   Indicates whether CA SiteMinder® 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 CA SiteMinder® 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.

   http_connection_pool_max_size

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

Important! Each connection established by CA SiteMinder® 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 CA SiteMinder® SPS opens when all available connections are being used to process requests.

http_connection_pool_connection_timeout_unit

Sets the timeout unit to seconds or minutes.

http_connection_pool_connection_timeout

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

http_connection_pool_wait_timeout

Defines the time, in milliseconds, that CA SiteMinder® SPS waits for an available connection. The default, 0, specifies that CA SiteMinder® 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.

http_connection_timeout

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

Note

- The timeout explicitly refers to the HTTP connection and not to the connection pool.
- If you configured the -Dhttp_connection_timeout parameter in the SmSpsProxyEngine.properties file during CA SiteMinder® SPS initiation, the value of -Dhttp_connection_timeout precedes the value of http_connection_timeout.

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.

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. CA SiteMinder® 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 CA SiteMinder® 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"

To configure using Administrative UI, perform the following steps:

1. Navigate to Proxy Configuration, Proxy Service.
2. Complete the following fields:

Support Multiple Protocols

Indicates whether the CA SiteMinder® SPS supports protocols other than HTTP. If you select this option, the CA SiteMinder® SPS supports HTTP and HTTPS protocols. If you do not select this option, the CA SiteMinder® SPS supports only the HTTP protocol. By default, this option is enabled.

Minimum Size

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

Maximum Size

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

Important! Each connection established by the CA SiteMinder® 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.

Incremental Factor

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

Connection Timeout

Defines the time and timeout unit the system waits before closing idle connections in the connection pool.

Wait Timeout

Defines the time, in milliseconds, that the CA SiteMinder® SPS waits for an available connection. If the value is 0, the CA SiteMinder® SPS waits for a connection until notified and invalidates the use of HTTP Connection Pool Max Attempts.

Maximum Attempts

Indicates the number of attempts that the CA SiteMinder® SPS makes to obtain a connection. This directive is only applicable if wait timeout is not zero. If the value is 0, the CA SiteMinder® SPS makes attempts indefinitely.

Connection Time Out

Defines the time, in milliseconds, spent on host name translation and establishing the connection with the server when creating sockets. If the value is 0, the CA SiteMinder® SPS does not enforce a limit.

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

3. Click Save.

Configure the Session Scheme Settings

A session scheme determines how a user identity is maintained and provides single sign-on during a session.

CA SiteMinder® SPS provides the following out-of-the-box session schemes that you can configure:

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

You can create custom session schemes too. You can define multiple session schemes. You can associate session schemes with user agent types for each configured virtual host, and create a session scheme mapping. Each session scheme must be associated with a Java class file that defines the session behavior. If you do not define a session scheme for a user agent, the default session scheme is used.

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 CA SiteMinder® 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. 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. 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.

Configure the Default Session Scheme

The default session scheme is the scheme that CA SiteMinder® SPS uses to establish and maintain user sessions when no other scheme is specified for a user agent type. You must configure a default session scheme. You can configure the default session scheme to use any available session scheme.

You can configure the default session scheme manually or using Administrative UI.

To configure manually, perform the following steps:

1. Open the server.conf file.
2. Navigate to the <SessionScheme name="default"> section.
3. Configure the following parameters:

   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 through the user agent type maintain session using traditional CA SiteMinder® cookies. CA SiteMinder® uses cookies to track sessions so a cookies scheme is supported by CA SiteMinder® SPS. Select the option to let the session scheme accept the SMSESSION cookies.

4. Save the changes.

To configure using Administrative UI, perform the following steps:

1. Navigate to Virtual Hosts, Sessions Schemes.
2. Click Actions, Edit against the default session scheme in the Available Session Schemes list.
3. Configure the following parameters:

   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 through the user agent type maintain session using traditional CA SiteMinder® cookies. CA SiteMinder® uses cookies to track sessions so a cookies scheme is supported by CA SiteMinder® SPS. Select the option to let the session scheme accept the SMSESSION cookies.

4. Click OK and Save.

Configure the SSL ID Session Scheme

A secure sockets layer (SSL) connection includes a unique ID that is created when an SSL connection is initiated. CA SiteMinder® SPS can use the unique ID as a token to refer to the session information of a user that is maintained in the in-memory session store. This scheme eliminates cookies as a mechanism for maintaining a user session. SSL ID session schemes can be supported using the Java classes that are packaged with CA SiteMinder® SPS.

Important! The limitation of the scheme is that the initial contact with CA SiteMinder® SPS establishes an SSL session ID. If a user SSL session is interrupted, a new SSL connection is established. The user must be re-authenticated and re-authorized because the new SSL connection connects to a new server though it is a virtual server on the same system. Also, forms that are used by HTML Forms Authentication Schemes must be served from the same host name as the protected resource.

You can configure the settings manually or using Administrative UI.

To configure the settings manually, perform the following steps:

1. Open the server.conf file.
2. Navigate to the <SessionScheme name="ssl_id"> section.
3. Configure the following settings:

   class

   Specifies the Java class that handles SSL ID session schemes.

   Default: com.netegrity.proxy.session.SSLIdSessionScheme

   accepts_smsession_cookies

   Specifies if SMSESSION cookies are accepted.
4. Save the changes.

To configure using Administrative UI, perform the following steps:

1. Navigate to Virtual Hosts, Sessions Schemes.
2. Click Actions, Edit against the ssl_idt session scheme in the Available Session Schemes list.
3. Configure the following parameters:

   Class

   Specifies the Java class that handles SSL ID session schemes.

   Default: com.netegrity.proxy.session.SSLIdSessionScheme

   Accepts SMsession Cookies

   Specifies if SMSESSION cookies are accepted.
4. Click OK and Save.

You must enable SSL in the httpd.conf file of the Apache Web Server to configure the SSL ID session scheme. To modify httpd.conf, perform the following steps:

1. Open the httpd.conf file located in the directory installation_home/secure-proxy/httpd/conf.
2. Locate the following line:

   ```
   #SSLOptions +StdEnvVars +ExportCertData +CompatEnvVars
   ```
3. Delete the # symbol at the beginning of the line.
4. Delete +CompateEnvVars from the line so that the line reads as follows:

   ```
   SSLOptions +StdEnvVars +ExportCertData
   ```
5. Save the changes.
6. Restart CA SiteMinder® SPS.

Configure the IP Address Session Scheme

In environments where IP addresses are fixed, you can use an IP address to refer to a user 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.

Configure the Mini-cookies Session Scheme

The mini-cookies session scheme stores session information in the CA SiteMinder® SPS in-memory session store and creates a cookie that contains an encrypted token that the CA SiteMinder® SPS returns to the user. The mini-cookie is a fraction of the size of a standard SiteMinder cookie and thus decreases the cost of access for devices such as wireless phoness. It provides an alternative for standard SiteMinder cookies.

You can configure the settings manually or using Administrative UI.

To configure the settings manually, perform the following steps:

1. Open the server.conf file.
2. Navigate to the <SessionScheme name="minicookie"> section.
3. Configure the following settings:

   class

   > Specifies the Java class that handles the session scheme. If you want to use the default mini-cookies session scheme, do not modify the value.
   >
   > Default: com.netegrity.proxy.session.MiniCookieSessionScheme

   accepts_smsession_cookies

   > Specifies if SMSESSION cookies are accepted.

   cookie_name

   > Defines 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 the CA SiteMinder® SPS instances that provide single sign-on.

4. Save the changes.

To configure using Administrative UI, perform the following steps:

1. Navigate to Virtual Hosts, Sessions Schemes.
2. Click Actions, Edit against the minicookie session scheme in the Available Session Schemes list.
3. Configure the following parameters:

   Class

   > Specifies the Java class that handles the session scheme. If you want to use the default mini-cookies session scheme, do not modify the value.
   >
   > **Default**: com.netegrity.proxy.session.MiniCookieSessionScheme

   Accepts SMsession Cookies

   > Specifies if SMSESSION cookies are accepted.

   cookie_name

   > Defines 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 the CA SiteMinder® SPS instances that provide single sign-on.

4. Click OK and Save.

Configure the Simple URL Rewriting Session Scheme

Simple URL rewriting appends a token to the requested URL and tracks a user session. The token is used to retrieve session information from the in-memory session store.

The simple_url schemes support simple URL rewriting that 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 CA SiteMinder® 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 the links are absolute, the simple URL rewriting scheme fails. Also, the token that CA SiteMinder® SPS appends to a URL is stripped from the URL when the request is forwarded. The token is appended only at the CA SiteMinder® SPS interaction level so that it does not interfere with back-end server processing.

You can configure manually or using Administrative UI.

To configure manually, perform the following steps:

1. Open the server.conf file.
2. Navigate to the <SessionScheme name="simple_url"> section.
3. Configure the following settings:

   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.

session_key_name

Specifies the SiteMinder ID (SMID) session identifier.

To configure using Administrative UI, perform the following steps:

1. Navigate to Virtual Hosts, Sessions Schemes.
2. Click Actions, Edit against the simple_url session scheme in the Available Session Schemes list.
3. Configure the following parameters:

   Class

   Specifies the Java class that handles the session scheme. If you want to use the cookieless rewriting session scheme session scheme, do not modify the value.

   Default: com.netegrity.proxy.session.SimpleURLSessionScheme

   Accepts SMsession Cookies

   Specifies if SMSESSION cookies are accepted.

   session_key_name

   Specifies the SiteMinder ID (SMID) session identifier.

4. Click OK and Save.

Enable Cookieless Federation for Rewriteable Session Schemes

Configure cookieless federation to let CA SiteMinder® SPS use rewritable session schemes such as simple URL session scheme in a federated environment.

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

Follow these steps:

1. Open the server.conf file.
2. Add the following code to the virtual host section for the virtual host that is serving FWS.

   ```
   cookielessfederation="yes"
   ```
3. Save the change.
4. Restart CA SiteMinder® SPS.

Note: No separate post filter, such as the CookielessFedFilter needs to be enabled for the CA SiteMinder® 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 CA SiteMinder® SPS is not acting as a federation gateway.

Rewrite FWS Redirects for Simple URL Session Schemes

If you deploy CA SiteMinder® SPS in a federated environment, one of the session schemes you can use at the site that is 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 CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® 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.

Configure the 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. CA SiteMinder® SPS can use this device ID as a token to refer to session information in the session store.

You can configure the settings manually or using Administrative UI.

To configure manually, perform the following steps:

1. Open the server.conf file.
2. Navigate to the <SessionScheme name="device_id"> section.
3. Configure the following parameters:

   class

   Specifies the Java class that handles the session scheme.

   Default: com.netegrity.proxy.session.DeviceIdSessionScheme

   accepts_smsession_cookies

   Specifies if SMSESSION cookies are accepted.

   device_id_header_name

   Defines the vendor specific device ID.

4. Save the changes.

To configure using Administrative UI, perform the following steps:

1. Navigate to Virtual Hosts, Sessions Schemes.
2. Click Actions, Edit against the device_id session scheme in the Available Session Schemes list.
3. Configure the following parameters:

   Class

   Specifies the Java class that handles the session scheme.

   Default: com.netegrity.proxy.session.DeviceIdSessionScheme

   Accepts SMsession Cookies

   Specifies if SMSESSION cookies are accepted.

   device_id_header_name

   Defines the vendor specific device ID.

4. Click OK and Save.

Configure the Session Store Settings

The session store settings determine how CA SiteMinder® SPS stores user sessions.

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

You can configure the settings manually or using Administrative UI.

To configure manually, perform the following steps:

1. Open the server.conf file.
2. Navigate to the SessionStore section.
3. Configure the following fields:

   max_size

   Defines the maximum number of concurrent user sessions that CA SiteMinder® SPS stores in the in-memory session.

   clean_up_frequency

   Defines the interval in seconds that CA SiteMinder® SPS waits before deleting expired sessions stored in the session store cache.

4. Save the changes.

To configure using Administrative UI, perform the following steps:

1. Navigate to Proxy Configuration, Session Store.
2. Complete the following fields:

   Max Sessions to Store

   Defines the maximum number of concurrent user sessions the CA SiteMinder® SPS stores in the in-memory session.

   Cleanup Frequency

   Defines the interval, in seconds, that the CA SiteMinder® SPS waits before cleaning out expired sessions residing in the session store cache.

3. Click Save.

Configure the Tomcat Settings

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

You can configure the settings manually or using X UI.

To configure manually, perform the following steps:

1. Open the abc.conf file and navigate to the Serversection.
2. Configure the following fields:

   parameter 1

      Description.

   parameter 2

      Description.

   parameter 3

      Description.

   parameter 4

      Description.

3. Save the changes.

To configure using X UI, perform the following steps:

1. Navigate to Path.
2. Complete the following fields in the Tomcat Settings section:

   parameter_1

      Description.

   parameter_2

      Description.

   parameter_3

      Description.

   parameter_4

      Description.

3. Click Save.

Configure the User Agents Settings

User agents define the device types such as web browsers, wireless phones, and PDAs by which users can access network resources. CA SiteMinder® SPS uses the header attribute to identify a user agent type when it receives a user request. You must define a user agent before you can associate the user agent with a session scheme to manage a virtual host.

You can define a user agent manually or using Administrative UI.

To define a user agent manually, perform the following steps:

1. Open the server.conf file.
2. Navigate to the <UserAgent> section.
3. Configure the following parameters:

   user_agent_name

   Defines the name of the user agent.

   header_name

   Contains the user-agent header of an HTTP request. The value indicates the device type that is requesting access. You can use regular expressions and provide a partial name as part of the expression.
4. Save the changes.

To configure using Administrative UI, perform the following steps:

1. Navigate to Virtual Hosts, User Agents and click Add.
2. Type the name of the user agent in Name.
3. Click Add in the Headers table.
4. Type the header name and value in Name and Value.
5. Click OK and Save.

Configure the Virtual Host Settings

The virtual host settings let CA SiteMinder® SPS act as a virtual host. You must define one default virtual host and can define multiple virtual hosts. By default, CA SiteMinder® SPS provides default virtual host settings that can be used for all the virtual hosts.

If you want to override the default virtual host settings for a virtual host, create a virtual host with the new values. If you do not define virtual host settings during the virtual host creation, CA SiteMinder® SPS uses the default value that is defined in the default virtual host settings.

Default Virtual Host Values

The default virtual host settings consist of the following sections:

- Virtual host details
- Default session scheme
- Session scheme mappings
- Web Agent configuration

Virtual Host Details

The following parameters define the virtual host:

Note: The parameter names are represented as they appear in the server.conf file and Administrative UI, respectively.

enablerewritecookiepath/Enable Rewrite Cookie Path

Rewrites the cookie path to the URI that the backend server set when it received the initial request from the client. This ensures that the backend server does not reset the cookie path to its own resource URI and the browser contains the correct cookie when the client sends subsequent requests.

enablerewritecookiedomain/Enable Rewrite Cookie Domain

Rewrite the cookie domain from the domain set to the domain that the backend server set when it received the initial request from the client.

enableproxypreservehost/ Enable Proxy Reserve Host

Preserves the HTTP HOST header file and sends it to the backend server.

When you enable the parameter, it takes precedence over a filter that is configured to control the HTTP HOST header. To disable the parameter and let the filter take precedence over the parameter, perform the following steps:

1. Open the server.conf file.
2. Add the following parameter in the Virtual Host section of the virtual host you want to configure:

   ```
   filteroverridepreservehost
   ```
3. Set the value of filteroverridepreservehost to yes.

Note You can enable filteroverridepreservehost only if a filter is available to control the HTTP HOST header.

requestblocksize/Request Block Size

Defines the block size of the request data that must be read at a time before the data is sent to the backend server. You can configure different values for each virtual host that you configure.sizes.

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.

responseblocksize/Response Block Size

Defines the block size of the response data that must be read at a time before the data is sent from the backend server to the user. You can configure different value for each virtual host you configure.

Limits: 1KB to approximately 352000 KB.

Note: You must define the block sizes in proportion to the available and allocated JVM heap size for the CA SiteMinder® SPS java process. Use large block sizes for large file transfers. Perform the following steps to define the 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.

Default Session Scheme

The default session scheme defines the session scheme that the virtual host uses by default.

Session Scheme Mapping

Session scheme mappings associate session schemes with user agent types. Map the defined user agent types with the defined session schemes.

The following parameters define the session scheme mapping:

Note: The parameter names are represented as they appear in the server.conf file and Administrative UI, respectively.

user_agent_name/User Agent Name

Identifies the user agent name that you want to map.

session_scheme_name/ Session Scheme Name

Identifies the session scheme that must be mapped.

Web Agent Configuration

The WebAgent.conf file defines the default web agent configuration. If you want to uses local configuration, you can point the WebAgent.conf file 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 Web Agent for a new virtual host, perform the following steps:

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.
3. 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.
4. Use a text editor to modify WebAgent.conf to reflect the new agent configuration object. Verify that the Web Agents have different log files.
5. 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.

6. 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.

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 CA SiteMinder® SPS with a redirection.

Note: A redirection that is the result of a request to the CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® SPS must be configured to handle the redirected requests.

Default Virtual Host

To let CA SiteMinder® SPS to act as a virtual host for one or more host names, you must define a virtual host as the default virtual host. You can define multiple virtual hosts.

To manually configure a default virtual host, modify the <VirtualHost name="default"> section in the server.conf file. To configure the default virtual host using Administrative UI, edit the default virtual host settings in the Virtual Hosts, Available Virtual Hosts page.

Create Virtual Host

You can define multiple virtual hosts and configure them to different settings other than the default virtual host values.

To create a virtual host manually, perform the following steps:

1. Open the server.conf file.
2. Create a virtual host section in the server.conf file with the fields as described in the default virtual host values.

   Note: If you do not define any settings, its default value is considered from the default virtual host values.
3. Save the changes.

To create a virtual host using Administrative UI, perform the following steps:

1. Navigate to Virtual Hosts, Virtual Hosts.
2. Click Add.
3. Follow the steps on the wizard.
4. Click OK.

Configure Log Settings

You can configure the CA SiteMinder® SPS and HttpClient log settings.

Configure CA SiteMinder® SPS Logs

You can configure the server log settings and log rolling settings manually or using Administrative UI.

To configure manually, perform the following steps:

1. Navigate to the following location:

   ```
   sps_home/Tomcat/properties
   ```

2. Open the logger.properties file.
3. To log events on a console, navigate to the SvrConsoleAppender section and set the following parameter:

   log_message_display_format_on_console

   Defines the display format of a log message on the console. You can define any log4j date pattern strings.

4. To log events in to a file, navigate to the SvrFileAppender section and set the following parameter:

   log_message_display_format_in_file

   Defines display format of a log message in the file. You can define any log4j date pattern strings.

5. Configure the following fields to define the logging settings:

   log level

   Defines 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.

   output format

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

   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
```

6. Perform one of the following steps:
   - To define log rolling based on file size, perform the following steps:

     logfile path

     Specifies the name and path of the log file.

     Default Name: server.log

     Default Path: install_dir_home/secure-proxy/proxy-engine/logs/

     true|false

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

     MaxFileSize

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

     MaxBackupIndex

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

   - To define log rolling based on file age, perform the following steps:

     date_pattern

     Specifies the date when the system must create a new log file. A new log file is created in the <logfile_name>.<date_format>.

7. Save the changes.

To configure using Administrative UI, perform the following settings:

1. Navigate to Administration, Logger.
2. Configure the following settings to define the logging settings:

   Log to File

   Specifies that the logs must be stored in a file. Select this option if you want to log messages in a file. By default, this option is enabled.

   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.

File Log Pattern

Defines the display format of a log message in the file. You can define any log4j date pattern strings.

3. To define log rolling based on file size, configure the following fields:

Append to Log File

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

Log Rotation Type

Specifies the log rolling mechanism.

Maximum File Size

Specifies the maximum size of the log file after which a new log file is created. This option is displayed only if you selected Log Rotation Type as onFileSize.

Maximum Backup Index

Defines the maximum number of log files that must be created. If the number of log files exceeds the maximum number that is specified, the oldest log file is deleted and a new log file is created. This option is displayed only if you select Log Rotation Type as onFileSize.

Rollover Schedule

Specifies the class name that defines the rollover schedule based on the selected log rotation type.

Default for OnAge Log Rotation Type:
org.apache.log4j.DailyRollingFileAppender

Default for OnSize Log Rotation Type:
org.apache.log4j.DailyRollingFileAppender

4. To define log rolling based on file age, configure the following fields:

Date Pattern

Specifies the date when a logfile must be created. You can define any log4j date pattern strings. This option is displayed only if you select Log Rotation Type as onAge.

Log File Name

Specifies the name of the log file.

5. Click Save.

Configure HttpClient Logging

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

To configure the settings manually, perform the following settings:

1. Open the server.conf file.
2. Navigate to the <Server> section and set the httpclientlog value to yes.
3. Save the change.
4. Navigate to the following location:

   ```
   sps_home\Tomcat\properties
   ```
5. Open the httpclientlogging.properties file.

6. Configure the following parameters:

   java.util.logging.FileHandler.formatter

   Defines 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`

   java.util.logging.FileHandler.pattern

   Defines 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

   Specifies the number of output files in a cycle.

   java.util.logging.FileHandler.limit

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

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

7. Save the changes.

To configure using Administrative UI, perform the following steps:

1. Navigate to Administration, Logger.
2. Complete the following fields in the Http Client Logging section:

   Maximum File Size

   Defines the maximum number of bytes to write to any on log file.

   Maximum Backup Index

   Specifies the number of output files in a cycle.

   Wire Level

   Defines the log level of the wire log.

   Log File Name

   Defines the file name of the HttpClient log file.

   Formatter Class Name

   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

3. Click Save.

Configure Proxy Rules

CA SiteMinder® SPS routes uses the proxy engine that is built into the server to route requests to appropriate servers in the enterprise. 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 how the requests are forwarded or redirected to resources that are located on destination servers. A set of proxy rules is defined according to the proxy rules DTD that is installed by default, and is stored in a proxyrules.xml configuration file. By default during installation, the following relative path to the proxyrules.xml file is generated and used in the rules_file parameter of the <ServiceDispatcher> section of the server.conf file:

```
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. The changes to the proxyrules.xml file are detected automatically.

Note: CA SiteMinder® SPS supports serving requests sent to the Multi-Byte Character Set (MBCS) URL of the back-end server satisfying the request.

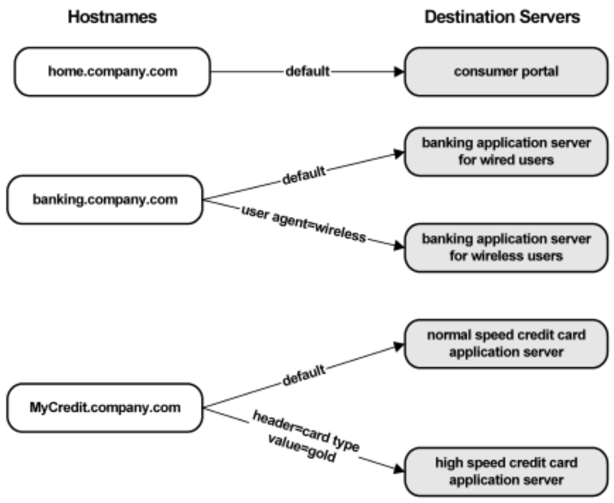
If an error is detected in the proxy rules when parsing the rules, CA SiteMinder® SPS records the error in the server log that is configured in the logger.properties file, ignores the changes, and uses the existing proxy rules.

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.

Planning Routes for Incoming Requests

Before you configure the proxyrules.xml file, map 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 use a HTTP header value to determine the destination server that will fulfill the request. CA SiteMinder® SPS can provide access to a number of virtual hosts.

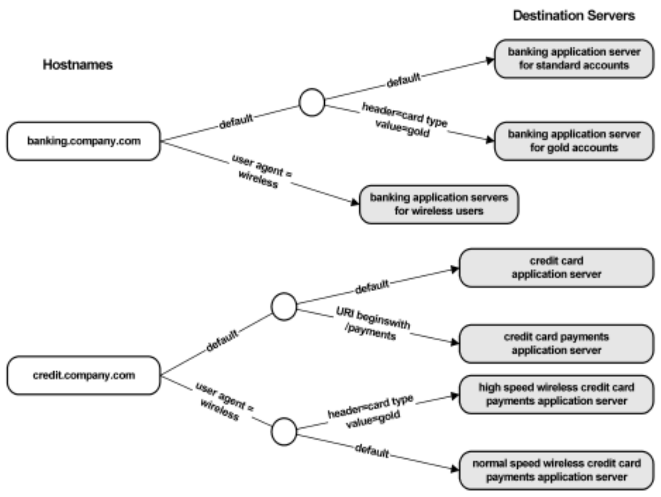
The following illustration shows how proxy rules can be used to route requests to appropriate destination servers.


Hostnames
Destination Servers
home.company.com
default
consumer portal
banking.company.com
default
banking application server for wired users
user agent=wireless
banking application server for wireless users
MyCredit.company.com
default
normal speed credit card application server
header=card type value=gold
high speed credit card application server

The proxyrules.xml file is processed from top to bottom. 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 that is 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 proxyrules.xml file is 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 that are nested within parent conditions:


Destination Servers
Hostnames
banking.company.com
default
default
banking application server for standard accounts
header=card type value=gold
banking application server for gold accounts
user agent = wireless
banking application servers for wireless users
credit.company.com
default
default
credit card application server
URI beginswith /payments
credit card payments application server
user agent = wireless
header=card type value=gold
high speed wireless credit card payments application server
default
normal speed wireless credit card payments application server

Proxy Rules Terminology

The following terms are used to describe the proxy rules:

Destinations

Defines a destination URL that fulfills a request. CA SiteMinder® 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

Defines a condition attributes of a request that determines the destination of the request. Each condition may have many cases that must be evaluated to direct the request to an 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 can include the following options:

URI

Specifies that the portion of the requested URL after the hostname must be used to determine how to route a request. As per the criteria described in the DTD, use a portion of a URI, such as the file extension of the requested resource, to route requests.

Query String

Specifies that the query string portion of a URI must be used to determine how to route a request. The query string includes all the characters following a '?' in the request.

Host

Specifies that the requested server hostname must be used 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

Specifies that the value of any HTTP header must be used 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 CA SiteMinder® responses may be used to determine how to route a request.

Cookie

Specifies that the existence or value of a cookie must be used to determine how to route a request. If a cookie value is encoded, specify the encoding scheme in encoding parameter. Only the base64 encoding scheme is supported.

Cases

Defines a set of values for conditions that provide the information for determining the 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.

Proxy Rules DTD

Use the proxy rules DTD to configure the proxy rules. The DTD file is available in the following location:

```
sps_home\secure-proxy\proxy-engine\conf\dtd
```

The following elements are configurable in DTD:

- nete:proxyrules
- nete:description
- nete:case
- nete:cond
- nete:default
- nete:forward
- nete:redirect
- nete:local
- nete:xprcond

nete:proxyrules

The nete:proxyrules is the mandatory root element in proxyrules.xml and has the following syntax:

```
<!ELEMENT nete:proxyrules (nete:description?,(nete:cond | nete:forward |
nete:redirect | nete:local)) >
```

You can define an optional nete:description element and one of the following elements:

- nete:cond
- nete:xprcond
- nete:forward
- nete:redirect
- nete:local

Debug Attribute

The debug attribute lets you manage logging and debug proxy rules. It has the following syntax:

```
<ATTLIST nete:proxyrules
debug (yes|no) "no"
```

Set the value to yes to enable logging. The log file location is determined by the TraceFileName parameter in the agent configuration object that you configured for CA SiteMinder® SPS. The TraceConfigFile parameter in the same agent configuration object must point to the Secure Proxy-specific trace logging configuration file. By default, the file is located at <install-dir>\proxy-engine\conf\defaultagent\SecureProxyTrace.conf.

For example: <nete:proxyrules xmlns:nete="http://www.ca.com/" debug="yes">

nete:description

The nete:description element 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.

It has the following syntax:

```
<!ELEMENT nete:description (#PCDATA)>
```

It can be a child of the nete:proxyrules element and may contain any description.

nete:case

The nete:case element defines the destination associated with a specific value for a condition defined in the nete:cond parent element. CA SiteMinder® SPS uses the value of the nete:case element to evaluate a case.

It has the following syntax:

```
<!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

Allows you to nest multiple conditions in a set of proxy rules.

nete:xprcond

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 CA SiteMinder® 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. CA SiteMinder® 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, CA SiteMinder® SPS uses a system for maintaining some part or all of the URI specified by a user. The URI points to a resource that lies on a destination server and must be interpreted to fulfill a request.

Either of the following may be appended to the URL specified in a forward or redirect destination:

$0

Appends the entire URI string from the user 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

Indicates that everything to the right of the matching text is appended to the forwarded or redirected request. Use it in nete:case elements where the parent nete:cond element specifies a URI substring match using the begins with comparison.

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: As the 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 nete:cond element specifies a condition that is evaluated to determine how the incoming requests is handled. You must include an attribute that must be evaluated.

It has the following syntax:

```
<!ELEMENT nete:cond (nete:case+,nete:default)>
```

You can define the following attributes:

```
<!ATTLIST nete:cond type (header | host | uri | query | cookie) #REQUIRED criteria
(equals | beginswith | endswith | contains | exists) "equals" headername CDATA
#IMPLIED>
```

header

Specifies that an HTTP header is used to determine the destination of a request. You must define a header name too. Conditions that use headers as the comparison require an additional argument of headername="value", where value is the name of the HTTP or CA SiteMinder® header.

Note: HTTP headers generated by CA SiteMinder® responses may be specified in the nete:cond elements.

For example:

```
<nete:cond type="header" headername="USER_AGENT">
```

This element indicates that a header is used, and that USER_AGENT is the header to be evaluated.

host

Specifies a host name in deployments where multiple virtual hosts are used. 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 that 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. Only the base64 encoding scheme is supported and cookie creation is not supported. 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 base64 encoding algorithm is used to decode the cookie value received from a httprequest and the result of the decoded value is compared 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 specified plain text is accepted as the cookie value and the nete:cond is verified.
- 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 specified encoded cookie value is accepted as plain text, and the plain text cookie value is used 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.

If no criteria are specified, equals is assumed the default criteria.

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.

Each nete:cond element must have one or more nete:case child elements. The nete:case children provide the unique values that are used 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 CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® SPS to a destination server. Filters can be written using the Filter API.

More information:

Filter API Overview (see page 202)

Filter Attribute

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 CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® 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.

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 CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® 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.

- $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

Response Handling

The CA SiteMinder® SPS uses SiteMinder responses to determine a destination for a request. Since transactions that are routed through the CA SiteMinder® SPS include an interaction between the CA SiteMinder® SPS web agent and SiteMinder, any SiteMinder responses gathered during the authentication and authorization process may be used by the CA SiteMinder® 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 CA SiteMinder® 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.

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 CA SiteMinder® SPS processes requests, and it encounters a nete:xprcond element in the proxy rules configuration file, the following occurs:

1. The CA SiteMinder® 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 CA SiteMinder® 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

Characters	Results
[: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
[: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)

Characters	Results
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)
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 CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® 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 URl 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 CA SiteMinder® SPS uses SiteMinder responses to determine a destination for a request. Since transactions that are routed through the CA SiteMinder® SPS include an interaction between the CA SiteMinder® SPS web agent and SiteMinder, any SiteMinder responses gathered during the authentication and authorization process may be used by the CA SiteMinder® 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 CA SiteMinder® 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 Manually

To modify manually, edit the proxyrules.xml file using a text editor. Verify that the proxy rules configuration file must be well-formed and valid, and 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 proxyrules.xml file are picked up automatically. When a request is received, CA SiteMinder® SPS 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 CA SiteMinder® SPS.

Manage Proxy Rules Using Administrative UI

To modify using Administrative UI, perform one of the following steps:

- Import the proxyrules.xml file and edit the proxy rules, if required, by performing the following steps:
  1. Navigate to Proxy Rules, Rules.
  2. Click Import.
  3. Click Browse to select the location of the file, and click Upload.
  4. You can view the imported proxy rules in the tree mode or the text mode.
  5. (Optional) Edit the proxy rules and click Save.
  6. (Optional) Edit the existing proxy rules in a tree mode or a text mode.
- Edit the existing proxy rules in a tree mode or a text mode by performing the following steps:
  1. Navigate to Proxy Rules, Rules.
  2. Click Text Mode.
  3. Edit the required proxy rules for your environment.
  4. Click Save.

Add Filters

Custom filters are filters defined by customer's needs. CA SiteMinder® 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.

To add filters in proxyrules.xml or text mode in the Admin UI, type the filter as required in the proxy rules.

Example:

<nete:forward filter="filter1"http://FQDN$0</nete:forward>

To add filters in the tree mode in the Admin UI, perform the following steps:

1. Click Proxy Rules, Filters.
2. Click Add.
3. Type the filter name and Java class name in Name and Class.
4. (Optional) To add filter parameters, click Add in the Parameters table and enter the details.
5. Click OK.

To apply a filter in the tree mode in the Admin UI, enter the name of the filter you want to apply when you create an action proxy rule.

Validate the Proxy Rules

You can validate the new proxy rules against the proxy rules DTD. The validation ensures that the new proxy rules are defined in the correct XML file configurations as per the DTD.

To validate the new proxy rules, click Validate in the Proxy rules page in any mode after you save the proxy rules.

Test the Proxy Rules

You can test the proxy rules to verify their functionality.

Follow these steps:

1. Click Proxy Rules, Proxy Rules Test.
2. Enter a sample test request as per the proxy rules you have defined.
3. Click Test Proxy Rules.
4. Verify that the result displayed in Result conforms with the proxy rules that are defined.

Export the Proxy Rules

You can export the new proxy rules through the Admin UI for future reference.

Follow these steps:

1. Click Export from any mode.
2. Click Save in the File Download dialog and specify the location to save the proxyrules.xml file.
3. Click OK.

Sample Proxy Rules Configuration Files

The CA SiteMinder® 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 CA SiteMinder® SPS requests based on the hostname specified in the request, CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® 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.

In this example, assume that the CA SiteMinder® 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 HTTP_HEADER has the following value: HTTP_HEADER="value1"	http://server1.company.com/index.html
http://www.company.com/index.html HTTP_HEADER has the following value: HTTP_HEADER="value2"	http://server2.company.com/index.html
http://www.company.com/index.html 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. CA SiteMinder® 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 CA SiteMinder® 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 User access resource via a Web browser.	http://home.company.com/index.html
http://www.company.com/index.wml 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 CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® SPS requests based on the hostname, certain headers, and device types. This file demonstrates how the CA SiteMinder® 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 CA SiteMinder® 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 with a header value of GOLD_USER="yes"	http://fast.company.com/bondtrading/index.html
http://bondtrading.company.com/index.html with a header value of GOLD_USER="no"	http://server2.company.com/bondtrading/index.html

Requested URL	Forwarded URL
http://www.company.com/index.wml with a USER_AGENT header value that contains a wireless device name	http://home.company.com/ wireless/index.wml
http://www.company.com/index.html 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 CA SiteMinder® 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 CA SiteMinder® 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
http://server.company.com/GOTO=server2.company.com/index.ht ml	http://server2.company.com/index.html
http://server.company.com/REDIR=server2.company.com/index.ht ml	http://server2.company.com/index.html 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 CA SiteMinder® SPS requests based on the existence of a cookie.

In this example, if a request contains a cookie header "mycookie", CA SiteMinder® SPS routes the request to www.company.com.

Proxy Rules Example—Routing Requests by Cookie Value

The example file proxyrules_example9.xml file routes CA SiteMinder® 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, CA SiteMinder® SPS routes the request to www.abcd.com. If a request contains a cookie header "mycookie" and the base64 encoding mechanism, CA SiteMinder® SPS routes the request to www.xyz.com.

Configure CA SiteMinder® SPS to Use FIPS

CA SiteMinder® SPS supports the requirements for cryptographic modules specified in the FIPS 140-2 standard. When you install CA SiteMinder® SPS, a dialog appears that prompts you to select the level of FIPS support your operating configuration requires. If you are upgrading an existing CA SiteMinder® SPS installation, CA SiteMinder® 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, CA SiteMinder® SPS server, and the Apache server pick up the changes.

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 CA SiteMinder® SPS.
- MIGRATE — Specifies that CA SiteMinder® SPS operates both with FIPS-compliant algorithms and algorithms used in earlier version of CA SiteMinder® SPS simultaneously while the data is migrated.
- ONLY — Specifies that only FIPS-compliant algorithms are used and accepted by CA SiteMinder® 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 CA SiteMinder® SPS in any 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 CA SiteMinder® SPS from COMPAT mode to MIGRATE mode.

To set CA SiteMinder® SPS to FIPS MIGRATE mode

1. Stop the CA SiteMinder® 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 CA SiteMinder® SPS services.

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

Configuration Process for FIPS ONLY Mode

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

- Verify that CA SiteMinder® SPS is running in full SSL mode.
- Verify that the server key used to configure CA SiteMinder® 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 ONLY Mode

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

Follow these steps:

1. Stop the CA SiteMinder® 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 CA SiteMinder® SPS is running in full SSL mode. If SSL is already enabled on Apache inside CA SiteMinder® 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 the CA SiteMinder® SPS services.

Using CA SiteMinder® SPS with Federation Security Services

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 CA SiteMinder® 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.

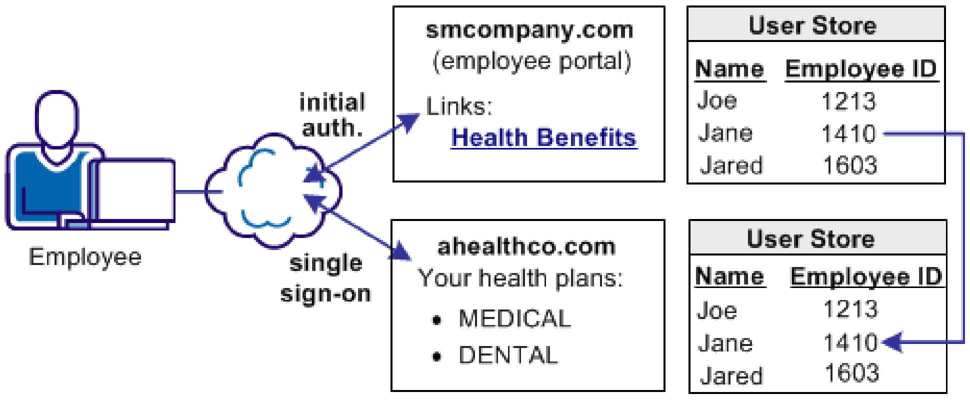
For more use cases, see the *CA SiteMinder Federation Security Services Guide*.

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.

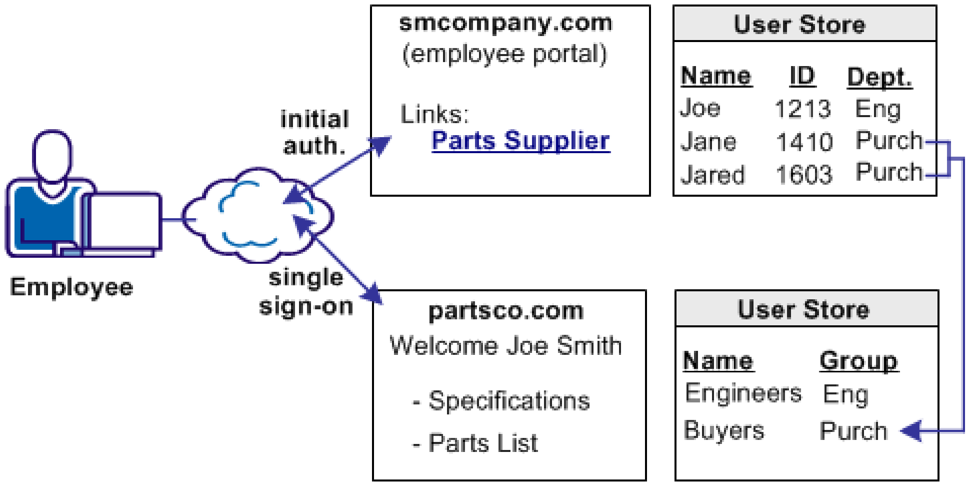
More information:

Solution 1: SSO Based on Account Linking (see page 113)

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 portion of the 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 Parts List portion 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, but 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:

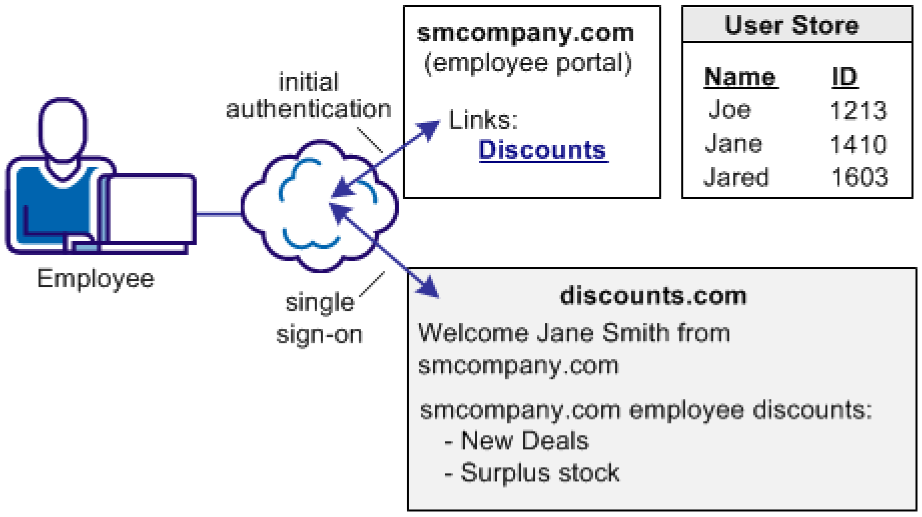
Solution 2: SSO Using User Attribute Profiles (see page 116)

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 smcompany.com and clicks a link to access 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 smcompany.com. The company allows all employees of smcompany.com to access discounts .com as long 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.

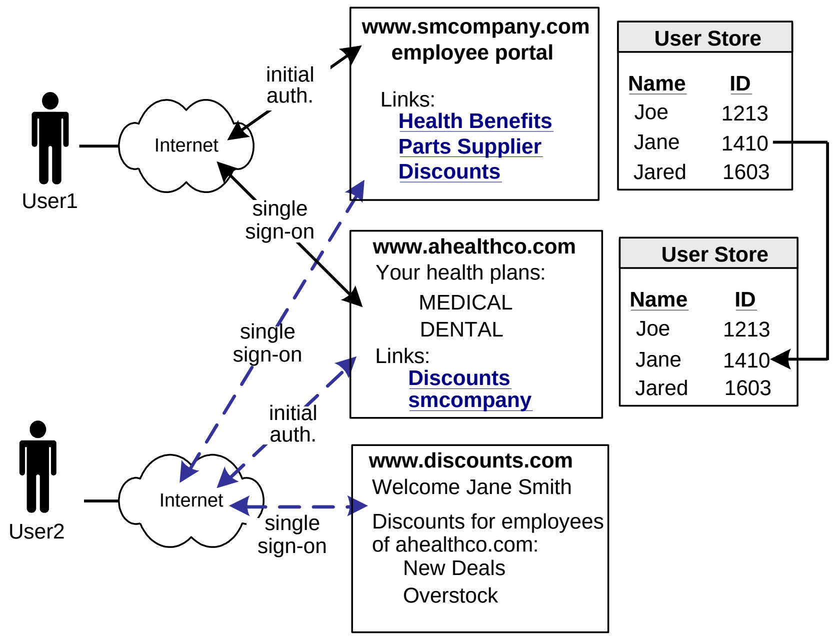
Additional attributes, such as the user name are passed from smcompany.com to discounts.com to personalize the interface for the individual user.

More information:

Solution 3: SSO with No Local User Account (see page 118)

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 120)

SPS Roles in a SiteMinder Federated Environment

CA SiteMinder® 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.

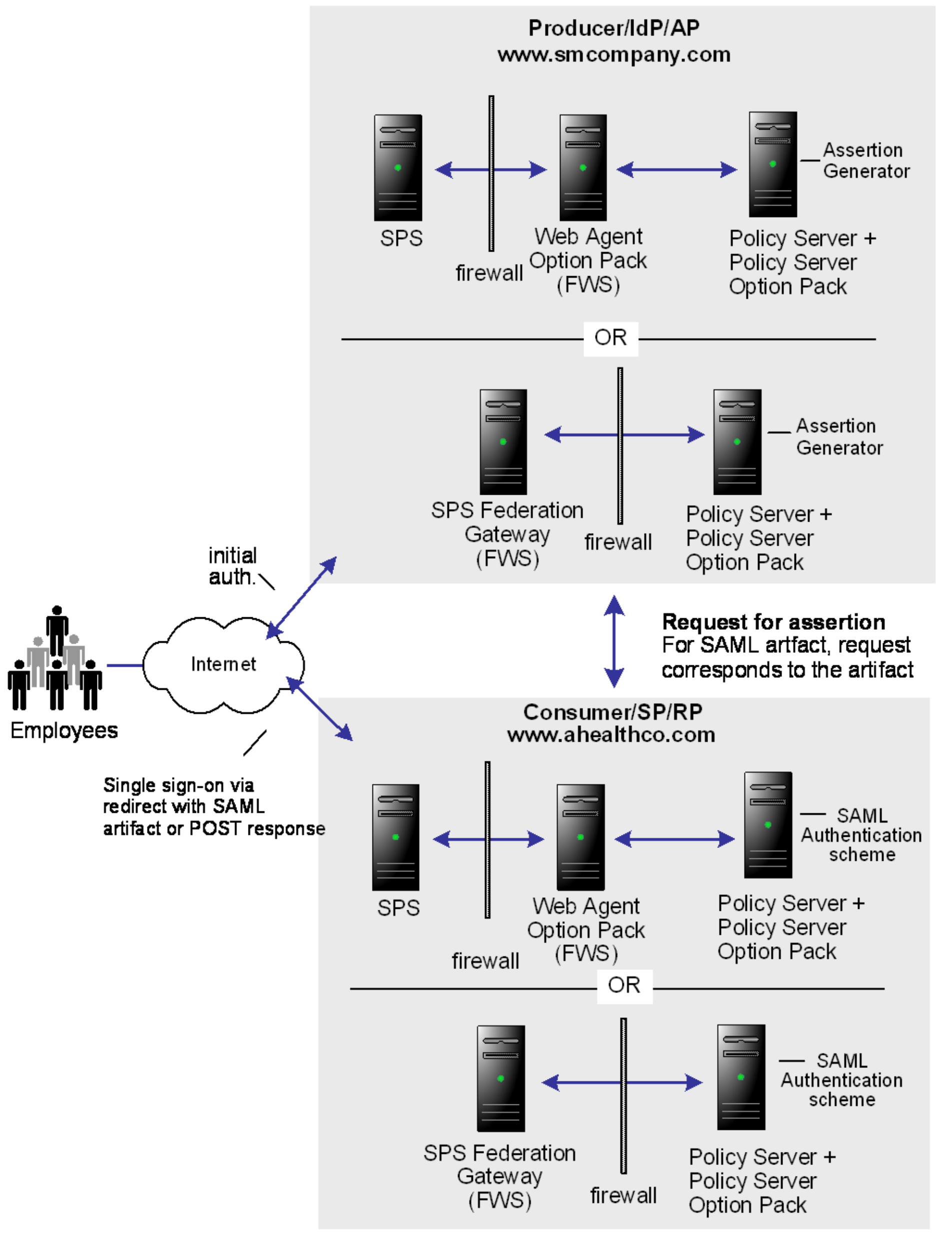
Solutions for SPS Use Cases

The following sections show CA SiteMinder® 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 108): Single Sign-on Based on Account Linking.

The following figure shows the solution based on account linking.


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.
Employees
Internet
Request for assertion
For SAML artifact, request corresponds to the artifact
Single sign-on via redirect with SAML artifact or POST response
Consumer/SP/RP
www.ahealthco.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

CA SiteMinder® is deployed at both sites and the installations are the same for both smcompany.com and ahealthco.com. CA SiteMinder® SPS with the Web Agent Option Pack or CA SiteMinder® 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. CA SiteMinder® 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. CA SiteMinder® 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 CA SiteMinder® 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

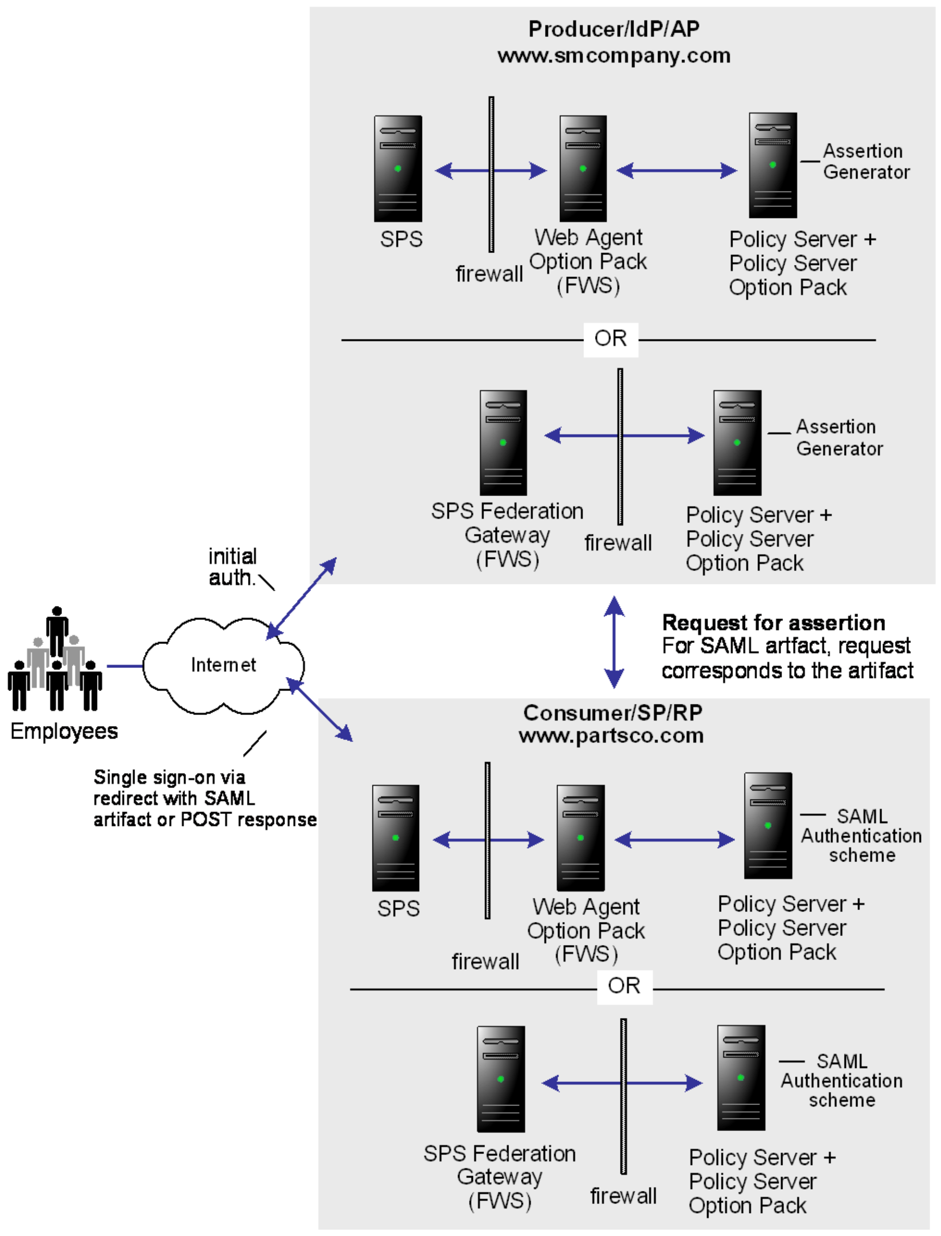
Solution 2 shows how SiteMinder Federation Security Services can be deployed at smcompany.com and partsco.com to solve Use Case 2 (see page 110): Single Sign-on Based on User Attribute Profiles.

CA SiteMinder® 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)
Policy Server + Policy Server Option Pack
Assertion Generator
OR
SPS Federation Gateway (FWS)
firewall
Policy Server + Policy Server Option Pack
Assertion Generator
initial auth.
Employees
Internet
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)
Policy Server + Policy Server Option Pack
SAML Authentication scheme
OR
SPS Federation Gateway (FWS)
firewall
Policy Server + Policy Server Option Pack
SAML Authentication scheme

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

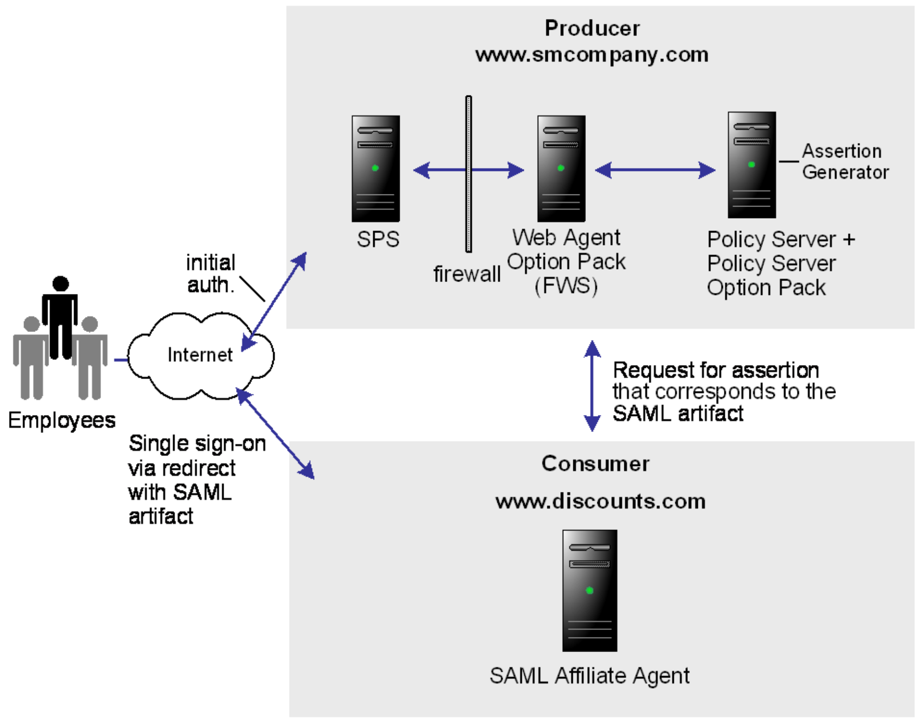
Solution 3 shows how SiteMinder Federation Security Services can be deployed at smcompany.com and discounts.com to solve Use Case 3 (see page 111): Single Sign-on with No Local User Account.

CA SiteMinder® is deployed at smcompany.com by installing CA SiteMinder® 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: CA SiteMinder® 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.


Producer
www.smcompany.com
SPS
firewall
Web Agent Option Pack (FWS)
Assertion Generator
Policy Server + Policy Server Option Pack
initial auth.
Internet
Employees
Single sign-on via redirect with SAML artifact
Request for assertion that corresponds to the SAML artifact
Consumer
www.discounts.com
SAML Affiliate Agent

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. CA SiteMinder® 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 CA SiteMinder® SPS at www.smcompany.com.
3. CA SiteMinder® 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. CA SiteMinder® 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.

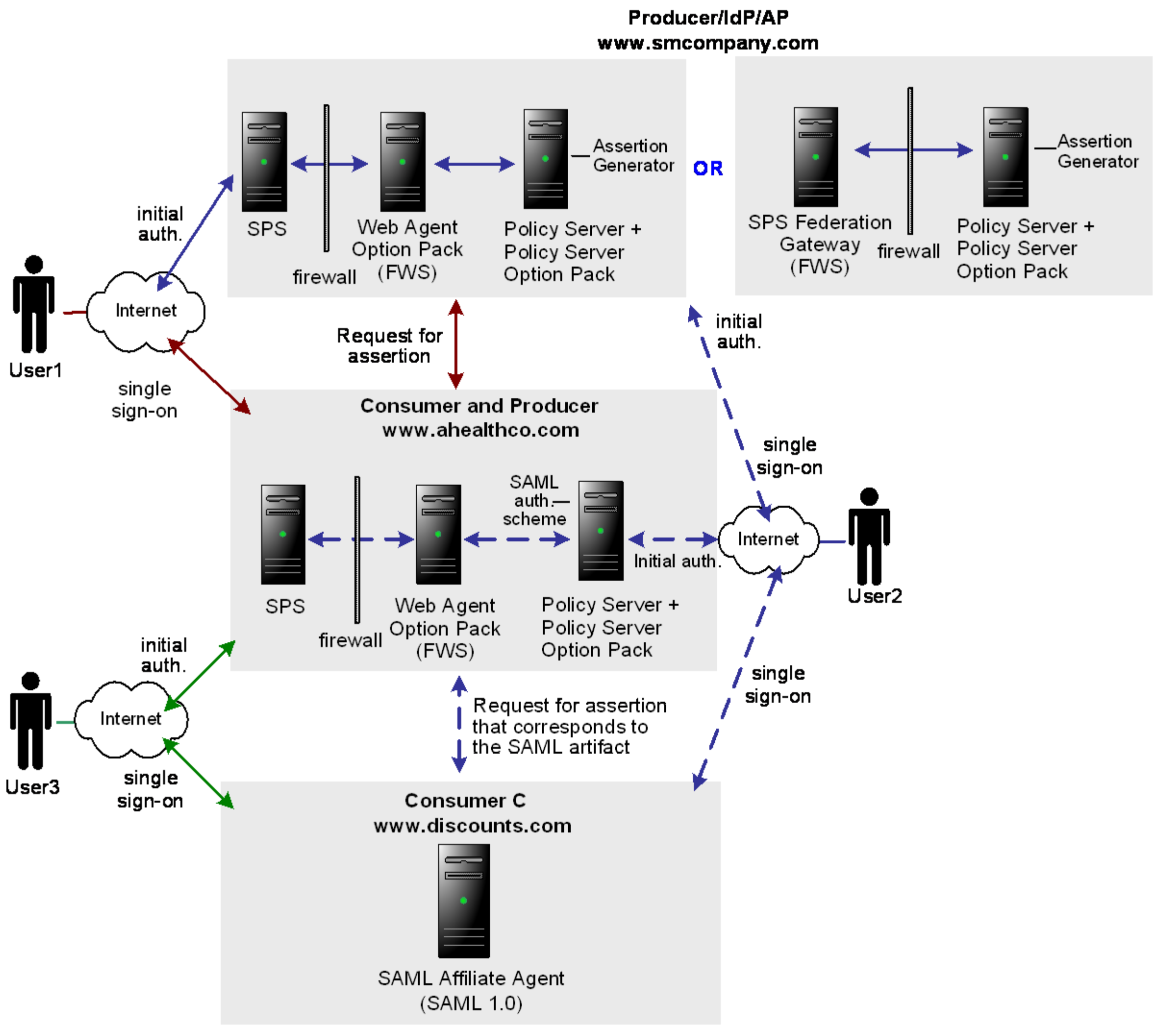
The administrator at smcompany.com uses the Policy Server User Interface to configure an affiliate for discounts.com. The affiliate is configured with attribute information to be passed to discounts.com. The assertion generator will include those attributes as part of the user profile in a SAML assertion 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 112): 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, CA SiteMinder® SPS with the Web Agent Option Pack can be installed across two machines or CA SiteMinder® 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, CA SiteMinder® 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.

Using CA SiteMinder® SPS in 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. CA SiteMinder® SPS verifies if cookieless federation is enabled for the virtual host requesting the redirect.
2. CA SiteMinder® SPS verifies if the session scheme is a rewritable scheme, such as the simple_url scheme.
3. If the scheme is rewritable, CA SiteMinder® SPS determines whether a session key has been created for the session and if this key is available to use.
4. CA SiteMinder® 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. CA SiteMinder® 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, CA SiteMinder® SPS replacing the Web Agent processes redirects using SAML authentication at the backend server.

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

1. CA SiteMinder® SPS receives a request from cookieless device, such as a mobile phone.
2. CA SiteMinder® SPS verifies if the cookieless federation is enabled for the virtual host requesting the redirect.
3. CA SiteMinder® 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. CA SiteMinder® SPS retrieves the session scheme being used.
5. CA SiteMinder® 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, CA SiteMinder® SPS rewrites the location header with the session key.

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

Enable Cookieless Federation at the Consuming Side

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

To enable cookieless federation for CA SiteMinder® 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.

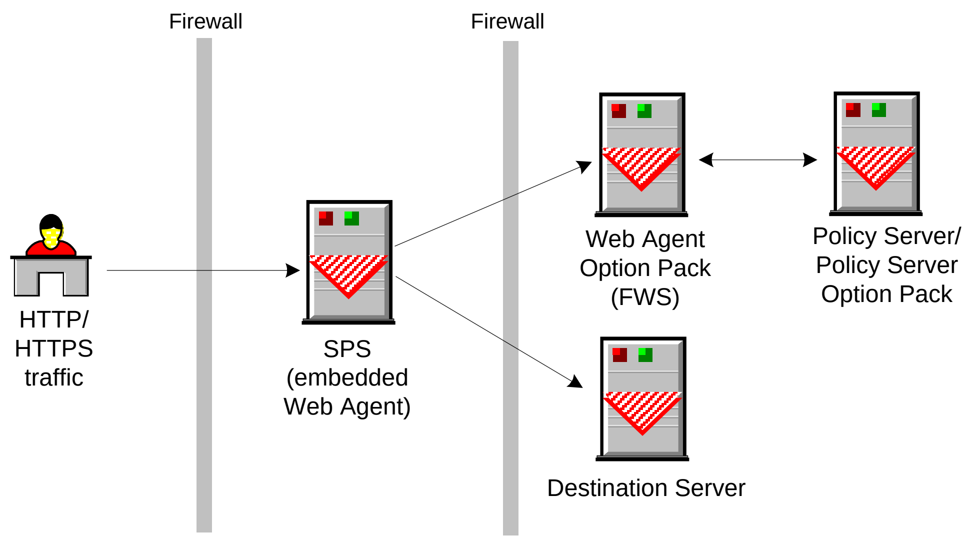
   CA SiteMinder® SPS is configured for cookieless federation at the consuming partner.

Using SPS as a Web Agent Replacement

To provide federated single sign-on, CA SiteMinder® SPS may be used as a substitute for the SiteMinder Web Agent. CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® SPS replaces a SiteMinder Web Agent.



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

Prerequisites for Using the SPS as a Web Agent Replacement

Before you configure CA SiteMinder® 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.
- In the SiteMinder Policy Server User Interface, define the host information (server and port number) for the CA SiteMinder® SPS system that generates assertions. The CA SiteMinder® 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 CA SiteMinder® SPS to operate in a federated environment is similar to the standard CA SiteMinder® SPS configuration process.

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

1. Install CA SiteMinder® 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 CA SiteMinder® SPS, you can modify the Apache web server file (httpd.conf), for example,

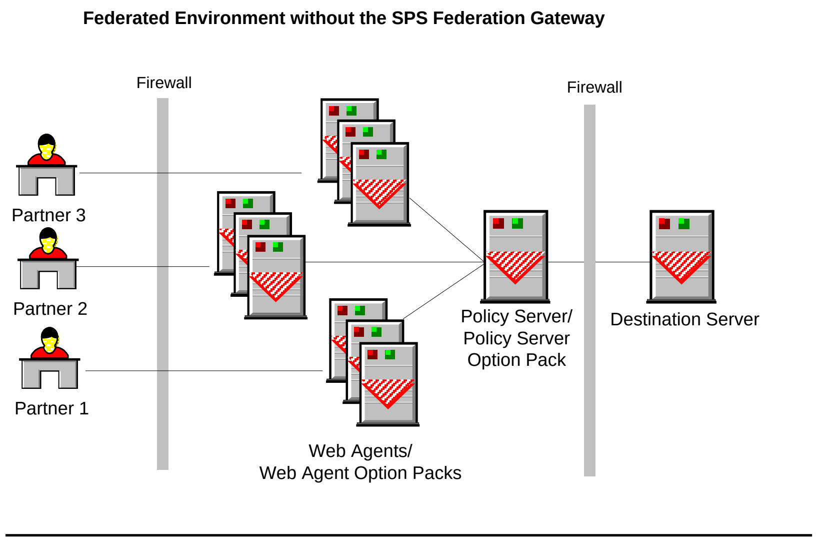
Using SPS as a Federation Gateway

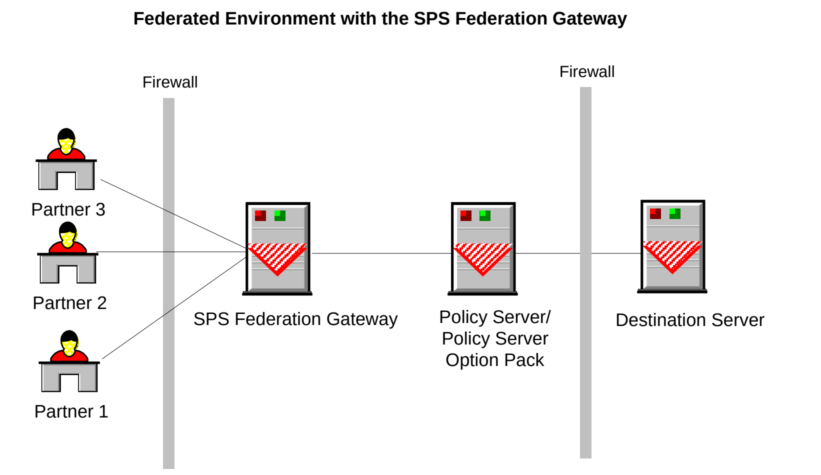
CA SiteMinder® SPS federation gateway simplifies the configuration involved in a federated environment. Typically, you have a federated environment where partners are communicating through many web servers. Each web server requires that you install and configure the Web Agent and the Web Agent Option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

If you enable CA SiteMinder® SPS as a federation gateway, the number of components that you have to install and set-up is reduced. The CA SiteMinder® SPS federation gateway has the standard embedded components of CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® SPS federation gateway.





Prerequisites for Using the Federation Gateway

Before you set up CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® SPS system that generate assertions. The CA SiteMinder® 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 CA SiteMinder® SPS federation gateway can sit at the producer site and consumer site.

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

1. Install CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® 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.

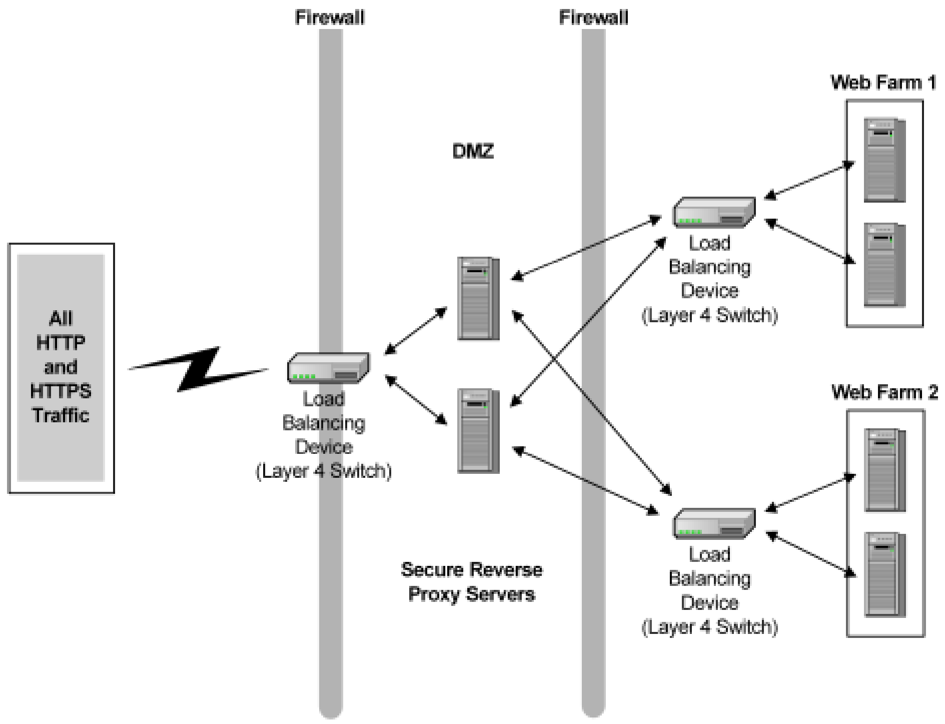
Chapter 4: Configuring CA SiteMinder® SPS

This section contains the following topics:

CA SiteMinder® SPS in an Enterprise

CA SiteMinder® 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 CA SiteMinder® SPS is intended to be an addition to enterprise architecture, rather than a replacement for other proxy technologies. As such, the CA SiteMinder® 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 CA SiteMinder® SPS can be inserted into a network to work in conjunction with load balancing devices.


Firewall
Firewall
Web Farm 1
DMZ
All HTTP and HTTPS Traffic
Load Balancing Device (Layer 4 Switch)
Load Balancing Device (Layer 4 Switch)
Web Farm 2
Secure Reverse Proxy Servers
Load Balancing Device (Layer 4 Switch)

Note: In addition to load balancing devices, caching devices can be placed on either side of the CA SiteMinder® SPS cluster.

Sticky-Bit Load Balancing

When using the cookieless session schemes supported by the CA SiteMinder® SPS, session information for users who access resources through CA SiteMinder® SPS is maintained in an in-memory session store. Because the session information is maintained at the CA SiteMinder® SPS where a user is first authenticated, the same CA SiteMinder® SPS should be used for all the user requests in a single session. When implemented in clusters, the CA SiteMinder® SPS must be used in conjunction with sticky-bit load balancers to provide a consistent connection to the same CA SiteMinder® SPS, enabling single sign-on when using session schemes other than the traditional SiteMinder cookie session scheme.

To deploy the CA SiteMinder® SPS using cookieless session schemes the following must be considered:

- In most deployments, the CA SiteMinder® 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 CA SiteMinder® SPS in a cluster, that CA SiteMinder® SPS services all of the user's requests. This capability is required because the CA SiteMinder® 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 CA SiteMinder® SPS in the cluster of servers.

- When configuring the settings for the CA SiteMinder® SPS, the default virtual host defined in the server.conf file of the CA SiteMinder® 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 CA SiteMinder® SPS.

- The load balancing device must point to the cluster of CA SiteMinder® 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 CA SiteMinder® SPS.

- If using SSL, a certificate must be issued to the load balancer, not the CA SiteMinder® SPS.

- The system or systems on which you install the CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® SPS to proxy for destination servers trusted by your enterprise.

Configuring Virtual Hosts

You can configure the CA SiteMinder® SPS with multiple hosts and act as a virtual host for one or more hostnames.

Follow these steps:

1. Edit the <VirtualHost> parameters of the server.conf file to configure the CA SiteMinder® SPS to act as a virtual host for one or more hostnames.
2. Edit the configuration file for the embedded Apache Web server.

Edit the Apache Configuration File To Handle Multiple Virtual Hosts

When you are running multiple virtual hosts in the same operating environment with the CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® SPS will use SiteMinder cookies session scheme to maintain sessions for all users who access server2.

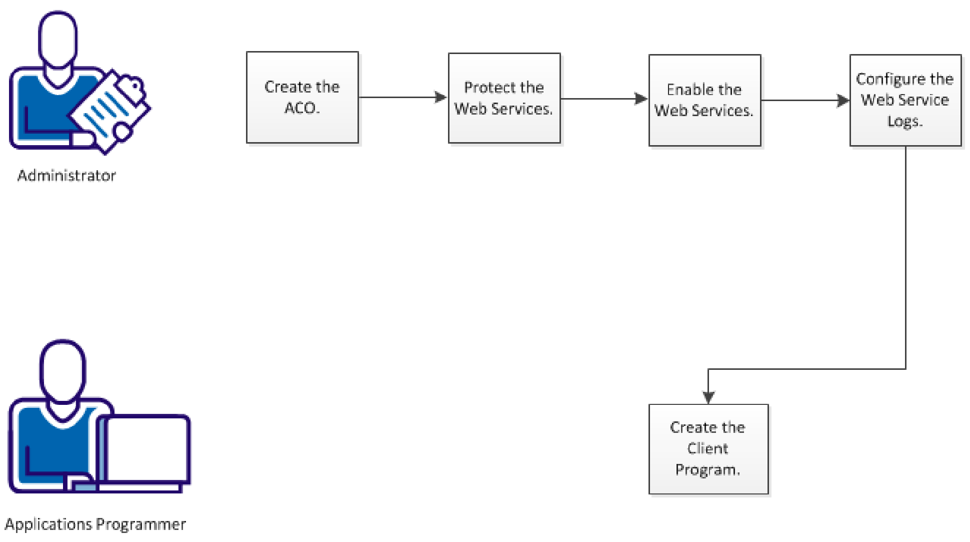
```
# Additional Virtual Host
<VirtualHost name="host2">
       requestblocksize="4"
        responseblocksize="4"
        hostnames="server2, server2.company.com"
        #addresses="192.168.1.15"
        # default session scheme
        defaultsessionscheme="default"

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

        #<WebAgent>
        #</WebAgent>
</VirtualHost>
```

Configuring the Authentication and Authorization Web Services

How to Work with the Authentication and Authorization Web Services

CA SiteMinder® currently offers an authentication web service and an authorization web service. The process of working with the CA SiteMinder® authentication and authorization web services includes the procedures shown in the following diagram:

How to Work with the Authentication and Authorization Web Services


Administrator
Create the ACO.
Protect the Web Services.
Enable the Web Services.
Configure the Web Service Logs.
Create the Client Program.
Applications Programmer

To work with the authentication and authorization web services, perform the following steps:

1. Create the ACO (see page 141).
2. Protect the Web Services (see page 142).
3. Enable the Web Services (see page 142).
4. Configure the Web Service Logs (see page 143).
5. Create the Client Program (see page 143).

Overview of the Authentication and Authorization Web Services

The CA SiteMinder® authorization and authentication web services are part of the CA SiteMinder® SPS installation. You can enable or disable them individually.

The web services configuration process presupposes configuration of the following CA SiteMinder® objects:

- One or more agents to protect target applications against which callers authenticate
- Realms, user directories, policies, and responses that are required for authentication and authorization

You can use the authentication and authorization web services to support an application that is not otherwise protected. A free-standing application on a mobile phone, for example, can authenticate a user when the appropriate CA SiteMinder® objects are available.

These web services support the SOAP 1.2 protocol and the HTTP-based RESTful architecture. The authentication and authorization web services provide the following functionality:

- login — Authenticates and returns a session token when the authentication is successful.

  Note: If the Enable User Tracking option is enabled, the response contains an identity token additionally.
- blogin — Authenticates and verifies whether the login is successful; does not return a session token.
- logout — Logs out the user or group of users.
- authorize — Returns an authorization status message and a refreshed session token.

The response to a request of an operation is dependent on the corresponding SiteMinder generated headers. If a resource is protected with the Anonymous authentication scheme, the response does not contain a session token but contains an identity token. The identity token can be used in the subsequent authorization request instead of a session token.

An authentication request includes the following parameters:

- appId
- resource string
- action
- user credentials

The appId references a user-defined logical name for the location of a hierarchy of resources, not a CA SiteMinder® Application object. Internally, the appId maps to an agent. CA SiteMinder® uses the agent name to determine the realm. The realm, the resource string, and user credentials are enough to authenticate the user.

An authorization request is simpler than an authentication request. The authorization request includes an appId, resource path, action, and session token, obtained from the login response. The web service validates the token and determines whether to grant access to the specified resource.

Configure the Web Services

By default, the web services feature is installed when you install or upgrade to CA SiteMinder® SPS 12.51.

To configure the web services, perform the following steps:

1. Create an ACO for the web services through WAMUI.
2. Protect the web services.
3. Enable the web services through Administrative UI.
4. (Optional) Configure the web services logs.

Create an ACO for the Web Services

You can manage the web services through an ACO. The ACO is also used for resource access protection and must be defined in AgentName. To use the web services, you must enable the enableauth parameter or the enableaz parameter or both.

Follow these steps:

1. Create an ACO that is based on the AuthAzServiceDefaultSettings template in WAMUI.
2. Configure the following parameters to use the web services as a service:

   AgentName

   Defines the names of the web agent that protects a resource, and the defaultagentname *or* the agent name of the ACO that protects the web services. You must append these values in AgentName.

   To define multiple web agents that protect an application, enter a multi-value pair in the following format:

   ```
   agent_name1,appID1
   agent_name2,appID2
   agent_namen,appIDn
   ```

   agent_name

   Defines the name of the web agent that protects a resource.

   appID

   Defines the reference name of the web agent that was specified in agent_name or of the application that is protected by the web agent. CA SiteMinder® uses this value in the web services requests, thus protecting the agent name from the users.

   Append the specified AgentName with the defaultagentname *or* the agent name of the ACO that protects the web services.

   To use the agent name of the ACO that protects the web services, define the agent name in the the following format:

   ```
   agent_name,hostname
   ```

   To use the defaultagentname of the ACO that protects the web services, define the agent name in the following format:

   ```
   agent_name
   ```

   enableauth

   Specifies the status of the authentication web service. If you want to use the authentication web service, set the value to yes.

   enableaz

   Specifies the status of the authorization web service. If you want to use the authorization web service, set the value to yes.

RequireAgentEnforcement

Specifies whether the web services must be protected by a CA SiteMinder® agent. In a production environment, we highly recommended that you set this value to yes and protect the web services by a CA SiteMinder® Agent. If you set the value to yes and the web services are not protected, the requests to web services fail.

Note: The value of RequireAgentEnforcement can be set to no in a test environment or if the web services are protected by any other mechanism other than CA SiteMinder®.

3. Save the changes.

Protect the Web Services

We recommend that you protect the web services in a production environment. Protecting the web agent of the web services lets CA SiteMinder® authenticate and authorize the web services client before a user request is processed. When you protect the web services in your production environment, CA SiteMinder® SPS includes the SMSESSION cookie into the user request. If the RequestSmSessionCookie ACO parameter is enabled, CA SiteMinder® ensures that the web services verify the user request for the SMSESSION cookie before processing the user request.

To protect the web services, we recommend that you configure CA SiteMinder® SPS to protect the web services root URL using the X.509 Client Certificate authentication scheme.

Enable the Web Services

Use the ACO that you created in the previous procedure to enable the web services through Administrative UI.

Note: If the values of enableauth and enableaz are set to no, the web services do not function even though you enable the support through CA SiteMinder® SPS Admin UI.

Follow these steps:

1. Navigate to Proxy Configuration, Auth and Az Web Services.
2. Type the unique host name of the web services virtual host in Host Name.
3. Type the name of the ACO that is created for the web services in Agent Configuration Object.
4. Click Save.

   The web services are enabled.

Configure the Web Services Logs

When you enable the web services, CA SiteMinder® SPS saves the logs of the web services in the server.log file. You can change the log location from server.log to authazws.log.

If you want to change the log location, perform the following steps:

1. Navigate to *sps_home*/proxy-engine/conf/webservicesagent.
2. Take a back-up of the file authaz-log4j.xml.
3. Open the original authaz-log4j.xml file, and perform the following steps:
   a. Uncomment the following AuthAZ_ROLLING appender tag:

```
<appender name="AuthAZ_ROLLING"
class="org.apache.log4j.DailyRollingFileAppender">
<param name="File" value="logs/authazws.log"/>
        <layout class="org.apache.log4j.PatternLayout">
        <param name="ConversionPattern" value="%d %-5p [%c] - %m%n"/>
</layout>
</appender>
```

   b. Uncomment all the occurrences of the following appender-ref for the AuthAZ_ROLLING tag:

```
<appender-ref ref="AuthAZ_ROLLING"/>
```

4. Save the changes and restart CA SiteMinder® SPS.

   The log location is changed to the authazws.log file, which is located in *sps_home*/proxy-engine/logs/.

Create the Client Program

The function of the client program is to issue authentication and authorization requests to the web service on behalf of another application. The client program requires code for a client stub. The stub manages, sends, and receives messages to communicate with the web service. The web service support a WSDL file (for the SOAP protocol) and a WADL file (for the REST architecture). You can access the WSDL or WADL file using a web browser, and then save it as an XML file.

Follow these steps:

1. Write the business logic for your application, which gathers the required credentials.
2. Create the client stub. Optionally, you can use the WSDL or WADL file with a third-party tool to generate the client stub.
   - To load the WSDL, use the following URL:

     `http://`*`hostname:port`*`/authazws/auth?wsdl`
   - To load the WADL, use the following URL:

     `http://`*`hostname:port`*`/authazws/AuthRestService/application.wadl`

   Note: To retrieve the metadata from these locations, be sure to set the DefaultAgentName parameter in the ACO to one of your agents.
3. Import the client stub and instantiate the stub object to invoke the web service.

The sections that follow list simplified sample SOAP and REST messages for reference.

Authentication SOAP Interface

These simplified samples show authentication works using the SOAP protocol. Most authentication schemes can be supported by an IdentityContext consisting of just three fields, username, password, binaryCredentials. Other schemes, requiring more fields are supported by additional operations whose inputs are tailored to the credential type.

The following example is an authentication web service normal user login request:

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
            xmlns:aut="http://ca.com/2010/04/15/authentication.xsd">
  <s:Header/>
  <s:Body>
    <aut:login>
      <identityContext>
        <binaryCreds>
        </binaryCreds>
        <password>user1</password>
        <userName>user1</userName>
      </identityContext>
      <appId>app1</appId >
      <action>GET</action>
      <resource>/*</resource >
    </aut:login>
  </s:Body>
</s:Envelope>
```

The blogin operation (Boolean Login) is similar to the login operation. However, blogin does not return a SMSESSION value in the response, as shown in the following example:

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
            xmlns:aut="http://ca.com/2010/04/15/authentication.xsd">
  <s:Header/>
  <s:Body>
    <aut:blogin>
      <identityContext>
        <binaryCreds>
        </binaryCreds>
        <password>user1</password>
        <userName>user1</userName>
      </identityContext>
      <appId>app1</appId >
      <action>GET</action>
      <resource>/*</resource >
    </aut:blogin>
  </s:Body>
</s:Envelope>
```

The following example represents a successful login response:

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope">
  <s:Header/>
  <s:Body>
    <aut:loginResponse
xmlns:aut="http://ca.com/2010/04/15/authentication.xsd">
      <return>
        <message>Authentication successful.</message>
        <resultCode>LOGIN_SUCCESS</resultCode>
        <sessionToken>session</sessionToken>
    <responses>
      <response/>
      <response/>
    </responses>
      </return>
    </aut:loginResponse>
  </s:Body>
</s:Envelope>
```

The following example represents a failed login attempt:

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope">
  <s:Header/>
  <s:Body>
    <ns2:loginResponse xmlns:ns2="http://webservice.sm.services.soa.ca.com/">
      <return>
        <message>Authentication failured</message>
        <resultCode>LOGIN_FAILED</resultCode>
        <smSessionCookieValue/>
      </return>
    </ns2:loginResponse>
  </s:Body>
</s:Envelope>
```

The following example represents an authentication web service user logout request:

Note: Even though a user has successfully logged out, the agent can still use the SessionToken to authorize, because it is considered to be a valid user credential.

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
            xmlns:aut="http://ca.com/2010/04/15/authentication.xsd">
  <s:Header/>
  <s:Body>
    <aut:logout>
      <smSessionCookieValue>session</smSessionCookieValue>
    </aut:logout>
  </s:Body>
</s:Envelope>
```

The following example represents a successful authentication web service logout response:

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope">
  <s:Header/>
  <s:Body>
    <ns2:logoutResponse
xmlns:ns2="http://ca.com/2010/04/15/authentication.xsd">
      <return>
        <message>Logout successful.</message>
        <resultCode>SUCCESS</resultCode>
      </return>
    </ns2:logoutResponse>
  </s:Body>
</s:Envelope>
```

Authentication REST Interface

REST means REpresentational State Transfer. In REST, service requests transform the state of objects accessible by URIs. HTTP drives state change using actions such as create, read, update, and delete.

The URI mapping for authentication and authorization consists of the appId and resourcePath. The resource state is the collections of authenticated or authorized users associated with the Resource. The service names for authentication are login, blogin, and logout.

A URI in this format, http://hostname:port/authazws/AuthRestService/login/*appID*/*Resource*, posts the following request:

```
<loginRequest>
      <binaryCreds></binaryCreds>
       <password>user1</password>
       <userName>user1</userName>
       <action>GET</action>
</loginRequest>
```

The login responses:

HTTP return code 200

```
<loginResponse>
<message>Authentication successful</message>
<resultCode>LOGIN_SUCCESS</resultCode>
<sessionToken>session</sessionToken>
<authenticationResponses>
    <response>
            <name>SM_SESSIONDRIFT</name>
            <value>0</value>
    </response>
</authenticationResponses>
</loginResponse>
```

HTTP return code 400

```
<loginResponse>
<message>Bad Request</message>
<resultCode>LOGIN_ERROR</resultCode>
</loginResponse>
```

HTTP return code 200

```
<loginResponse>
<message>Authentication Failed</message>
<resultCode>LOGIN_FAILED</resultCode>
<authenticationResponses>
    <response><name>SM_AUTHREASON</name>
    <value>0</value>
    </response>
</authenticationResponses>
</loginResponse>
```

HTTP return code 500

```
<loginResponse>
<message>System</message>
<resultCode>Server Error</resultCode>
</loginResponse>
```

The bLogin operation (Boolean Login) is similar to login. A URI in this form, http://host:port#/blogin/appId/resourcePath posts as shown in the login request. It returns yes or no in the response message.

A URI in this format, http://host:port#/authazws/AuthRestService/logout/, posts the following the logout request:

```
<logoutRequest>
<sessionToken>session</sessionToken>
</logoutRequest>
```

The authentication web service logout responses:

```
<logoutResponse>
<message>Logout Successful</message>
<resultCode>LOGOUT_SUCCESS</resultCode>
<smSessionCookieValue>yyy</smessionCookieValue>
</logoutResponse>

<logoutResponse>
<message>Logout Failed</message>
<resultCode>LOGOUT_FAILURE</resultCode>
<smSessionCookieValue>yyy</smessionCookieValue>
</logoutResponse>
```

Authorization SOAP Service

The following XML approximates an authorization request to the web service:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:aut="http://ca.com/2010/04/15/authorization.xsd">
   <soapenv:Header/>
   <soapenv:Body>
      <aut:authorize>
    <sessionToken>session</sessionToken>
          <appId></appId>
          <action>GET,POST</action>
     <resource>/domainAdmin/a.jsp</resource>
      </aut:authorize>
   </soapenv:Body>
</soapenv:Envelope>
```

The following example represents an authorization web service AUTHORIZED response:

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
   <env:Header/>
   <env:Body>
      <ns2:authorizeResponse
xmlns:ns2="http://ca.com/2010/04/15/authorization.xsd">
         <return>
            <message>Authorization Successful</message>
            <resultCode>AUTHORIZED</resultCode>
    <sessionToken>aklaks</sessionToken>
    <authorizationResponses>
    <response/>
    </authorizationResponses>
         </return>
      </ns2:authorizeResponse>
   </env:Body>
</env:Envelope>
```

The following example represents an authorization web service UN AUTORIZED response:

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
   <env:Header/>
   <env:Body>
      <ns2:authorizeResponse
xmlns:ns2="http://ca.com/2010/04/15/authorization.xsd">
         <return>
            <message> Authorization Failed</message>
            <resultCode>NOTAUTHORIZED</resultCode>
         </return>
      </ns2:authorizeResponse>
   </env:Body>
</env:Envelope>
```

Note: For an authorization web service request with a valid session token, the NOTAUTHORIZED authorization response has the following constraints:

1. You can configure the response with only the following attributes in the WAMUI:
   - SM_ONREJECTTEXT
   - SMREDIRECTURL or SM_REDIRECTURL
   - SMERROR
2. The response does not contain a session token.

Authorization REST Interface

The REST interface for authorization is http://hostname:port/authazws/AuthRestService/authz/*appID*/*Resource*:

```
<authorizationRequest>
<action>POST</action>
<resource>RealmA/index.html</resource>
<sessionToken>affl;;alkf;l;fd</sessionToken>
</authorizationRequest>
```

HTTP return Code 200:

```
<authorizationResult >
<message>The user is authorized.</message>
<resultCode>AUTHORIZED</resultCode>
</authorizationResult >
```

Configure the Security Token Service

CA SiteMinder® SPS supports Secure Token Service (STS) for Office 365 to provide a WS-Trust-based mechanism for token issuance and translation. You can deploy one or multiple STS instances on a CA SiteMinder® SPS machine.

Deploy Multiple CA SiteMinder® SPS Instances

To deploy multiple STS instances, all the STS instances must have the same log4j configuration so that each STS instance logs in to the individual log file.

Follow these steps:

1. Perform *one* of the following tasks:
   - On Windows, perform the following steps:
     a. Navigate to *installation_home*/proxy-engine/conf.
     b. Open the SmSpsProxyEngine.properties file and uncomment the STS_AGENT_LOG_CONFIG_FILE variable in the file.
     c. Save the changes.
   - On UNIX, perform the following steps:
     a. Navigate to *installation_home*/proxy-engine.
     b. Open the proxyserver.sh file and uncomment the STS_AGENT_LOG_CONFIG_FILE variable in the file.
     c. Save the changes.
2. Navigate to installation_home/proxy-engine/conf/sts-config/globalconfig.
3. Open the agent-multiinstance-log4j.xml file.
4. Perform the following steps for each STS instance:
   a. Create an appender for the STS instance.

      Note: By default, the file contains one appender for an STS instance.
   b. Replace [SPS ROOT FOLDER] with the CA SiteMinder® SPS root folder path in the appender.
   c. Replace [STS Service Name] with the service name of the STS instance in the appender.
5. Save the changes.
6. Restart CA SiteMinder® SPS.

   A log file for each STS instance is created in *installation_home*/proxy-engine/logs with the following format:

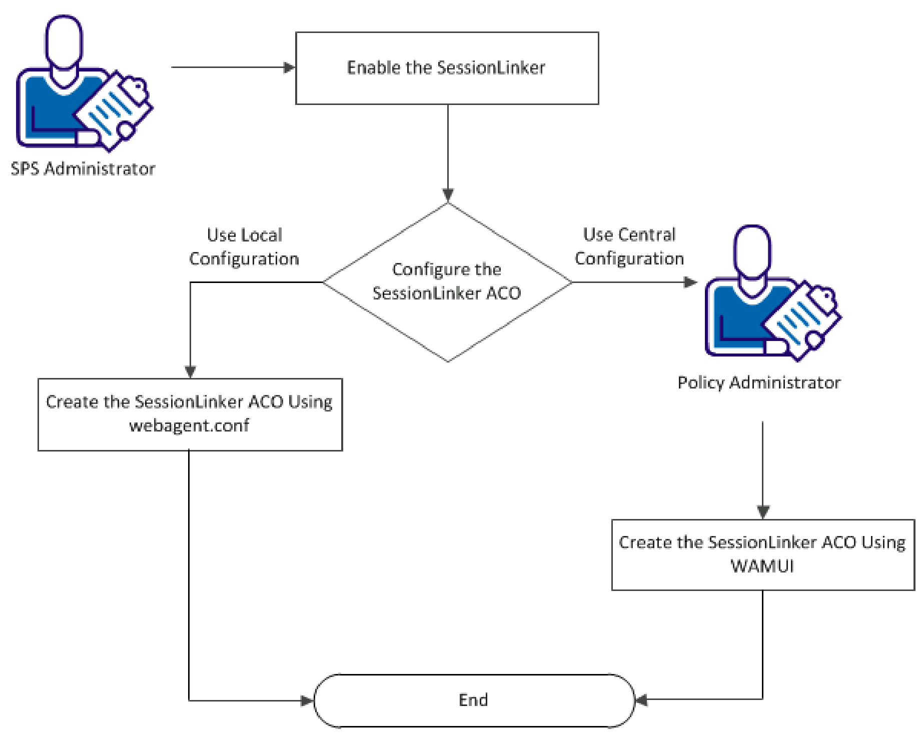
   `STS_service_name.log`
7. Verify that each STS instance logs in to the individual log files.

Configuring SPS to Support the SessionLinker

The SessionLinker synchronizes the CA SiteMinder® session with the third-party application session for better security. By default, CA SiteMinder® SPS installs SessionLinker in a disabled mode.

The following diagram describes how the CA SiteMinder® SPS administrator and policy administrator can configure CA SiteMinder® SPS to support the SessionLinker:

Configuring the SPS to Support the SessionLinker


SPS Administrator
Enable the SessionLinker
Use Local Configuration
Configure the SessionLinker ACO
Use Central Configuration
Policy Administrator
Create the SessionLinker ACO Using webagent.conf
Create the SessionLinker ACO Using WAMUI
End

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 CA SiteMinder® SPS server.

Create the NPS_Session_Linker ACO

CA SiteMinder® 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 (the number of seconds in a 24 period)

Limits: Cannot be *less* than the *maximum number of seconds* that cookies from the third party (foreign) application are accepted.

To configure CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® 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.

Working with Cookies

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.

SessionLinker Troubleshooting

If an error occurs, consider the following possibilities to troubleshoot the error:

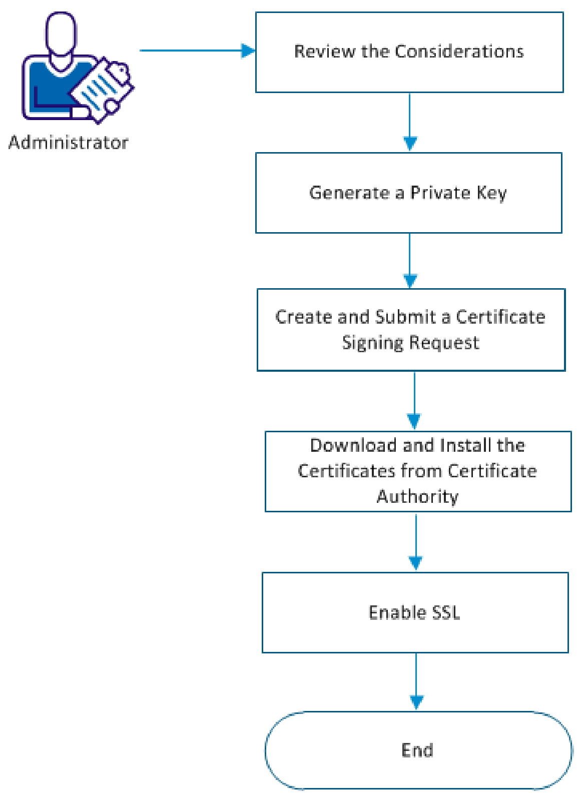
- Verify that the valid SMSESSION cookie and FOREIGN SESSION cookie are set at the user and are passed to the CA SiteMinder® SPS.
- If you enabled the SessionLinker using the webagent.conf file, verify that the web agent is enabled.
- Verify that the SessionLinker ACO syntax is correct.
- If agent tracing is enabled in the SessionLinker ACO, verify the logs and trace messages in the agent logs and trace.
- Verify that the CA SiteMinder® 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 CA SiteMinder® 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.

Configuring SSL for SPS

CA SiteMinder® SPS supports SSL to provide a secure communication between client and server. CA SiteMinder® SPS uses the OpenSSL cryptography toolkit that implements the SSL v2/v3 and Transport Layer Security (TLS v1) network protocols and related cryptography standards that are required by the 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 *installation_home*\SSL\bin folder or the corresponding UNIX directory.

The following flowchart describes how to configure SSL for CA SiteMinder® SPS:

Configure SSL for CA SiteMinder® SPS


Administrator
Review the Considerations
Generate a Private Key
Create and Submit a Certificate Signing Request
Download and Install the Certificates from Certificate Authority
Enable SSL
End

Follow these steps:

1. Review the considerations (see page 162).
2. Generate a private key using *one* of the following steps:
   - Generate a private encrypted RSA server key (see page 163).
   - Generate a private unencrypted RSA server key (see page 164).
3. Generate and submit a certification signing request using *one* of the following steps:
   - Generate and submit a certification signing request to a Certification Authority (see page 165).
   - Generate and self-sign a certification signing request (see page 165).
4. Download and install the certificates from the Certification Authority (see page 166).
5. Enable SSL using *one* of the following steps:
   - Enable SSL for an encrypted private key (see page 168).
   - Enable SSL for an unencrypted private key on Windows (see page 167).
   - Enable SSL for an unencrypted private key on UNIX (see page 167).

   SSL is configured.

Review the Considerations

Before you configure SSL, review the following information about private keys and server certificates:

- As the server certificate and private key work together, use the server certificate with the corresponding private key.
- The server certificate must be digitally signed by a Certificate Authority (CA). If you want to enable SSL for an internal demo, the server certificate may be self-signed with your own private key.
- The SSLCertificateFile and SSLCertificateKeyFile directives in the SSL.conf file must point to the corresponding certificate and key files.
- If you are using Apache virtual host feature, each virtual host you want to secure must have its own private key and server certificate.

Generate a Private Key

SSL uses keys to encrypt and decrypt messages. Keys come in pairs: one public key, and one private key.

Keys use various cryptographic algorithms and key exchange methods. For generating private keys for use with certificates, the RSA key exchange method with the Date Encryption Standard (DES) cryptographic algorithm is commonly used. The key output file is in an encrypted ASCII PEM format.

Generate a Private Encrypted RSA Server Key

If you want to protect your server key, generate a private encrypted RSA server key.

Follow these steps:

1. Open a command-line window.
2. Navigate to the following directory

   ```
   installation_home\SSL\bin
   ```

   installation_home

   Defines the directory where CA SiteMinder® SPS is installed.

   Default: (Windows) [32-bit] C:\Program Files\CA\secure-proxy

   Default: (Windows) [64-bit] C:\CA\secure-proxy

   Default: (UNIX/Linux) /opt/CA/secure-proxy

3. Execute the following command:

   ```
   .\openssl genrsa -des3 -out ..\keys\server.key [numbits]
   ```

   server

   Specifies the fully qualified domain name of the server.

   (Optional) ***numbits***

   Specifies the size in bits of the private key that must be generated.

   Default: 1024

   Range: 1024 - 2048

   The private encrypted server keys are created and written to the specified key output file. The key output file is in the encrypted ASCII PEM format. As the file is encrypted, you will be prompted for a pass-phrase to protect it and decrypt it later if you want.

Generate a Private Unencrypted RSA Server Key

If you do not want to protect your server key, generate a private unencrypted RSA server key.

Follow these steps:

1. Open a command-line window.
2. Navigate to the following directory

   ```
   installation_home\SSL\bin
   ```

   installation_home

   Defines the directory where CA SiteMinder® SPS is installed.

   Default: (Windows) [32-bit] C:\Program Files\CA\secure-proxy

   Default: (Windows) [64-bit] C:\CA\secure-proxy

   Default: (UNIX/Linux) /opt/CA/secure-proxy
3. Execute the following command:

   ```
   .\openssl genrsa -out ..\keys\server.key [numbits]
   ```

   server

   Specifies the fully qualified domain name of the server.

   (Optional) *numbits*

   Specifies the size in bits of the private key that must be generated.

   Default: 1024

   Range: 1024 - 2048

   The private unencrypted server key is generated.

Generate and Submit a Certificate Signing Request

Generate a certificate request or Certificate Signing Request using the private key. You can send the Certificate Signing Request to a Certificate Authority for signing into a certificate, or you can create a self-signed certificate for an internal demo.

Generate and Submit a Certificate Signing Request to a Certificate Authority

Generate a Certificate Signing Request and submit it to a Certificate Authority that your organization uses.

Follow these steps:

1. Open a command-line window.
2. Execute the following command:

   ```
   .\openssl req -config .\openssl.cnf -new -key ..\keys\server.key -out
   ..\keys\server.csr
   ```
3. Enter the values as prompted.

   The system generates a certificate request with the certificate file name and a request number.
4. (Optional) Record the file name and Certificate Signing Request for the future reference.
5. Submit the Certificate Signing Request to the Certificate Authority.

Generate a Self-signed Certificate

If you want to enable SSL for an internal demo, generate a self-signed certificate.

Follow these steps:

1. Open a command-line window.
2. Execute the following command:

   ```
   .\openssl req -config .\openssl.cnf -new -x509 -key ..\keys\server.key -out
   ..\certs\cert_name.crt
   ```
3. Place the output in the following location:

   sps_home\SSL\certs

   A self-signed certificate is generated.

Download and Install the Certificates from the Certificate Authority

Download the signed certificates from the Certificate Authority.

Follow these steps:

1. Log in to the CA SiteMinder® SPS host from which you issued the certificate requests.
2. Open the httpd-ssl conf file.

   Default Path: *installation_home*\httpd\conf\extra\httpd-ssl.conf
3. Verify that the directives of the server key and certs are correct.
4. Set the value of the SSLPassPhraseDialog variable to custom.
5. Set the value of the SSLCustomPropertiesFile variable to *installation_home*>\httpd\conf\spsapachessl.properties.
6. Verify that the reference to RootCA is set.
7. Perform the following steps to add RootCA or the self-signed certificate into ca-bundle.cert:
   a. Open the certificate in a notepad and copy the lines from BEGIN to END.
   b. Open ca-bundle.cert in a notepad and paste the lines of the certificate at the end.
   c. Save the changes.

Enable SSL

You can enable SSL for an encrypted or unencrypted private key.

Enable SSL for an Unencrypted Private Key on Windows

To enable SSL for an unencrypted private key on Windows, generate the spsapachessl.properties file.

Follow these steps:

1. Open a command-line window with administrative privileges.
2. Navigate to the following directory:

   ```
   installation_home\httpd\bin
   ```

3. Run the following script file:

   ```
   configssl.bat -enable
   ```

   Note: If an overwrite warning appears, confirm that you want to overwrite the existing spsapachessl.properties file.

   SSL is configured.

Enable SSL for an Unencrypted Private Key on UNIX

To enable SSL for an unencrypted private key on UNIX, edit the spsapachessl.properties located in the following location:

installation_home/httpd/conf/spsapachessl.properties

Follow these steps:

1. Open the spsapachessl.properties file in a text editor.
2. Add or edit the following line:
3. Perform one of the following tasks:
   - If apache.ssl.enabled= exists in the file, set the line to the following value:

     ```
     apache.ssl.enabled=Y
     ```

   - If apache.ssl.enabled= does not exist in the file, add the line in the following format:

     ```
     apache.ssl.enabled=Y
     ```

4. Save the changes.

   SSL is configured.

Enable SSL for an Encrypted Private Key

To enable SSL for an encrypted private key, generate the spsapachessl.properties file.

Follow these steps:

1. Open a command-line window with administrative privileges.
2. Navigate to the following directory:

   Windows

   ```
   installation_home\httpd\bin
   ```

   UNIX

   ```
   installation_home/httpd/bin
   ```

3. Run the following script:

   Windows

   ```
   configssl.bat -enable passphrase
   ```

   UNIX

   ```
   configssl.sh passphrase
   ```

   Note: The passphrase value must match the passphrase value of the server key. If an overwrite warning appears, confirm that you want to overwrite the existing spsapachessl.properties file.

4. The passphrase is encrypted and is stored in the spsapachessl.properties file.
5. Restart the Secure Proxy Service.

   SSL is configured.

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>
```

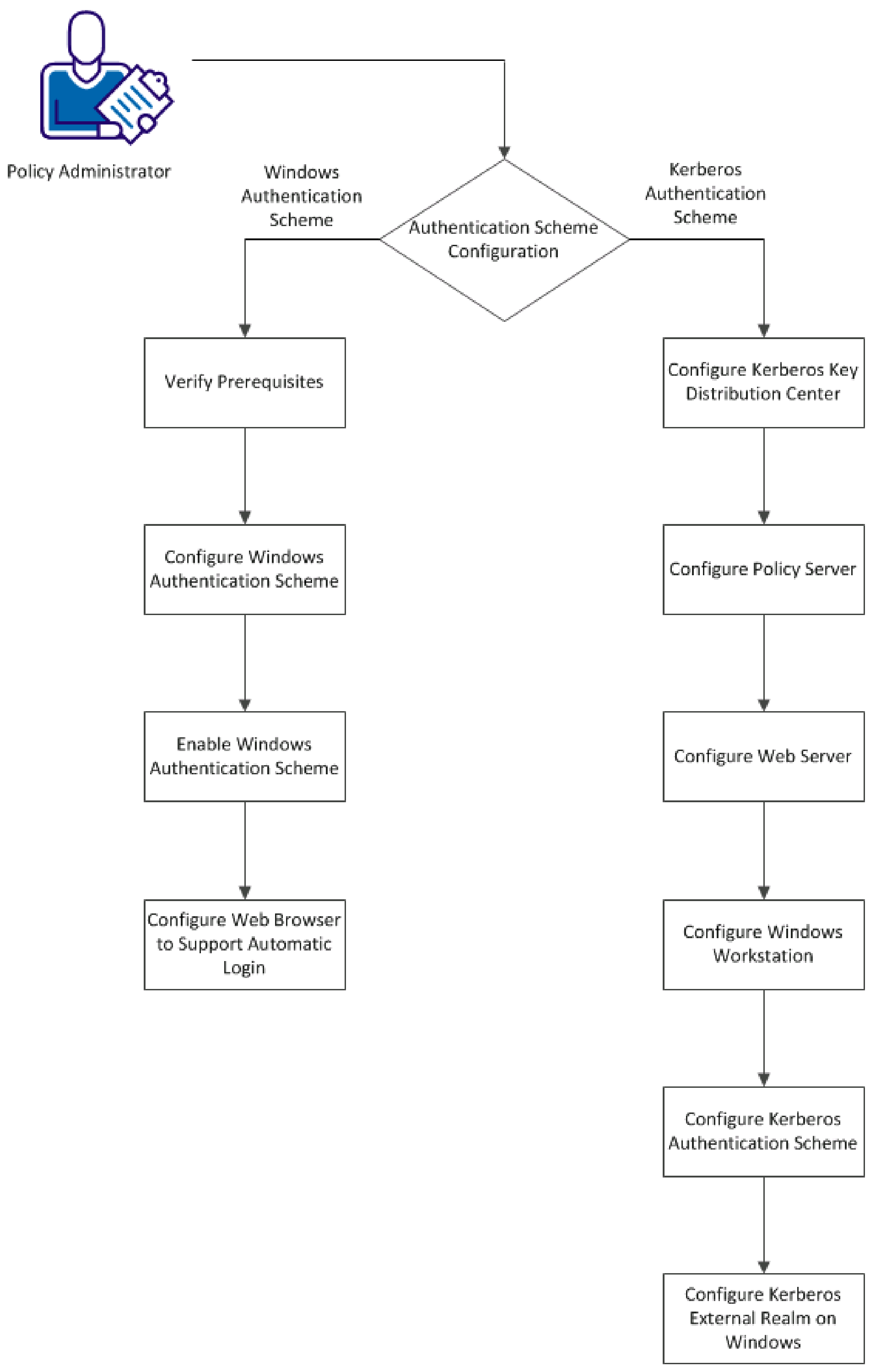
Configure SPS to Support Integrated Windows Authentication

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 CA SiteMinder® SPS to support IWA:

Support for Integrated Windows Authentication


Policy Administrator
Authentication Scheme Configuration
Windows Authentication Scheme
Kerberos Authentication Scheme
Verify Prerequisites
Configure Windows Authentication Scheme
Enable Windows Authentication Scheme
Configure Web Browser to Support Automatic Login
Configure Kerberos Key Distribution Center
Configure Policy Server
Configure Web Server
Configure Windows Workstation
Configure Kerberos Authentication Scheme
Configure Kerberos External Realm on Windows

You can configure *one* of the following authentication schemes with CA SiteMinder® SPS:

- Windows authentication scheme on Windows server
- Kerberos authentication scheme on Windows and UNIX server

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 CA SiteMinder® SPS to support IWA:

1. Configure a Windows domain controller.
2. Add CA SiteMinder® 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.
3. Click Create Authentication Scheme.

   Verify that the Create a new object of type Authentication Scheme is selected.

   Click OK
4. Enter a name and protection level.
5. Select Windows Authentication Template from the Authentication Scheme Type list.

   Scheme-specific settings appear.

6. 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 CA SiteMinder® 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

7. Click Submit.

   The authentication scheme is saved and can be assigned to a realm.

Note: If a user authentication fails in NTLM authentication, the authentication process continues until the browser stops it. To resolve the issue, create the following redirect responses that redirect the user to a custom page when the authentication fails:

- Rule with onauthreject and onauthusernotfound
- Response with Webagent-onreject-redirect

Enable Windows Authentication Scheme

The CA SiteMinder® SPS now supports Windows authentication scheme that is configured on the Policy Server. To let the CA SiteMinder® SPS support the Windows authentication scheme, perform the following steps:

1. Create the WindowsNativeAuthentication ACO parameter 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 parameter, the CA SiteMinder® 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.

CA SiteMinder® 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.

CA SiteMinder® 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. CA SiteMinder® 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. **Example:** HTTP/win2k8sps.test.com@TEST.COM
SmpsServicePrincipal	Specify the Policy Server principal name. **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/sol10
        ps.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. **Example:** HTTP/win2k8sps.test.com@TEST.COM
SmpsServicePrincipal	Specifies the Policy Server principal name. **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.t
        est.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.

Configure Security Zones on CA SiteMinder® SPS

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.

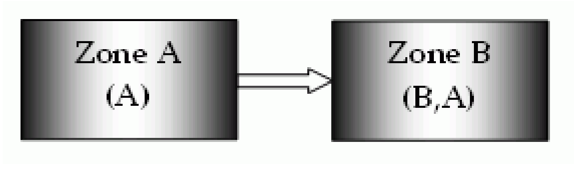
With SSO Security Zones, APP1 can be placed in zone Z1 and APP2 can be placed in zone Z2. Now logging into APP1 creates a Z1SESSION cookie and access to APP2 results in a Z2SESSION cookie. With different names, the cookies no longer overwrite each other so there is only one login per application now, 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 CA SiteMinder® SPS Security Zones

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

- Configure security zones on multiple CA SiteMinder® 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 CA SiteMinder® SPS server.

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

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

To configure security zones on multiple Web servers of an CA SiteMinder® 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 5: Using CA SiteMinder® SPS APIs

This section contains the following topics:

Session Scheme API

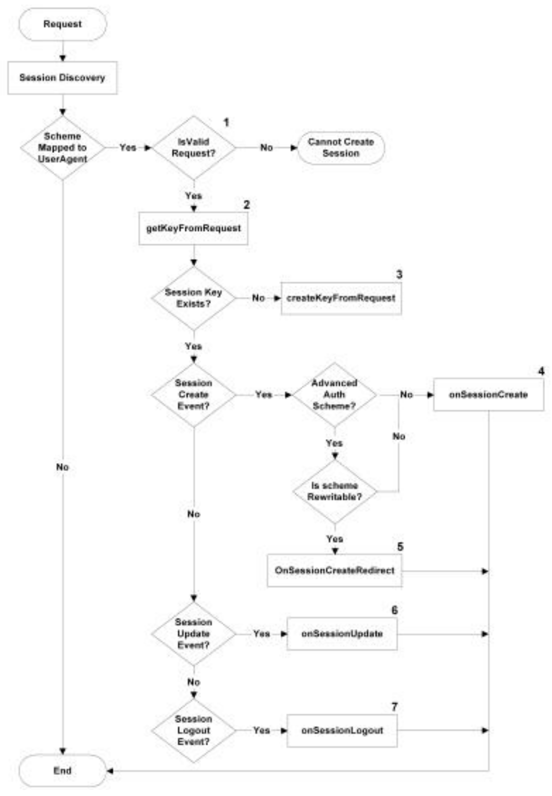
The CA SiteMinder® SPS supports a Java API that allows you to develop custom session schemes. These schemes can be assigned to user agent types for each virtual host configured on the CA SiteMinder® SPS.

Overview of Session Scheme API Processing

The CA SiteMinder® 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 200).

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 CA SiteMinder® 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/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/lib.
   - Add the class files for your custom session scheme in a jar in the *sps_home/*Tomcat/lib directory then configure scheme in the CA SiteMinder® SPS server.conf file.
7. Restart the CA SiteMinder® 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 CA SiteMinder® 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.

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 CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® SPS.

Session Storage API

The CA SiteMinder® 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. CA SiteMinder® 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 CA SiteMinder® SPS can process a single custom filter or a group of custom filters for each request. When you create a custom filter group, the CA SiteMinder® 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 203)

How SPS Processes Custom Filters

The CA SiteMinder® SPS includes an API for inserting pre-processing and post-processing into the proxy stage of a request.

In a standard CA SiteMinder® SPS transaction, the following process occurs:

1. User requests a resource.
2. CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® 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 CA SiteMinder® SPS receives a request or a response, the CA SiteMinder® 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 CA SiteMinder® 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) Performs the filtering. Parameters: request - the proxy request data response - the proxy response data Throws: ProxyFilterException - thrown if failure processing filtering.
ProxyFilterConfig	getFilterConfig() Returns this filter's ProxyFilterConfig object. (ProxyFilterConfig object that initialized this filter).
void	init(ProxyFilterConfig config) Called when the filter is created to perform any required initialization. Parameters: config - a ProxyFilterConfig object containing the filters's configuration and initialization parameters Throws: 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 CA SiteMinder® SPS transactions, as listed in the following table.

Return Value	Method
Void	doFilter(ProxyRequest prequest, ProxyResponse presponse) throws ProxyFilterException 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. Specified by: doFilter in interface ProxyFilter Parameters: request - the proxy request data response - the proxy response data Throws: ProxyFilterException - thrown if failure processing filtering
Void	doPreFilter(ProxyRequest prequest, ProxyResponse presponse) throws ProxyFilterException Performs pre-filtering. Override this method to perform filtering tasks before the request is sent to the target server. Parameters: request - the proxy request data response - the proxy response data Throws: ProxyFilterException - thrown if failure processing filtering
Void	doPostFilter(ProxyRequest prequest, ProxyResponse presponse) throws ProxyFilterException Performs post-filtering. Override this method to perform filtering tasks after the response is received from the target server. Parameters: request - the proxy request data response - the proxy response data Throws: ProxyFilterException - thrown if failure processing filtering

Return Value	Method
ProxyFilterConfig	getFilterConfig() Returns this filter's ProxyFilterConfig object. Specified by: getFilterConfig in interface ProxyFilter Returns: ProxyFilterConfig the ProxyFilterConfig object that initialized this filter.
Void	init(ProxyFilterConfig config) throws ProxyFilterException Called when the filter is created to perform any required initialization. **Note:** When overriding this method, the first statement should call the parent init method "super.init(config);". Specified by: init in interface ProxyFilter Parameters: config - a ProxyFilterConfig object containing the filters's configuration and initialization parameters Throws: 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() Returns the name of this filter.

Return Value	Method
java.lang.String	getInitParameter(java.lang.String name) Returns a String containing the value of the named initialization parameter, or null if the parameter does not exist. Parameters: name - a String specifying the name of the initialization parameter
java.util.Enumeration	getInitParameterNames() 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) Adds a header with the specified name and value. This method allows response headers to have multiple values. Parameters: name - a String specifying the header name value - a String specifying the header value
byte[]	getContent() 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) 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. Parameters: name - a String specifying the header name

Return Value	Method
java.util.Enumeration	getHeaderNames() Returns an Enumeration of all the header names. If no headers exist, this method returns an empty Enumeration.
int	getStatusCode() Returns the HTTP response status code of the response to the proxy request.
java.lang.String	removeHeader(java.lang.String name) 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. Parameters: name - a String specifying the header name
void	setContent(byte[] content) Sets the content of the response to the proxy request. This overwrites the content to be returned to the proxy client. Parameters: content - the byte array containing the content
void	setHeader(java.lang.String name, java.lang.String value) Sets a header with the specified name and value. If a header with the same name exists it will be overwritten. Parameters: name - a String specifying the header name 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. Parameters: message - Message of exception
ProxyFilterException(java.lang.String message, java.lang.Throwable rootCause)	Constructs a new ProxyFilterException with the specified message and root cause. Parameters: message - Message of exception 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. Parameters: 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) 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. Parameters: name - a String specifying the header name
java.util.Enumeration	getHeaderNames() Returns an Enumeration of all the header names. If no headers exist, this method returns an empty Enumeration.
javax.servlet.http.HttpServletRequest	getOriginalRequest() Returns the original HttpServletRequest made to the proxy.

Return Value	Method
java.lang.String	getSessionKey() 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. **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() 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() 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() 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() 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) 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. Parameters: name - a String specifying the header name

Return Value	Method
void	setHeader(java.lang.String name, java.lang.String value) Sets a header with the specified name and value. If a header with the same name exists it will be overwritten. Parameters: name - a String specifying the header name value - a String specifying the header value
byte[]	getContent() 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) Sets the content of the POST data to the proxy request. This overwrites the content to be sent to the backend server. Parameters: 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/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/lib directory.
   - Add the class files for your filter in a jar in the *sps_home/*Tomcat/lib directory.
6. Configure the CA SiteMinder® 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 CA SiteMinder® SPS.

Filter API Example

The CA SiteMinder® 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 CA SiteMinder® SPS. For absolute links to be handled properly by the CA SiteMinder® 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 CA SiteMinder® SPS request.

Chapter 6: Troubleshooting

This section contains the following topics:

A Pop-up Window Appears in the Browser after SSL Configuration

Symptom:

When I access CA SiteMinder® SPS using an internet browser after SSL configuration, a pop-up window is displayed.

Solution:

If you use an internet browser with weak encryption algorithm to access CA SiteMinder® SPS for the first time after configuring SSL, a pop-up window is displayed. You can allow the pop-up and can continue to access CA SiteMinder® SPS. If you do not want the pop-up to display, perform one of the following steps:

- Use strong internet browsers such as Internet Explorer 8.0, Chrome, Mozilla Firefox that support strong encryption algorithms.
- If you are not using an X509 based authentication scheme, perform the following steps:
  a. Navigate to <install_path>/httpd\conf\extra\.
  b. Open the httpd-ssl.conf file.
  c. Navigate to SSLVerifyClient and set the value to none.
  d. Save the change.
  e. Restart CA SiteMinder® SPS.

Unable to Start Apache on UNIX systems

Symptom:

When running the CA SiteMinder® 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 CA SiteMinder® SPS startup command again (sps-ctl start or startssl).

Non-english Input Characters Contain Junk Characters

Valid on UNIX/Linux

Symptom:

Some non-English input characters are not displayed correctly in the console window.

Solution:

Verify the terminal settings of your console window. Confirm that the console does not clear high (eight) bit of input characters. Execute 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 is Cached for Every Request

Symptom:

I do not want CA SiteMinder® SPS to cache the DNS name look-up settings of the server.

Solution:

The CA SiteMinder® 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 NETE_SPS_JAVA_HOME\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 CA SiteMinder® SPS caches the successful DNS name look-ups. Enter a positive integer. If you enter a negative value, the CA SiteMinder® SPS caches the DNS settings.

   Default: -1

Resource Request Fails

Symptom:

CA SiteMinder® 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 217)
Configure SPSAgentTrace Logs (see page 218)
Configure the mod_jk.log File (see page 219)
Configure the httpclient.log File (see page 219)

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.
4. Restart CA SiteMinder® SPS.

   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.

The Installation Program Displays Warnings

Valid on UNIX

Symptom:

When I install CA SiteMinder® SPS, the installation wizard displays warnings that few files must be configured manually.

Solution:

If you do not have root permissions, you can install CA SiteMinder® SPS but the automatic installation process cannot complete all the installation steps. The installation wizard displays warnings that help you determine the files that must be configured manually.

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:

CA SiteMinder® 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 commands:
  - 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 are using SSL, be sure that 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 CA SiteMinder® SPS using a browser.

Solution:

To access the CA SiteMinder® 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.

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/

Virtual Hosts Configuration Fails

Symptom:

The configuration of virtual hosts fails.

Solution:

For information about configuring virtual hosts, see www.apache.org.

SPS Not Forwarding Requests

Symptom:

When I access a resource, the 404 File Not Found browser error is displayed and the action is not logged in the Web Agent log.

Solution:

Verify the name and IP address of the virtual host in the server.conf file.

Error in Accessing a SharePoint Page

Symptom:

When accessing a SharePoint page through the SPS, the CA SiteMinder® 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 CA SiteMinder® SPS proxyrule.xml file, the backend is an internal URL with a public URL pointing to the CA SiteMinder® 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