CA SiteMinder®

SAML Affiliate Agent Guide

6.x QMR 6

CA Product References

This document references the following CA products:

- SiteMinder®
- TransactionMinder®
- Identity Manager

Contact CA

Contact CA Support

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

- 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

Provide Feedback

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

If you would like to provide feedback about CA product documentation, complete our short customer survey, which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 1: SAML Affiliate Agent Overview

This section contains the following topics:

Introducing the SAML Affiliate Agent

The SAML Affiliate Agent is part of SiteMinder's Federation Security Services, which enables businesses with partner relationships to share security and customer profile information. It is a stand-alone component that provides single sign-on and session management capabilities to a consumer site that does not use the SiteMinder Policy Server and SiteMinder Web Agent.

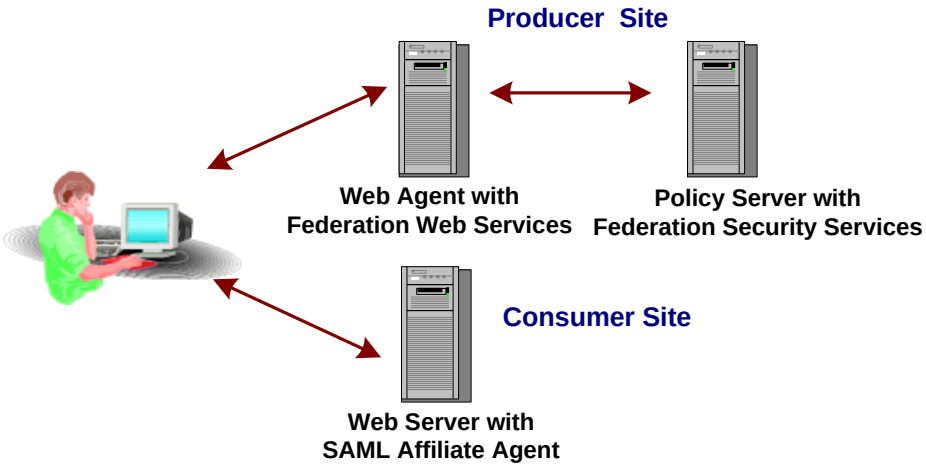
The SAML Affiliate Agent supports only the SAML 1.0 specification.

Note: Federation Security Services and the SAML Affiliate Agent are separately-licensed from SiteMinder.

SAML is the Security Assertion Markup Language, developed by the Organization for the Advancement of Structured Information Standards (OASIS). SAML defines an XML framework for exchanging authentication and authorization information.

When business partners share security and user profile information, they can provide their customers with seamless access to services across a federated network. For example, if a customer visits a producer then proceeds to a partner site within the federated network, the partner might present the user with a customized welcome page that includes their name and customer information. The partner gets this information from the producer.

The following illustration shows the partner relationship:



At the producer site, there is a full SiteMinder installation. At the consumer site there is only a SAML Affiliate Agent; there is no Policy Server or Web Agent. The SAML Affiliate Agent provides user information to a Web server for use with its Web applications, and it protects only those resources specified in its configuration file. The applications can use the provided information to personalize Web content for each user.

The SAML Affiliate Agent:

- Provides a seamless connection from the producer to the consumer without requiring a user to re-identify or provide additional information about themselves.
- Extracts user information from the producer. This information could include a user's buying preferences, the identity of the producer, or demographic information that is useful for tracking customers. This information can then be set as cookies or header variables for the Web server at the consumer.
- Can determine that the user has been registered at the producer, and optionally, that the user has an active SiteMinder session.
- Identifies the user without requiring a full SiteMinder installation at the consumer site.
- Allows Web applications to personalize Web content based on the information received from the producer.
- Can share a session with the producer to provide single sign-on and single sign-off.
- Can notify the producer when users visit specific resources at the consumer.

Federation Security Services Terminology

The following terms describe the SAML Affiliate Agent and related federation security services functions.

affiliate

An organization within a network that has a business relationship with a main business producer. Affiliates typically share user and security information with other businesses.

Affiliate Server

The UNIX daemon or NT service that communicates to the producer on behalf of the consumer.

affiliate Web server plug-in

The plug-in to the Web server where the SAML Affiliate Agent is installed.

assertion

A piece of data, transmitted in an XML document, that contains authorization and authentication information about a user. The assertion is used to facilitate secure business transactions.

assertion generator

Component at the producer that creates the SAML assertion and SAML artifact.

consumer

A site that uses a SAML assertion to authenticate a user.

Federation Security Services

Application installed at the producer that enables businesses to share security and user profile information via SAML assertions.

federation web services

Application installed on the producer-side Web Agent. These services provide assertion retrieval, session synchronization, and notification services.

portal

An entry site for users to connect to related sites across the Internet. In a SiteMinder network, the producer is the site where the Policy Server and Web Agent reside and which has the user's identity.

PortalQueryURL

URL at the producer. The SAML Affiliate Agent sends a user to this URL if that user does not have the profile and session cookies to access an affiliate resource.

producer

A site that generates SAML assertions.

producer-side Web Agent

The Web Agent installed on the Web server at the producer that handles the authentication of users trying to access resources at the consumer.

SAML Affiliate Agent

The SiteMinder Agent that is installed at a consumer site. This Agent consists of the affiliate Web server plug-in and the Affiliate Server.

SSLInterceptorURL

A URL at the consumer that enables the SAML Affiliate Agent to receive the SAML artifact. After the user's initial visit to the producer, the Web Agent redirects the user's browser to this URL at the consumer. The SAML artifact is added to the redirect URL. After the SAML Affiliate Agent receives the artifact, it makes a call on the SSL back channel to the producer to retrieve the assertion associated with that artifact. Using SSL ensures that the artifact is encrypted and is not sent in clear text.

SAML artifact

A 42-byte, hex-encoded ID that references an assertion stored with the session server at the producer. The artifact enables the SAML Affiliate Agent to retrieve an assertion document from the producer.

SAML assertion

See assertion.

SSL back channel

SSL channel between the producer and the consumer used to exchange data, such as the assertion document. This channel does not go through a user's browser.

Components of the SAML Affiliate Agent

The SAML Affiliate Agent combines these two components:

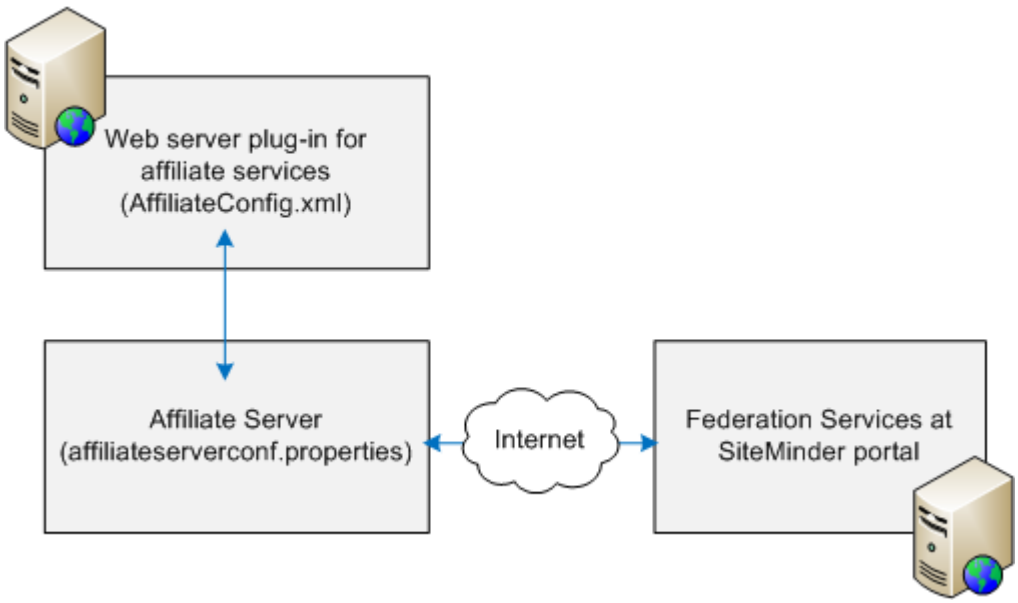
- Web server plug-in for affiliate services

  This component is a plug-in to the Web server where the SAML Affiliate Agent is installed. The plug-in is configured through the AffiliateConfig.xml file.

- Affiliate server

  This is a UNIX daemon or NT service, depending on the Web server operating system. The server is configured through the affiliateserverconf.properties file. The Affiliate Server communicates to the producer on behalf of the affiliate Web server plug-in. This server must be up and running for the SAML Affiliate Agent to operate.

The following illustration shows the SAML Affiliate Agent components:



How the SAML Affiliate Agent is Invoked

When a user accesses a protected resource on a Web server at a consumer site, the SAML Affiliate Agent intercepts the request and tries matching the resource to a protected consumer resource specified in the Agent's configuration file. If a match is found, the SAML Affiliate Agent redirects the user's browser to the producer to request user information. If the requested resource does not match what is defined in the SAML Affiliate Agent's configuration file, it passes the request to the consumer Web server, which processes the request as usual.

Communication Across the Federated Network

At the consumer site, certain resources are designated as affiliate resources. A user can access an affiliate resource by selecting links at the producer or consumer site. When a user tries to access a protected affiliate resource, the SAML Affiliate Agent redirects the user's browser back to the producer site. The producer authenticates the user and collects the user profile information.

When the producer-side Web Agent receives a request from the SAML Affiliate Agent, it first verifies to see if a session cookie for the user exists. This cookie is present if the user has an active SiteMinder session. If there is no session cookie, the Web Agent checks for an identity cookie. The identity cookie, a persistent cookie that stores the user's identity, is present only if user tracking is enabled at the Policy Server.

If the user is authenticated at the producer before visiting the consumer, the Web Agent can create the session or identity cookie. If either cookie exists, the Web Agent sends the identity information to the Policy Server so it can gather specific user information, which is placed in an assertion and retrieved by the SAML Affiliate Agent at the consumer site.

Using information in the SAML assertion, the SAML Affiliate Agent creates local cookies in the browser and uses these cookies to maintain user and session information for its applications.

If neither the session or identity cookie exists at the producer, the producer-side Web Agent challenges the user for credentials so that at least one of the cookies can be issued. If the user presents invalid credentials, the producer denies the user access.

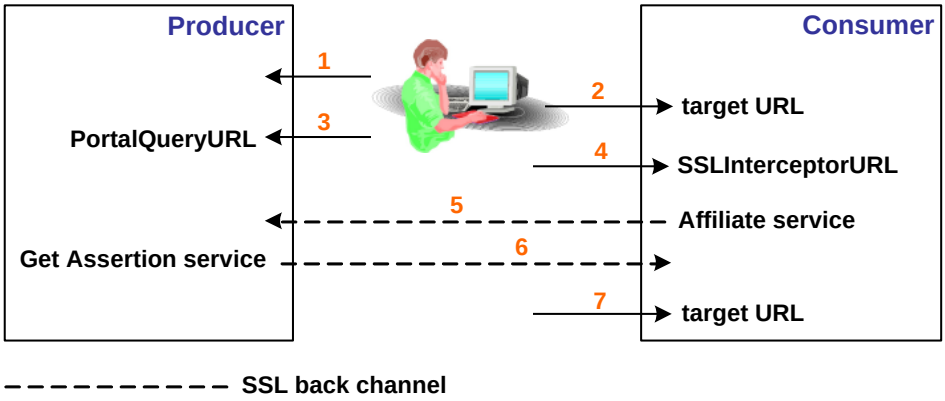
If the user authenticates at the producer, but the producer redirects the user back to the consumer without a SAML artifact, the SAML Affiliate Agent denies the user access to the resource because the user is unknown. How the SAML Affiliate Agent handles unidentified users depends on how you configure the Agent.

More information:

Manage User Access to Consumers (see page 101)
Scenario 1: User Visits Producer First (see page 16)
Scenario 2: User Visits Consumer Before Visiting the Producer (see page 18)

Scenario 1: User Visits Producer First

When the user visits the producer before going to the consumer, the producer and consumer communicate as follows:



The communication flow is as follows:

1. The user authenticates at the producer site, and the Web Agent sets a cookie (session, identity or both) in the user's browser.
2. The user selects a protected resource at the consumer through a link at the producer.

   The SAML Affiliate Agent does not recognize the user because this is the user's first visit to the consumer site.
3. The SAML Affiliate Agent adds the SSLInterceptorURL and the target URL to the redirect URL, then redirects the request to the producer (specifically, to the PortalQueryURL) to retrieve user information.

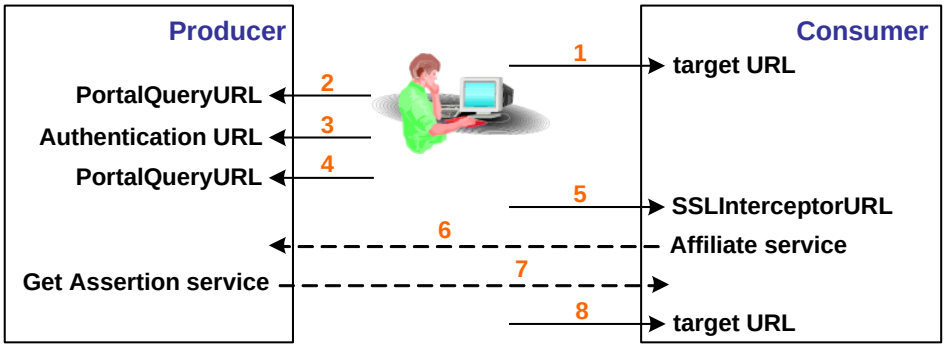
   At the producer, the assertion generator creates a SAML assertion from response data sent by the Policy Server. The assertion is stored in a persistent session store. The assertion generator also creates a SAML artifact to identify the assertion, which is appended to the redirect URL sent to the consumer.
4. The user is redirected back to the consumer, to the SSLInterceptorURL, with the artifact and the target URL.
5. The Affiliate Server uses the artifact to request the assertion.
6. The producer returns the assertion to the consumer.

   The SAML Affiliate Agent validates the assertion, produces a local session cookie, and sets headers based on information from the assertion.
7. Finally, the user is directed to the originally requested target URL.

If the user returns to the consumer site for a second time, the SAML Affiliate Agent refers to the information in the local cookies. These cookies are transient; the information is only valid for the duration of the user's browser session or for the period specified by the configured timeout value.

Scenario 2: User Visits Consumer Before Visiting the Producer

When the user visits the consumer first, the consumer and the producer communicate as follows:



The communication flow is as follows:

1. A user tries to access a resource at the consumer before logging in at the producer.
2. The Affiliate Agent redirects the user to a URL at the producer (known to the Affiliate Agent as the PortalQueryURL). The SAML Affiliate Agent appends the SSLInterceptorURL and the target resource to the URL.
3. At the producer, the user is sent to the AuthenticationURL, and prompted to log in. If the log in is successful, the AuthenticationURL invokes the redirect.jsp file, which redirects the user back to the PortalQueryURL.
4. The PortalQueryURL is now invoked with a session and the assertion generator produces a SAML artifact, which references an assertion.
5. The Web Agent at the producer returns the user to the consumer with the artifact appended to the protected redirect URL, known as the SSLInterceptorURL.
6. The Affiliate Server uses the artifact to request the assertion from the producer.
7. The producer returns the assertion to the consumer. The SAML Affiliate Agent then creates a session for the user.

   The SAML Affiliate Agent validates the assertion, produces a local session cookie, and sets headers based on information from the assertion.
8. Finally, the user is directed to the originally requested target URL.

How a SAML Affiliate Agent Handles Unidentified Users

The SAML Affiliate Agent may not be able to identify a user who tries to access an affiliate resource. This may happen because the user visits the consumer site and does not have an account at the producer site. This user is considered to be an unknown user. Depending on the needs of the consumer site, the SAML Affiliate Agent may or may not deny access to users who have not been authenticated by the producer.

More Information

Manage User Access to Consumers (see page 101)

Features of the SAML Affiliate Agent

The following sections discuss features of the SAML Affiliate Agent.

SAML Assertions

SiteMinder can issue SAML assertions. An assertion is a collection of authentication and authorization data that is passed from the producer to the SAML Affiliate Agent, which uses it to further validate client requests. SAML assertions let companies share user identities and authorization privileges securely.

Note: For information about SAML specifications and background documentation, go to the Organization for the Advancement of Structured Information Standards (OASIS).

When the user attempts to access a resource at a consumer site, the SAML Affiliate Agent redirects them to the producer for authentication. If the user authenticates successfully, the producer generates a SAML artifact. The artifact is returned to the consumer, and the SAML Affiliate Agent uses it to pull the actual assertion document from the producer.

The assertion document holds the entitlement and session information. The SAML Affiliate Agent uses this information to issue local session and profile cookies that permit access to the requested resource. If access is permitted, the user's browser is directed to the target resource. If access is denied, the user's browser is redirected to the URL specified by the NoAccessURL attribute in the SAML Affiliate Agent's configuration file. The NoAccessURL overrides any web server-standard "No Access Allowed" message.

Notifications of Activity at the Consumer

The notification service is used by the SAML Affiliate Agent to notify the producer when users access particular resources at the consumer site.

The elements that make up a notification message include:

- Consumer name
- URL accessed by the user
- User Distinguished Name (DN)
- User session key (if available)
- Time of access

More Information

Configure the Notification Service (see page 114)

Shared Sessions for Seamless Communication

The consumer can share a single session with the producer, in which information about a user's activity and logout at the consumer site is sent from the consumer to the producer. This provides users with a seamless session when they access resources at the producer and the consumer.

At the consumer site, you configure the session model in the AffiliateConfig.xml file. At the producer, the session server, which is a component of the Policy Server, enables shared sessions.

More Information

Shared Sessions (see page 110)

Persistence API and the KeyProvider API

You can extend the capabilities of the SAML Affiliate Agent by using the two C language APIs included with the SAML Affiliate Agent:

- Persistence API—Integrates custom assertion management routines with SAML Affiliate Agent operations.

  Use this API to implement custom persistence libraries.

- Key Provider API—Provides the SAML Affiliate Agent with different encryption keys on a periodic basis. The Agent uses these keys to encrypt and decrypt its cookies.

  Use this API to implement custom key provider libraries.

SAML Affiliate Agent Installation and Configuration

The SAML Affiliate Agent consists of two components:

- Web server plug-in for affiliate services

  This component is a plug-in to the Web server where the Agent is installed.

- Affiliate server

  This is a UNIX daemon or NT service, depending on the web server operating system. This server must be up and running for the SAML Affiliate Agent to operate.

After you install the SAML Affiliate Agent, you must:

- Configure the Affiliate Server by modifying the affiliateserverconf.properties file
- Configure the Affiliate Web server plug-in by modifying the AffiliateConfig.xml file

More Information

Prepare the Installation (see page 23)
Configure a SAML Affiliate Agent (see page 55)

Chapter 2: Prepare the Installation

This section contains the following topics:

Installation Overview

The SAML Affiliate Agent acts as a SAML 1.x consumer so it is installed at the consumer site. Consequently, this overview describes how to set up the consumer site in a SiteMinder federated network. The details for setting-up the producer site are described in the *SiteMinder Federation Security Services Guide*.

Note: Federation Security Services and the SAML Affiliate Agent are licensed separately from SiteMinder.

In the overview procedures:

- The variables *web_agent_home* and *saml_affiliate_agent_home* refer to the installed locations of the Web Agent and SAML Affiliate Agent respectively.
- The procedures refer to the latest SiteMinder releases. For other compatible versions, go to Technical Support and search for the SiteMinder Platform Matrix.

Important! Perform these steps in the order shown.

More Information

Environment Variables Modified by the Agent Installation (see page 181)

Set up the Producer Site

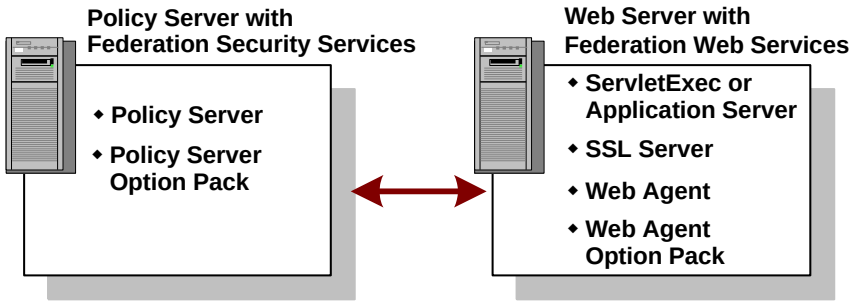
SiteMinder Federation Security Services enables business partners to share security information.

At the producer site, a SiteMinder federated network includes the following components:

- Policy Server—protects the entire site's resources
- Web Server with an installed Web Agent and Web Agent Option Pack—handles the authentication of users requesting a consumer resource.

  The Web Agent Option Pack installs the Federation Web Services application, which acts as the gateway to the Policy Server for the SAML Affiliate Agent.

The following illustration shows the components for Federation Security Services.

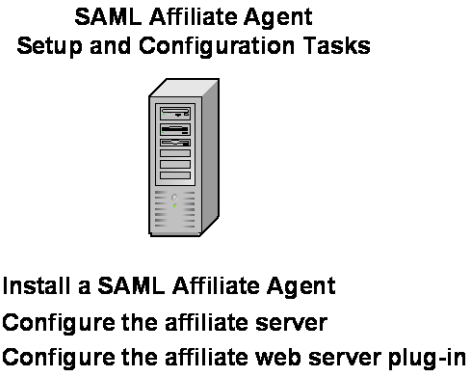


Federation Web Services can be installed on two Agent-side Web servers.

Note: For details on setting up the producer site, see the chapter on setting up a federation security services network in the *Federation Security Services Guide*.

Set Up the Consumer Site

The SAML Affiliate Agent is set up at the consumer site. The following illustration shows the set-up tasks.



Check the Web Server Setup at the Consumer

At the consumer site, set up one of the following configurations:

- Two web server instances with a SAML Affiliate Agent installed on each instance:
  - One web server for use by the SAML Affiliate Agent's Web server plug-in
  - One web server set up as an SSL web server for use with the SAML Affiliate Agent's Web server plug-in

    This server should have a certificate and support SSL connections so it can handle SSL requests that contain security information.
- One web server that supports both HTTP and HTTPS protocols. For example, framework Web Agents can have one web server instance with one SSL port and one non-SSL port for use by the Agent's web server plug-in.

  Important! The SAML Affiliate Agent is *not* compatible with the FIPS 140-2 encryption standards.

For a list of supported web servers and operating systems, go to Technical Support, and search for the SiteMinder Platform Matrix.

Note: Do not install a SAML Affiliate Agent with a SiteMinder Web Agent or XML Agent on the same Web server instance. Although this configuration is unlikely to occur in an actual federated network, it may be an issue in a testing environment.

Confirm Sun Java Software for the SAML Affiliate Agent

The systems installed with the SAML Affiliate Agent need the Java Runtime Environment (JRE) installed. For the specific version, go to Technical Support and search for the SiteMinder Platform Matrix.

You can download a JRE from Sun Microsystems.

Set Up the SAML Affiliate Agent

To setup the SAML Affiliate Agent

1. Install a Web server and configure it to accept HTTPS and HTTP connections.
2. Install the SAML Affiliate Agent.
3. Modify the AffiliateConfig.xml file, in *saml_affiliate_agent_home*/config, as follows:
   a. Set the Enabled attribute to yes to enable the Agent.
   b. Specify the protected resources using the AffiliateResource attribute.
   c. You may also want to modify the following settings:
      - AffiliateName
      - AffiliatePassword (you provide this at installation time)
      - AssertionIssuer
      - CookieDomain
      - NoAccessURL
4. (Optional) If you are using a root certificate authority (CA) for SSL connections between the producer and the consumer, and this CA is not listed in the AM.keystore file, you need to add it. Use the Java keytool utility included with the JDK.

   Important! The SAML Affiliate Agent is *not* compatible with the FIPS 140-2 encryption standards.

   The SAML Affiliate Agent can ensure that the producer communicating with the consumer can present a certificate that has been verified by a trusted CA.

   Note: For information about using keytool, go to Sun Microsystems.
5. Test your system.

   Try accessing a protected resource on both http and https ports—you should be challenged. If not, check the SAML Affiliate Agent logs to troubleshoot the problem.

Chapter 3: Install a SAML Affiliate Agent

This section contains the following topics:

Introduction to SAML Affiliate Agent Installation

Instructions on installation or upgrade tasks are as follows:

- Installing/Uninstalling: Instructions are included in this chapter.
- Reinstalling/Reconfiguring: If you re-run the SAML Affiliate Agent installation program, you have two choices:
  - Reinstall—this option reinstalls files without changing the Agent's configuration.
  - Reconfigure—this option lets you go through the configuration again but reconfigures all selected Web servers the same way.
- Upgrading (All Platforms)

For a list of supported Web Servers and operating systems that work with SiteMinder products, go to Technical Support, and search for the SiteMinder Platform Matrix.

More Information

Upgrade to SAML Affiliate Agent v6.x QMR 6 (see page 121)

Before You Begin

For a list of supported Web Servers and operating systems that work with SiteMinder products, go to Technical Support, and search for the SiteMinder Platform Matrix.

More Information

Prepare the Installation (see page 23)

Select a Producer-side Web Server for Back-channel Connections

The Affiliate Server component of the SAML Affiliate Agent communicates with the producer on behalf of the Affiliate Web server plug-in using a back-channel SSL connection. The Affiliate server cannot communicate with an SSL-enabled Web server if that server is configured to accept client certificates. Therefore, the Web server at the producer must be configured to ignore client certificates.

Close Windows Applications Before Installing

Before running the SAML Affiliate Agent installation, close all applications, especially the Windows Services or Computer Management dialog boxes. These applications might cause errors with the installation and uninstallation of the SAML Affiliate Agent.

Choose an Installation Mode

The table that follows describes the installation methods available for the SAML Affiliate Agent.

The procedures in the table are for new installations.

Installation Mode	Description
GUI	Uses an installation wizard to install the SAML Affiliate Agent.
Console (UNIX only)	Uses command line questions about the installation in a UNIX console window.
Unattended	Installs the SAML Affiliate Agent without end-user interaction. Use the unattended installation mode to automate additional installations on other machines in your network. **Note:** You must install the SAML Affiliate Agent using GUI or Console mode *before* running an unattended installation. The initial installation creates a properties file that contains the settings for the unattended installation.

More Information

Install the SAML Affiliate Agent in GUI Mode (see page 29)
Install the SAML Affiliate Agent in Console Mode (UNIX Only) (see page 32)
Run the SAML Affiliate Agent Unattended Installation (see page 36)

Install the SAML Affiliate Agent in GUI Mode

The installation procedure assumes you are performing a new installation of the SAML Affiliate Agent. If the system is running a version previous to SAML Affiliate Agent 6.x QMR 6, upgrade to the 6.x QMR 6 version.

More Information

Upgrade to SAML Affiliate Agent v6.x QMR 6 (see page 121)

Run the GUI Installation

To install a SAML Affiliate Agent, you must be logged into the account under which the web server was installed.

Note the following:

- Running a GUI-mode installation using the Exceed application can truncate text in the dialogs because of unavailable fonts. This limitation has no affect on Agent installation and configuration.
- If you want to run a GUI installation on a UNIX system using telnet or other terminal emulation software, use an X-Windows session running in the background.

  Additionally, set the DISPLAY variable to your terminal, as follows:

  DISPLAY=111.11.1.12:0.0

  export DISPLAY

  111.11.1.12:0.0 is the XServer IP address and location.

  If you try to run in GUI mode on a UNIX system using a telnet window without an X-Windows session, the installer throws a Java exception and exits.

  If you prefer, you can run a command-line installation from a console window.
- The variable *saml_affiliate_agent_home* refers to the installed location of the SAML Affiliate Agent.

To install the SAML Affiliate Agent

1. Exit all applications that are running and stop the web server.
2. Download the software from the Technical Support site.

3. Complete one of the following steps:

   Windows: Navigate to the win32 folder then double-click nete-af-*version*-win32.exe.

   Solaris: From the solaris folder, copy nete-af-*version*-sol.bin to a local directory, navigate to that directory and enter the following in a console window:

   ./nete-af-*version*-sol.bin

   Linux: From the linux folder, copy nete-af-*version*-linux.bin (for Linux 2.1) or nete-af-*version*-rhel30.bin (for Linux 3.0) to a local directory, navigate to that directory and enter the following in a console window:

   ./nete-af-*version*-linux.bin (for Linux 2.1)

   ./nete-af-*version*-rhel30.bin (for Linux 3.0)

   The setup program prepares the installation files.

4. In the Introduction dialog, read the information then click Next.

5. Read the License Agreement and select the option to accept the agreement. Then, click Next.

   If you do not accept the agreement, the installation terminates.

6. Read the notes in the Important Information dialog, then click Next.

7. In the Choose Install Folder dialog, accept the default installation location or use the Browse button to select a different location. Click Next.

8. In the Web Server dialog, do one of the following:

   - Select two web server instances, one of which is the SSL web server.
   - Select one instance with two ports, one of which is the SSL port.

   The SAML artifact is sent across an SSL connection to the consumer.

   Note: IIS and Sun Java System 6.0 can use one web server for SSL and non-SSL connections.

9. If prompted, specify the location of the Java Runtime Environment (JRE) by accepting the default location or by using the Browse button to select a different location.

10. In the URL Information dialog, respond to the configuration prompts then click Next:

    Important! Enter a root URL in the form http://address.domain.com:port—do not enter any additional text.

    When you specify a value for a root URL, the installation script appends additional information to it in the AffiliateConfig.xml file. For example, if you enter https://interceptor.domain.com:90 for the SSL Interceptor Root URL, the script appends /smafa/amts/test1.htm to it.

    a. Affiliate Cookie Domain—enter the domain for the local server where the SAML Affiliate Agent is installed, such as .netegrity.com.

    b. SSL Interceptor Root URL—enter the URL at the consumer site where the producer redirects users during consumer requests. The URL is for the secure web server at the consumer, where the SAML Affiliate Agent is installed. We recommend that you use an SSL connection and being the URL with https://, such as

       https://mysslserver.example.com:90

       The SSLInterceptorURL enables the SAML Affiliate Agent to obtain the SAML artifact, which identifies the SAML assertion stored at the producer. The assertion contains user profile and session information. After the SAML Affiliate Agent gets the artifact, it calls the producer across the SSL back channel to retrieve the actual assertion.

       For all web servers, add the HTTPSPorts attribute to the AffiliateConfig.xml file and specify the same port number as you specify for the SSLInterceptorURL attribute. The HTTPSPorts attribute must be added to the GlobalInfo tag in the AffiliateConfig.xml file.

    c. Federation Web Services Root URL—enter the URL to the producer-side web server where the Web Agent Option Pack is installed. The URL must be a secure URL that begins in the form https://, such as

       https://myserver.ca.com:81

11. In the Passwords dialog, complete the following:

    a. Respond to the Shared Secret prompt by entering the secret twice. The SAML Affiliate Agent uses the shared secret to encrypt consumer cookies.

       The secret encrypts consumer cookies. You do not have to specify a corresponding secret in the Policy Server.

    b. Respond to the Affiliate Password prompt by entering the password twice. The SAML Affiliate Agent uses the password to communicate with the Policy Server at the producer site.

       This password must match the password for a consumer defined in the Policy Server User Interface.

       Note: For information about configuring a consumer, see the *Federation Security Services Guide*.

12. Optionally, respond to the prompt about optional UNIX configuration. If you are using the Bourne shell, you can include the nete-af-env.sh environment variable in the .profile file.
13. In the Pre-Installation Summary dialog, confirm the configuration settings then select Install.

    The setup program copies files to the specified location.
14. In the Install Complete dialog, select whether to restart your system now or later, then click Done to exit the installer.

    For Agents installed on IIS 6.0 Web Servers and Apache 1.x Web Servers, start the Affiliate Server before starting the web server.
15. Restart the web server.

More Information

Install the SAML Affiliate Agent in Console Mode (UNIX Only) (see page 32)
HTTPS Ports for SSL Connections (see page 69)
Check that the Affiliate Server Has Started (see page 47)

What To Do Next

1. Complete the tasks applicable for your Web server:
   - Configure an IIS Web Server for the SAML Affiliate Agent.
   - Configure a Sun Java System Web Server for the SAML Affiliate Agent.
   - Configure an Apache Web Server for the SAML Affiliate Agent.
2. After the specific Web server tasks are complete, verify that the SAML Affiliate Server has started.

More Information

How to Configure an IIS Web Server for the SAML Affiliate Agent (see page 39)
Configure a Sun Java System Web Server for the SAML Affiliate Agent (see page 43)
Configure an Apache Web Server for the SAML Affiliate Agent (see page 44)
Check that the Affiliate Server Has Started (see page 47)

Install the SAML Affiliate Agent in Console Mode (UNIX Only)

The following sections discuss installing the SAML Affiliate Agent in Console Mode (UNIX only).

Installation Notes

This procedure assumes that you are performing a new installation of the SAML Affiliate Agent. If the system is running a SAML Affiliate Agent previous to 6.x QMR 6, upgrade to the 6.x QMR 6 version.

Notes:

- When the installation program prompts with a question, the default entry is displayed in brackets **[]**. Press ENTER to accept the default.
- When you are prompted to enter a root URL, it must be in the form:

  http://address.domain.com:port or https://address.domain.com:port

  Important! Do not enter any additional text.

  When you specify a URL, the installation script appends additional information to that URL. For example, if you enter:

  http://interceptor.domain.com:90

  The script enters:

  http://interceptor.domain.com:90//smafa/amts/test1.htm

  in the AffiliateConfig.xml file.
- The UNIX installation script allows you to configure only one SAML Affiliate Agent per web server at a time. To install the SAML Affiliate Agent on more than one web server on a UNIX system, run the SAML Affiliate Agent binary (nete-af-*version-operating-system*.bin) and select the Reconfigure option.

More Information

Upgrade to SAML Affiliate Agent v6.x QMR 6 (see page 121)

Run the Installation in Console Mode

Before you install the SAML Affiliate Agent, you must be logged into the account where the Web server is installed, and have sufficient permissions to run and modify this Web server.

To install the SAML Affiliate Agent on UNIX systems

1. From the SAML Affiliate Agent installation media, copy the installation binary for your operating system to a local directory:
   - Solaris: Navigate to the solaris folder and copy nete-af-*version*-sol.bin
   - Linux: Navigate to the linux folder and copy nete-af-*version*-linux.bin (for Linux 2.1) or nete-af-*version*-rhel30.bin
2. At a prompt, enter the following command:

   ./nete-af-*version-operating_system*.bin -i console

   where *operating_system* is sol for Solaris systems, linux for Linux 2.1 systems, or rhel30 for Linux 3.0 systems.

   The installation script prepares the License Agreement file.
3. Press ENTER to read the License Agreement.

   Press ENTER to page through the agreement.
4. Enter Y to accept the license agreement.
5. Press ENTER to read the Release Notes.

   Press ENTER to page through the notes.
6. Enter Y to continue with the installation.
7. Specify the directory where the installation should place the Agent files. To accept the default location, press ENTER.
8. Choose the type of Web server to configure for the SAML Affiliate Agent by entering 1 for Sun Java System Web servers or 2 for Apache Web servers.
9. Enter the Web server's root directory, then press ENTER—for example:
   - For Sun Java System: /export/smuser/sunone/servers
   - For Apache: /usr/local/apache2
10. For Sun Java System Web Servers, select the Web server instance to configure for the SAML Affiliate Agent.
11. Enter the location of an installed Java Runtime Environment (JRE).
12. Enter the Cookie Domain for the consumer, such as .consumer.com.

13. Enter the SSL Interceptor Root URL, which is the URL at the consumer site where the producer redirects users during consumer requests. This is a URL to the consumer's secure Web server where the SAML Affiliate Agent is installed. We recommend that you use an SSL connection, and that the URL begin with https://, such as

    https://mysslserver.example.com:90

    The SSLInterceptorURL enables the SAML Affiliate Agent to obtain the SAML artifact, which identifies the SAML assertion stored at the producer. The assertion contains user profile and session information. After the SAML Affiliate Agent gets the artifact, it makes a call on the SSL back channel to the producer to retrieve the actual assertion.

    For all web servers, you must add the HTTPSPorts attribute to the AffiliateConfig.xml file and specify the same port number as you specify for the SSLInterceptorURL attribute. The HTTPSPorts attribute must be added to the GlobalInfo tag in the AffiliateConfig.xml file.

14. Enter the Federation Web Services Root URL, which is the URL to the producer Web server where the SiteMinder Option Pack for the Web Agent is installed. This must be a secure URL that begins https://, such as

    https://myserver.ca.com:81

15. Respond to the Shared Secret prompt by entering the secret that the SAML Affiliate Agent will use to encrypt consumer cookies. Re-enter the secret when prompted.

    This secret is used locally to encrypt consumer cookies. You do not have to specify a corresponding secret in the Policy Server.

16. Respond to the Affiliate Password prompt by entering the password that the SAML Affiliate Agent will use to communicate with the Policy Server at the producer site.

    This password must match the password for a consumer defined in the Policy Server User Interface.

    Note: For information on configuring a consumer, see the *SiteMinder Federation Security Services Guide*.

17. Confirm the installation settings by entering Y.

    The installation script copies the files to the specified installation directory.

18. For UNIX installations, respond to the prompt about UNIX configuration. If you are using the Bourne shell, include the nete-af-env.sh environment variable in the .profile file.

19. If applicable, make note of the final instruction (if you did not log in as root) to update the /etc/rc2.d directory.

    Cannot write to /etc/rc2.d/ directory. To complete daemon setup, copy

    '/export/smuser/netegrity/affiliateagent/bin/S98smaffserver'

    to '/etc/rc2.d/'

20. Restart the Web server for the changes to take effect.

More Information

HTTPS Ports for SSL Connections (see page 69)

What to Do After Installing the SAML Affiliate Agent

After you install the Agent, do the following:

1. Complete the tasks for your Web server:
   - When you install the SAML Affiliate Agent on a Sun Java System Web server, configuration settings are added to the magnus.conf file, and the obj.conf file, files which are loaded automatically when the Web server is started. These added settings are used to initialize the SAML Affiliate Agent.
   - Configure the Apache Web Server for the SAML Affiliate Agent.
2. After the specific Web server tasks are complete, verify that the SAML Affiliate Server has started.

More Information

Configure an Apache Web Server for the SAML Affiliate Agent (see page 44)
Check that the Affiliate Server Has Started (see page 47)

Run the SAML Affiliate Agent Unattended Installation

After you have installed the SAML Affiliate Agent on one system, you can automate installations on other Web servers using the unattended installation feature. An unattended installation lets you install or uninstall the SAML Affiliate Agent without any user interaction.

Prepare an Unattended Installation

Unattended installation uses the nete-af-installer.properties file to propagate the SAML Affiliate Agent installation set up across all Agents in your network. In this properties file, you define installation parameters, then copy the file and the Affiliate Agent executable file to any Web server in your network to run an unattended installation.

The nete-af-installer.properties file is installed in the following location:

saml_affiliate_agent_home/install_config_info/affiliateagent/
nete-af-installer.properties

The default parameters, paths, and passwords in the file reflect the information you entered during the initial SAML Affiliate Agent installation.

To install the nete-af-installer.properties file

1. Run an initial installation of the SAML Affiliate Agent.
2. Open the nete-af-installer.properties file and modify parameters as needed.
3. Save the file.

More Information

Properties File for Unattended Installations (see page 149)
Install the SAML Affiliate Agent in Console Mode (UNIX Only) (see page 32)
Install the SAML Affiliate Agent in GUI Mode (see page 29)

Run an Unattended Installation

You should have completed an initial SAML Affiliate Agent installation and, if necessary, modified the nete-af-installer.properties file. Now, you can use the file to run subsequent SAML Affiliate Agent installations.

To run an unattended installation

1. From a system where the SAML Affiliate Agent is already installed, copy the following files to a local directory.
   a. nete-af-*version*-win32.exe (Windows executable) or nete-af-*version-operating_system*.bin (UNIX binary) from the SiteMinder DVD or from where it resides on your system
   b. nete-af-installer.properties file from *affiliate_agent_home*
2. Place these files on your local system.
3. Open a console window and navigate to the directory where you copied the files.

4. Run the Agent executable with the -f and -i silent options, as follows:

   agent_executable -f *properties_file* -i silent

   Assuming that you run the installation from the directory where the executable and properties file are located, the command would be:

   Windows:

   nete-af-*version*-win32.exe -f nete-af-installer.properties - i silent

   Solaris:

   ./nete-af-*version*-sol.bin -f nete-af-installer.properties -i silent

   Linux 2.1:

   ./nete-af-*version*-linux.bin -f nete-af-installer.properties -i silent

   Linux 3.0:

   ./nete-af-*version*-rhel30.bin -f nete-af-installer.properties -i silent

   Note: If you are not running the command from the directory with the two files, specify the full path to each file. For Windows, if there are spaces in the directory paths, enclose the entire path between quotation marks.

   When the installation is complete, you return to the command prompt.

5. Check to see if the installation completed successfully by looking in the CA_SiteMinder_SAML_Affiliate_Agent _v*version*_InstallLog.log file, located in the *saml_affiliate_agent_home*/install_config_info directory. This log file contains the results of the installation.

Stop an Unattended Installation in Progress

To manually stop the installation

- Windows: Open the Windows Task Manager and stop the nete-af-*version*-win32.exe and affl_minder.exe processes.
- UNIX: Press Cntrl + C.

Note: You can run an unattended installation to reinstall the SAML Affiliate Agent on the same system where you initially performed an installation in GUI or console mode.

What To Do After Running the Unattended Installation

After running an unattended installation, do the following:

1. Complete the tasks for your Web server:
   - Configure an IIS Web Server for the SAML Affiliate Agent.
   - Configure a Sun Java System Web Server for the SAML Affiliate Agent.
   - Configure an Apache Web Server for the SAML Affiliate Agent.
2. After reviewing the additional information, verify that the SAML Affiliate Server has started.

More Information

How to Configure an IIS Web Server for the SAML Affiliate Agent (see page 39)
Configure a Sun Java System Web Server for the SAML Affiliate Agent (see page 43)
Configure an Apache Web Server for the SAML Affiliate Agent (see page 44)
Check that the Affiliate Server Has Started (see page 47)

Configure the SAML Affiliate Agent for Your Web Server

The following sections explain how to configure the SAML Affiliate Agent for each supported web server.

How to Configure an IIS Web Server for the SAML Affiliate Agent

If you installed the SAML Affiliate Agent on an IIS Web server, you need to configure the IIS Web Server using the following process:

1. Change the user account directory security settings.
2. Assign operating system permissions to a user account.
3. Allow IIS to execute ISAPI extensions.
4. If you are not using the default IIS web site (or need to restore it), edit the metabase.

Modify the Directory Security Settings for the Default IIS User Account

IIS does not allow any plug-in to write to a file system unless it is configured with a user account that has proper rights to do so. Therefore, to enable the SAML Affiliate Agent to create a log file on the local system, you need to modify the directory security settings for the IIS default user account.

To modify the directory security settings for the default user account

1. Open the IIS Internet Services Manager.
2. Select the Web server where the SAML Affiliate Agent is installed and right-click Properties.
3. In the Properties dialog box, click Edit to display the master properties for the WWW services.
4. In the WWW Service Master Properties dialog box, select the Directory Security tab.
5. In the group box that begins with "Anonymous access"..., click Edit.
6. In the Authentication Methods dialog box, do the following:
   a. Select Anonymous Access.
   b. Deselect Basic authentication.
   c. Deselect Integrated Windows authentication.
   d. In the Anonymous access group box, click Edit.
   e. In the Anonymous User Account dialog box, enter a name and password of a user account for access by anonymous users. Also, deselect Allow IIS to control password, then click OK.

      This account should have the "right to act as part of the operating system." Refer to Windows documentation to grant this right to a user account.
   f. Exit all dialog boxes and the Internet Services Manager.
7. When prompted, apply the security changes to all child components of the Web server.
8. Restart the Web server.

   The directory security permissions are changed.

Assign Operating System Permission to the Default IIS User Account

The default user account of the IIS Web server needs permissions to act as part of the operating system.

To assign operating system permission

1. Open the Control Panel and open Administrative Tools, Local Security Policy, Local Policies, User Rights Assignment.
2. Select Act as part of the operating system.
3. Add users to the Local Security Policy Setting dialog box, then click OK.
4. Exit from the control panel.

   The permissions are assigned.

Allow IIS to Execute the SAML Affiliate Agent ISAPI Extension

You must add an ISAPI extention to the IIS 6.0 Web server and grant the server permission to execute it before configuring the SAML Affiliate Agent. This extension will execute the SAML Affiliate Agent ISAPI.

To add the extension and permission

1. Open the Internet Information Services (IIS) Manager, and then expand the web server you are configuring for the Agent.
2. Double-click Web Service Extensions

   The Web Service extensions pane appears.
3. Click the Add a new Web service extension link.

   The New Web Service Extension dialog box opens.
4. In the Extension name field, enter ISAPI60AffiliateAgentDLL, and then click Add.

   The Add File dialog box opens.
5. Click the Browse button, and then navigate to the ISAPI60AffiliateAgent.dll file in the *web_agent_home*/bin/plugins directory. If the proper file does not appear, click the Files of type drop-down list and select ISAPI dll files.
6. Click Open

   The path to the file appears in the Add File dialog box.
7. Click OK.

   You return to the New Web Service Extension dialog box.

8. Select the Set extension status to allowed check box.
9. Click OK.

   The New Web Service Extension dialog box closes, and the server has permission to execute the extension.

How to Use a Non-Default IIS Website

SiteMinder requires the default IIS web site for proper installation. By default, this site exists when you install an IIS web server. If any of the following conditions exist, edit the Metabase before configuring a SiteMinder IIS Web Agent :

- If the default IIS website no longer exists.
- If the default IIS website has been renamed.
- If you want to install the SiteMinder virtual directories on a different (non-default) IIS website.

The actual tools and steps involved in editing the Metabase depend on the version of IIS you are using. For example, if you are using an r6.0 SP6 SiteMinder Web Agent, on IIS 6.0, you would edit the Metabase using the following process:

Note: For more information, see your Microsoft documentation, or go to http://support.microsoft.com/

1. Download and install the Metabase Explorer from Microsoft by doing the following:
   a. Go to the Microsoft Downloads website.
   b. Search for "IIS 6.0 Resource Kit Tools," which includes the Metabase Explorer.
   c. Download and install the tools.
   d. Create a backup copy of your metabase.xml file.
2. On your IIS web server, open the IIS Manager. Find the website on which you want to install the SiteMinder Web Agent, and note its identifier (number) for future reference.
3. Close the IIS Manager, and open the Metabase Explorer.
4. Expand the following key:

   ```
   LM\W3SVC\
   ```

5. Expand the key that corresponds to the identifier from Step 2.

   A list of sub keys appears.

6. Right-click the key from Step 5, select Rename, and then change the value of the key to 1.
7. From the list of sub keys in the left pane, expand the following key:

   `root`

   A list of keys appears in the right pane of the Metabase Explorer.
8. Double-click the following key:

   `AppRoot`

   The AppRoot Properties dialog appears. The Value Data field shows the following string:

   `/LM/W3SVC/identifier_number/Root`
9. Change the value of the *identifier_number* to 1, and then click OK.
10. Close the Metabase Explorer.
11. Run the Configuration Wizard to reconfigure your IIS Web Agent.
12. Repeat Steps 3 through 10, but change the number 1 back to the original identifier from in Step 2.
13. Restart the IIS web server.

Configure a Sun Java System Web Server for the SAML Affiliate Agent

When you install the SAML Affiliate Agent on a Sun Java System (iPlanet/Sun ONE) Web server, configuration settings are added to the magnus.conf file (Sun Java System 6.0 only), and the obj.conf file, files which are loaded automatically when the Web server is started. These added settings are used to initialize the SAML Affiliate Agent.

If you plan to use the Sun Java System Administration console, you must apply these changes before making any modifications with the console. Otherwise, the SAML Affiliate Agent configuration may be lost. If you lose your configuration, rerun the configuration script.

To apply changes to the Sun Java System files

1. Log on to the Sun Java System Administration Server console.
2. Select the server you want to manage.
3. Apply manual changes then load the new configuration settings.
4. Restart the Web server.

Note: You do not have to modify any of these files. This information in the appendix is only for reference.

More Information

Modifications to Sun Java System Files During Agent Installation (see page 165)

Configure an Apache Web Server for the SAML Affiliate Agent

After you install the Agent, modify the httpd.conf configuration file to enable the Apache web server to operate with the SAML Affiliate Agent.

Some of the files in the following procedure use the Solaris file extension (.so) in the file names. The extension can differ depending on your operating system.

To modify the httpd.conf file

1. Stop the web server.
2. Navigate to the conf directory: /*Apache_home*/conf

   Example: $ cd /usr/apache/conf
3. Open the httpd.conf file.
4. Do one of the following:

   For Apache 2.x, add the following lines to the DSO configuration section:

   - LoadModule smAffiliate_module *plugins_directory*/libApache20AffiliateAgent.so

     where *plugins_directory* is the full path to the SAML Affiliate Agent plugins directory.
   - SmAffiliateInitFile *saml_affiliate_agent_home*/config/AffiliateConfig.xml

     Place this entry after the LoadModule entry. Do not use a relative path for this location.

The modified sections of the file must resemble the sample text below for Apache 2.0.43:

```
# Dynamic Shared Object (DSO) Support
#
# To be able to use the functionality of a module which
# was built as a DSO you have to place corresponding
#`LoadModule' lines at this #location so the
# directives contained in it are actually available
# _before_ they are used. Statically compiled modules
# (those listed by `httpd -l') do not need to be loaded
# here.
#
# Example:
# LoadModule foo_module modules/mod_foo.so
#
LoadModule smAffiliate_module
/export/aa259/affiliateagent/bin/plugins/libApache20AffiliateAgent.so
SmAffiliateInitFile
/export/netegrity/affiliateagent/config/AffiliateConfig.xml
```

For Apache 1.x, add the following lines to the DSO configuration section:

- LoadModule smAffiliate_module *plugins_directory*/libApacheAffiliateAgent.so

  where *plugins_directory* is the full path to the SAML Affiliate Agent plugins directory.

- SmAffiliateInitFile *saml_affiliate_agent_home*/config/AffiliateConfig.xml

  Place this entry after the LoadModule entry. Do not use a relative path for this location.

5. Save the modified httpd.conf file.
6. If you have not already done so, enable the SAML Affiliate Agent.
   a. Open the AffiliateConfig.xml file, located as follows:

      saml_affiliate_agent_home/config

   b. In the PerAffiliateInfo section, set the Enabled attribute to **yes**.
7. Restart the web server and run the nete-af-env.sh script.

   Note: Any time you have to restart the Apache Web server with the SAML Affiliate Agent, run the nete-af-env.sh script to set required environment variables.
8. Compile the Apache web server, as instructed in the next section.

Compile Apache on UNIX Platforms

For the Apache Web server to work with the SAML Affiliate Agent, you need to compile it with the appropriate settings. This applies to any UNIX platform where the Apache Web server is installed.

To compile the Apache Web server

1. Enter the following commands in a console window:

   LIBS=-lpthread

   export LIBS

2. Enter the next series of commands as follows:

   configure --enable-module=so --prefix=*install_target_dir*

   make

   make install

3. (Optional) Set the LD_PRELOAD environment variable, as instructed in the next section.

Set LD_PRELOAD for Apache/Linux SAML Affiliate Agents

For the SAML Affiliate Agent v6.x on an Apache server running Linux, you must set the LD_PRELOAD environment variable as follows:

LD_PRELOAD=*saml_affiliate_agent_home*/bin/plugins/libbtunicode.so

Setting the LD_PRELOAD environment variable binds the libbtunicode.so library to all applications executed by the user account for which it is set.

Important! Reset the LD_PRELOAD variable to its original value before you uninstall the SAML Affiliate Agent.

If this variable is not set, the following problems may occur:

- You may encounter a core file generated when the Affiliate Agent shuts down. You can avoid the core file generation by configuring the LD_PRELOAD environment variable.
- When a SAML Affiliate Agent is running on an Apache Web Server, a load change may spawn multiple child processes that eventually consume 100% of the CPU cycles.

Check that the Affiliate Server Has Started

You need to ensure that the Affiliate Server has started after you completed the SAML Affiliate Agent installation. Review the appropriate section that follows to check that the server has started.

Note: For SAML Affiliate Agents installed on IIS 6.0 or Apache 1.x Web servers, the Affiliate Server must be running before you start the IIS 6.0 Web server.

Check the Affiliate Server on Windows Systems

To check the affiliate server on Windows

1. Open the Services control panel.
2. Check the Affiliate Minder Service and make sure the service status is started. If not, right-click it and select Start.
3. Exit the Services Control Panel.
4. After the Affiliate Server is running, you will need to perform initial agent configuration.

More Information

Perform the Initial Agent Configuration (see page 48)

Check the Affiliate Server on UNIX Systems

To check the affiliate server on UNIX

1. From a command window, enter the command ps -ef | grep sma and check that the smaffdaemon is listed.

   If not, go to the SAML Affiliate Agent's bin directory, such as affiliateagent/bin and enter ./start-all-am.
2. After the Affiliate Server is running, you will need to perform initial agent configuration.

More Information

Perform the Initial Agent Configuration (see page 48)

Perform the Initial Agent Configuration

Performing the initial configuration on Windows involves:

- Editing the AffiliateConfig.xml file
- Checking for the Affiliate Server log file

Edit the AffiliateConfig.xml File

The SAML Affiliate Agent Web server plug-in is configured through the AffiliateConfig.xml file. The initial configuration of this file ensures that the file reflects the correct settings to establish federated communication. All of these settings are in the PerAffiliateInfo section.

The AffiliateConfig.xml file resides in:

Windows: *saml_affiliate_agent_home*\config\

UNIX: *saml_affiliate_agent_home*/config/

The file defines elements and attributes in three sections—GlobalInfo, settings that apply for the consumer and producer, PerPortalInfo, settings that apply only to the producer, and PerAffiliateInfo, settings that apply only to the consumer.

Note: Restart the Web server after modifying the AffiliateConfig.xml file.

- AffiliateName

  Enter the name of the consumer as determined by the producer. For example:

  <AffiliateName>SampleConsumer</AffiliateName>

- AffiliatePassword

  The AffiliatePassword defines the encrypted password shared between the consumer and the producer. The password is used by the consumer to identify itself to the producer.

  <AffiliatePassword>cZQwIBUYl2xkn7IR4vkI2931bPuosh9U</AffiliatePassword>

  Important! The SAML Affiliate Agent is *not* compatible with the FIPS 140-2 encryption standards.

- AssertionAudience

  Enter the location of the document that describes the terms and conditions of the business agreement between the producer and the consumer. The audience is determined by the administrator at the producer.

  <AssertionAudience>http://www.netegrity.com/SampleAudience</AssertionAudience>

- AssertionIssuer

  Enter the URL that issues assertions for specific consumers. The consumer can accept assertions from only this issuer. For example,

  <AssertionIssuer>http://www.netegrity.com/SiteMinder</AssertionIssuer>

- AffiliateResource

  Enter a value against which the SAML Affiliate Agent compares each requested URL to determine if the consumer resource is protected. For example,

  <AffiliateResource MatchingRule="StrictPrefix">protected</AffiliaterResource>

- CompanySourceID

  Enter the producer's source ID. This ID must match the source ID that the producer uses. For example,

  <CompanySourceID>b818452610a0ea431bff69dd346aeeff83128b6a</CompanySourceID>

- PortalName

  Enter the name of the producer site associated with the consumer. For example,

  <PortalName>SamplePortal</PortalName>

  Note: The name must match the name you enter for the portalName attribute in the PerPortalInfo section of the AffiliateConfig.xml file.

Check for the Affiliate Server Log File

To ensure that the Affiliate Server's log file has been created by the installation, go to *agent_install_directory*/log and look for the affiliateserver.txt file. For example, on UNIX, /export/smuser/affiliateagent/log/affiliateserver.txt

Reconfigure the SAML Affiliate Agent

When you first installed the SAML Affiliate Agent, you were prompted to select a Web server, then to specify a consumer cookie domain and several URLs for federated communication.

You can edit the AffiliateConfig.xml file or re-run the configuration script, as documented in the following procedures.

The instructions that follow reflect the GUI mode procedures. For UNIX systems, you can edit the AffiliateConfig.xml file using console mode by executing the SAML Affiliate Agent binary file (nete-af-*version-operating_system*.bin) with the -i console command argument. The command-line upgrade prompts will be similar to GUI mode prompts.

Note: If you edit the AffiliateConfig.xml file, remember to save the file and restart the Web server.

To reconfigure these Web server settings

1. Exit all applications that are running and stop the Web server.
2. Re-run the SAML Affiliate Agent installation program:

   Windows: nete-af-*version*-win32.exe

   Solaris: nete-af-*version*-sol.bin

   Linux 2.1: nete-af-*version*-linux.bin

   Linux 3.0: nete-af-*version*-rhel30.bin
3. In the Reinstall dialog box, select Reconfigure, then click Next.
4. Go through the installation program and modify any settings.
5. After completing the installation, restart the Web server.

Reinstall the SAML Affiliate Agent

Install the SAML Affiliate Agent over an existing SAML Affiliate Agent of the same version to restore lost application files.

The reinstallation instructions that follow reflect the GUI mode procedures. For UNIX systems, you can reinstall using the console mode by executing the SAML Affiliate Agent binary file (nete-af-*version-operating_system*.bin) with the **-i console** command argument. The command-line upgrade prompts are similar to GUI mode prompts.

To reinstall the SAML Affiliate Agent

1. Exit all applications that are running and stop the web server.
2. Re-run the SAML Affiliate Agent installation program:

   Windows: nete-af-*version*-win32.exe

   Solaris: nete-af-*version*-sol.bin

   Linux 2.1: nete-af-*version*-linux.bin

   Linux 3.0: nete-af-*version*-rhel30.bin

3. In the Reinstall dialog box, select Reinstall or Reconfigure, then click Next.

   For UNIX only: If the SAML Affiliate Agent installation directory exists, the program gives you the option of aborting the installation, specifying a new installation directory, or overwriting the existing directory.

4. In the Pre-Installation Summary dialog, review the settings, then click Install.
5. In the Install Complete dialog, decide whether to restart your system now or later, then click Done to exit the installer.
6. Restart your web server after the installation is complete.

Uninstall the SAML Affiliate Agent

Before you uninstall the Affiliate Agent, you may want to make copies of your registry settings and Agent configuration settings, including nete-af-installer.properties, to have as a back up.

After you complete the uninstallation of the SAML Affiliate Agent from your system, an erroneous error message might appear:

"Uninstallation unable to proceed;"

This message does not signify an actual failure to uninstall the Agent. It occurs only if the SAML Affiliate Agent has been reinstalled at least once prior to uninstallation.

Uninstall the SAML Affiliate Agent From Windows Systems

Before you uninstall the Agent:

- You may want to make copies of your Agent configuration settings to have as a back up.
- Specify the JRE in the PATH environment to avoid receiving a message that the Java virtual machine could not be found.

To uninstall the SAML Affiliate Agent

1. Open the Services Control Panel.
2. Stop the Web server.
3. Stop the Affiliate Server by right-clicking the Affiliate Minder Service and selecting Stop.
4. Close the Services Control Panel.
5. Return to the main Control Panel and double click Add/Remove Programs.

6. Select CA SiteMinder SAML Affiliate Agent QMR6 and click Change/Remove.

   An InstallAnywhere dialog box is displayed, prompting you to remove the software.

7. Click Uninstall.

   The installation program removes the files.

8. In the SAML Affiliate Agent Uninstall dialog box, click Done.

9. When the uninstallation is finished, exit the Control Panel.

10. Restart the Web server.

Uninstall the SAML Affiliate Agent From UNIX Systems

Before you remove the SAML Affiliate Agent:

- You may want to make copies of your Agent configuration settings to have as a back up.
- For Agents installed on an Apache/Linux Web server, reset the LD_PRELOAD variable before you uninstall the SAML Affiliate Agent.
- Specify the JRE in the PATH environment to avoid receiving a message that the Java virtual machine could not be found. Add the JRE to the PATH variable as follows:

  ```
  PATH=${PATH}:/jre_home/bin
  export PATH
  ```

  jre_home is where JRE resides

To uninstall the SAML Affiliate Agent

1. Log onto the UNIX system.
2. Stop the Web server(s).
3. Stop the Affiliate Server by navigating to the SAML Affiliate Agent's bin directory, such as affiliateagent/bin, and entering ./stop-all-am
4. Do one of the following:
   - Navigate to *saml_affiliate_agent_home*/install_config_info/nete-af-uninstall, then enter the following command:

     ./uninstall

   - Navigate to *saml_affiliate_agent_home* and enter the following command:

     nete-af-uninstall.sh

   The uninstallation program starts.

5. In the SAML Affiliate Agent *version* Uninstall window, click Uninstall.

   Uninstallation begins and the SAML Affiliate Agent is removed from the system.

   For Sun Java System Web servers, the obj.conf file(s) and magnus.conf files are restored to its original settings prior to the Web Agent installation.

   Note: If you are uninstalling an SAML Affiliate Agent from an Apache Web server, remove the lines from the httpd.conf file that you added during the installation.

6. Change to your home directory (the current directory may be deleted).
7. Restart the Web server(s).

Note: If you try to install the SAML Affiliate Agent again, and a SAML Affiliate Agent installation directory exists, the program gives you the option of aborting the installation, specifying a new installation directory, or overwriting the existing directory.

Chapter 4: Configure a SAML Affiliate Agent

This section contains the following topics:

Configure the Affiliate Server

The following sections discuss configuring the Affiliate Server.

Guidelines for Modifying the Affiliateserverconf.properties File

Use the following guidelines when modifying the affiliateserverconf.properties file:

- Use a text editor.
- Do not add extra spaces between the parameter name, the equals sign (=), and the parameter value.
- For configuration changes to take effect, save the affiliateserverconf.properties file and restart the Affiliate Server.

To view the affiliateserverconf.properties file, go to:

saml_affiliate_agent_home/config

More Information

Restart the Affiliate Server (see page 59)

Affiliate Server Communication with the Web Server Plug-in

The serverport parameter defines the server port on which the Affiliate Server listens to the Affiliate Web Server plug-in.

The port number is 4444; do not change this port number.

Specify the Number of Threads for Requests

The threadpoolsize setting determines how many threads the Affiliate Server has available to process requests from the web server plug-in to contact the producer. The default number of threads is 10.

Important! We recommend that you do not modify threadpoolsize unless directed to do so by Technical Support.

Once you are directed by technical support, you can make the following changes to the threadpoolsize:

- To decrease the number of threads, set the parameter to a lower value, for example, threadpoolsize=7.
- To increase the number of threads do one of the following:
  - set the parameter to a higher value, for example, threadpoolsize=20
  - increase the number of available threads by enabling the flexiblethreadpool parameter. By setting this parameter to yes (flexiblethreadpool=yes), the Affiliate Server creates one additional thread for a new request if all 10 default threads are in use. If the newly added thread is in use as subsequent requests come in, other threads are added one at a time.

    The default setting for the flexiblethreadpool parameter is no for all Web servers except IIS 6.0 and Apache 1.x. For IIS 6.0 and Apache 1.x, the default is yes. If you accept the default, and all 10 threads are in use, the Affiliate Server places requests in a queue until a thread becomes available.

Configure Logging

You can configure the Affiliate Server to write messages to a log file.

Record Messages in a Log File

To write error messages to a log file, set the logfilerequired setting to yes and enter a file name in the logfilename directive. For example:

logfilerequired=yes
logfilename=/opt/netegrity/affiliateagent/log/affiliateserver.txt

Note: This parameter is *not* supported for SAML Affiliate Agents configured with IIS 6.0 and Apache 1.x web servers. When used with these web servers, the Affiliate Server and the SAML Affiliate Agent both use a single log file. The logfilename directive specifies the location of the log file.

Specify Log Levels

You can configure the Affiliate Server to generate different levels of log messages. Choosing a log level can facilitate troubleshooting by dictating the severity and extent of the logged messages.

The following is an example of a log message:

Feb 25, 2009 10:41:52 AM[6631688:smmtserver] Server Started listening for requests

The following table lists the log levels that you can select:

Log Level	Type of Message
0	No error messages are logged.
1	Error messages only. For example, the inability of the Affiliate Server to communicate with the producer.
2	Error and informational messages. These messages provide information about processes without a lot of detail.
3	All messages, including those of the least importance, such as header details and cookie variables, as well as trace messages, flow state messages, all with the highest level of detail. This is the default log level.

For the chosen log level, the Affiliate Server prints messages for that level and messages from any lower level. For example, if you choose level 3, all messages from levels 0 through 3 are printed.

To specify the log level

1. Set the loglevel attribute to a value from 0 through 3.

   For example: loglevel=2

2. If you are using any of the following web servers, you must restart the web server to apply the changes:

   - IIS 5.0
   - Apache 1.x, 2.x (*except* those on UNIX platforms)
   - Sun Java System (UNIX platforms only)

Append Messages to Log Files

To add messages to an existing log file instead of rewriting the entire file each time logging is invoked, enable the logappend setting, as follows:

logappend=yes

The default is no.

Note: This parameter is *not* supported for SAML Affiliate Agents configured with IIS 6.0 and Apache 1.x web servers. When used with these web servers, the Affiliate Server and the SAML Affiliate Agent both use a single log file. The logfilename directive specifies the location of the log file.

Determine the Date Format in Log Files

You can specify how the date and time is displayed in the affiliate server log by entering setting the dateformatpattern.

The default format is: MMMM d, yyyy H:mm:ss.S a

Format	Description
MMMM	month in alphabetic form, such as June
d	date in numeric form, such as 19
yyyy	year
H:mm:ss	hour:minute:seconds
S	milliseconds
a	AM or PM

You can use any combination of characters appropriate for java.text.DateFormat (see Java documentation); you are not required to use any of the default characters.

Determine SSL Cache Time

The sslcachetime setting specifies how long the Affiliate Server saves SSL connection information that it receives from the producer-side Web server where Federation Web Services is installed. The default value for this parameter is 360000 milliseconds.

The consumer-side client reestablishes the SSL connection after the cache time expires.

To change this setting, enter a new value, in milliseconds, for example:

sslcachetime=460000

Setting this value to a negative integer is equivalent to infinite cache time.

Note: Setting this value higher improves performance of the SSL back channel connection but may decrease the level of security of the connection because of less frequent SSL handshaking between the SAML Affiliate Agent and the Federation Web Services.

Designate Message Factories

The msgfactories parameter defines the Java classes that contain messages sent by the web server plug-in to the Affiliate Server.

Do not change the value of this parameter.

Close the Artifact Back Channel After Assertion Retrieval

For artifact single sign-on, the Affiliate Server retrieves the assertion from the producer across an SSL back channel. You need to ensure that the Affiliate Server closes or times out the socket connection for the back channel after the assertion is retrieved. If the back channel connection is not closed, all the connection resources may be used up.

The sockettimeout attribute lets you set a specific timeout value for the SSL back channel connection between the producer and the Affiliate Server.

The default value for the sockettimeout is 300000 milliseconds (5 minutes). The maximum value allowed is 2,147,483,647 milliseconds (597 hours).

Restart the Affiliate Server

If you make any changes to the affiliateserverconf.properties file, you need to restart the Affiliate Server.

Go to the appropriate section to restart the Affiliate Server.

Restart the Affiliate Server on Windows Systems

To restart the affiliate server on Windows

1. Open the Component Services control panel.
2. Right-click the Affiliate Minder Service and select Stop.

   Windows stops the service.
3. Right-click the Affiliate Minder Service again and select Start.
4. Exit the Component Services dialog box.

Restart the Affiliate Server on UNIX Systems

To restart the affiliate server on UNIX

1. From a command window, go to the SAML Affiliate Agent's bin directory, such as affiliateagent/bin.
2. Enter ./stop-all-am

   The server will stop.
3. Enter ./start-all-am to restart the server.

Configuring the Affiliate Web Server Plug-in

The AffiliateConfig.xml file is organized into the following sections:

- Global settings—define overall Web server settings where the SAML Affiliate Agent is installed.
- Portal settings—identify one or more producers and specify the location of specific services at each site. These settings apply on a per-producer basis; multiple producer sections can exist in one file.
- Affiliate settings—enable SAML Affiliate Agent operation, specify how the consumer communicates with the producer, and determine how users are managed.

Note: Multiple consumer sites can exist in one configuration file by adding a new PerAffiliateInfo section for each consumer.

The AffiliateConfig.xml file resides in:

- Windows: *saml_affiliate_agent_home*\config
- UNIX: *saml_affiliate_agent_home*/config

More Information

Configure Global Settings (see page 67)
Configure Portal Settings (see page 89)
Configure Affiliate Settings (see page 95)

Guidelines for Modifying the AffiliateConfig.xml File

Use the following guidelines when you modify the AffiliateConfig.xml file:

- Use an XML editor.
- For configuration changes to take effect, save the AffiliateConfig.xml file and restart the web server.
- There must be a one-to-one correspondence between consumers and producers in the AffiliateConfig.xml file, regardless of whether a single consumer is associated with multiple producers, or one configuration file is being shared by multiple consumers. You must have one PerAffiliateInfo section associated with one PerPortalInfo section.
- Do not add extra spaces between the attribute name, the equals sign (=), and the attribute value.
- In XML, attribute values must always be enclosed in quotation marks, for example, LogToConsole="yes".
- The SharedSecret and the AffiliatePassword must be set by the EncryptKey utility; they cannot be changed in the file directly.

Requirement for AffiliateConfig.xml on Solaris Platforms

For Solaris systems that have Web servers installed with the SAML Affiliate Agent, the <DOCTYPE> element in the AffiliateConfig.xml file must be enclosed in comments or the Agent fails to operate.

The format to comment text out is:

```
<!--
commented text
-->
```

The beginning of the Affiliateconfig.xml file should appear as follows:

```
<?xml version="1.0"?>
<!--
<!DOCTYPE AffiliateConfig SYSTEM "AffiliateConfig.dtd">
-->
```

In general, the AffiliateConfig.xml file must use well-formed XML to operate.

Modify the SAML Affiliate Agent Configuration

To modify the SAML Affiliate Agent's configuration, do one of the following:

- Run the installation program and select the Reconfigure option. This option lets you go through the configuration again but reconfigures all selected Web servers the same way.
- Edit the AffiliateConfig.xml file directly. If you edit the AffiliateConfig.xml file directly, remember to save the file and restart the Web server.
- Run the Configuration Wizard.

More Information

Modify Individual Server Configurations (see page 62)

Modify Individual Server Configurations

The SAML Affiliate Agent's Configuration Wizard allows you to modify the Agent's configuration on a per-Web server basis. The Configuration Wizard is only available after you install the SAML Affiliate Agent.

You can run the Configuration Wizard for each Web Server that requires a unique configuration. To establish a unique configuration for a Web server, select a single Web server and run the Wizard. Upon completing the configuration, the selected Web server will have its own AffiliateConfig.xml configuration file; no other Web servers will be affected. Repeat this procedure for each server that requires a unique configuration.

Note: Before you modify the configuration, we recommend that you make copies of the registry settings and Agent configuration settings to have as a back up.

Run the Configuration Wizard

The installation instructions that follow reflect the GUI mode prompts. For UNIX systems, you can install the Option Pack by using console mode by executing the Option Pack binary file with the -i console command argument. The command line installation prompts are similar to the GUI mode prompts.

Note the following:

- Running the Configuration Wizard using the Exceed application may cause text in the dialog boxes to be truncated because of unavailable fonts. This limitation has no affect on Agent installation and configuration.
- If you want to run the Configuration Wizard on a UNIX system via telnet or other terminal emulation software, you must have an X-Windows session running in the background.

  Additionally, you need to set the DISPLAY variable to your terminal, as follows:

  DISPLAY=111.11.1.12:0.0

  export DISPLAY

  If you try to run in GUI mode on a UNIX system via a telnet window without an X-Windows session, the installer throws an Java exception and exits.

  If you prefer, you can run the Wizard from the command-line in a console window.
- The variable *saml_affiliate_agent_home* refers to the installed location of the SAML Affiliate Agent.

To run the Configuration Wizard

1. Go to *saml_affiliate_agent_home*/config.
2. Do one of the following to start the Wizard:
   - For Windows: double-click affl_config.exe.
   - For UNIX: enter the following in a console window:

     ./affl_config.bin
3. Review the information in the Introduction dialog box and click Next.
4. In the Web Server dialog box, select one Web Server that you want to customize, then click Next.
5. Modify the entries in the URL Information dialog box, but note the following about adding entries:

   When prompted to enter a root URL, use the following syntax:

   http://address.domain.com:port or https://address.domain.com:port

   Do not enter any additional text.

When you specify a value for a root URL, the installation script appends additional information to it in the AffiliateConfig.xml file. For example, if you enter https://interceptor.domain.com:90 for the Federation Web Services Root URL, the script enters https://interceptor.domain.com:90/smafa/amts/test1.htm in the AffiliateConfig.xml file.

The entries are as follows:

Affiliate Cookie Domain

Enter the domain for the local server where the SAML Affiliate Agent is installed, such as .mydomain.com.

SSL Interceptor Root URL

Enter the URL at the consumer site where the producer redirects users during consumer requests. This is a URL to the consumer's secure Web server where the SAML Affiliate Agent is installed. We recommend that you use an SSL connection and that the URL begin with https://, such as https://mysslserver.example.com:90

The SSLInterceptorURL enables the SAML Affiliate Agent to obtain the SAML artifact, which identifies the SAML assertion stored at the producer. The assertion contains user profile and session information. After the Agent gets the artifact, it makes a call on the SSL back channel to the producer to retrieve the actual assertion.

For all web servers, you must add the HTTPSPorts attribute to the AffiliateConfig.xml file and specify the same port number as you specify for the SSLInterceptorURL attribute. The HTTPSPorts attribute must be added to the GlobalInfo tag in the AffiliateConfig.xml file.

Important! The SAML Affiliate Agent is *not* compatible with the FIPS 140-2 encryption standards.

Federation Web Services Root URL

Enter the URL to the Web server at the producer where the Web Agent and Web Agent Option Pack are installed. This must be a secure URL that begins in the form https://, such as https://myserver.ca.com:81

6. Modify the entries in the Passwords dialog box:
   a. Enter the Shared Secret twice. This is the secret that the SAML Affiliate Agent uses to encrypt consumer cookies. Confirm the secret by re-entering it.
   b. Enter the Affiliate Password twice. This is the password that the SAML Affiliate Agent uses to communicate with the Policy Server at the producer site. Confirm the password by re-entering it.

      This password must match the password for a consumer defined in the Policy Server User Interface. For more information about consumers, see the *Federation Security Services Guide*.
7. Review the information in the Configuration Summary dialog box, then click Install.

8. Upon completion, you will have a unique AffiliateConfig.xml file in one of the following locations:

   - IIS

     saml_affiliate_agent_home\bin\IIS

   - Sun Java System

     Windows: *Sun_Java_System_home*\https-*hostname*\config

     UNIX: *Sun_Java_System_home*/https-*hostname*/config

   - Apache

     Windows: *Apache_home*\conf

     UNIX: *Apache_home*/conf

   These locations apply only after running the Configuration Wizard. When you first install the Agent, the file's default location is *saml_affiliate_agent_home*\config (Windows) or *saml_affiliate_agent_home*/config (UNIX).

9. Re-run the Configuration Wizard for each configuration you want to modify.

Chapter 5: Configure Global Settings

This section contains the following topics:

Global Configuration Settings

The global configuration attributes are used across all producers and all consumers defined in the AffiliateConfig.xml file.

This chapter contains the global settings in the GlobalInfo section of the AffiliateConfig.xml file. To see these settings, view the AffiliateConfig.xml file in *saml_affiliate_agent_home*/config.

Processing URL-Encoded Targets

If the URLs of your target resources are URL-encoded, you need to set the following parameter for the SAML Affiliate Agent:

SupportURLEncodedTarget

Specifies how the SAML Affiliate Agent processes a target URL that includes URL-encoded characters. If the value of this parameter is set to yes, then the SAML Affiliate Agent double encodes TARGETURL query parameter (if it is already URL-encoded) before redirecting to the portal. The user will be redirected to URL encoded target if access is granted.

If the value of this parameter is set to no, then the SAML Affiliate Agent performs URL decoding before redirecting the user to the target page.

The setting of this parameter has no effect on target resources that are not URL encoded.

If your target resources use URL encoding, set the value of this parameter to yes.

Enable the Agent to Work with Multiple AuthTrans Functions

AuthTrans functions are directives that initialize the Sun ONE Web Server. When a SAML Affiliate Agent is installed on a Sun ONE Web server, you can enable the Agent to operate with other AuthTrans functions. These functions are defined in the Web server's obj.conf file.

The Web server executes AuthTrans functions in the order that they are listed in the obj.conf file. By default, the SAML Affiliate Agent is the first AuthTrans function (AuthTrans fn="SiteMinderAffiliate") and it returns a REQ_PROCEED. When the Sun ONE server reads finds a function that returns a REQ_PROCEED command, it executes this command; no other AuthTrans functions are executed.

To allow other AuthTrans functions to execute (for example, Oracle Application Server relies on an AuthTrans function to execute), set the EnableOtherAuthTrans attribute to yes. For example:

EnableOtherAuthTrans="yes"

HTTP and HTTPS Port Number Configuration

The following parameters enable you to set HTTP and HTTPS port numbers:

- GetPortFromHeaders
- HTTPSPorts

Ensuring HTTP Request and Response Port Numbers Match

You may have a network architecture where the port number in a request does not always match the port number the SAML Affiliate Agent uses when sending a response.

The GetPortFromHeaders parameter lets the SAML Affiliate Agent create a header with the originally requested port number. The Agent then uses this port number instead of pulling the port number from the web server service structures so the request and response port numbers match.

The GetPortFromHeaders setting is also useful for applications that redirect traffic to specific web servers for load balancing without modifying the actual HTTP headers. In this case, when the Agent performs redirects, the user must be redirected back to the proper external port, and not the internal one used by the web server.

To use the port number in the HTTP HOST request header, set the GetPortFromHeaders attribute to yes, as follows:

GetPortFromHeader="yes"

The default is no.

Note: You must enable this attribute for a SAML Affiliate Agent on an Apache Web server.

HTTPS Ports for SSL Connections

You may want to secure requests to the SAML Affiliate Agent using SSL connections.

The HTTPSPorts parameter specifies the secure ports the SAML Affiliate Agent listens on if you are using an SSL connection to the web server where the Agent is installed. If you specify a value for this parameter, you must include all the ports for all the web servers that serve secure requests. If you do not specify a value, the Agent reads the HTTP scheme from the web server's context.

If a server is behind an SSL accelerator (which converts HTTPS to HTTP), the requests are treated as SSL connections by your browser.

To define your HTTPS ports, set the value of the HTTPSPorts parameter to the port numbers that use SSL. Use commas to separate multiple port numbers.

Example (single port)

HTTPSPorts="80"

Example (multiple ports)

HTTPSPorts="223, 224"

In the file, it should be entered as an attribute in the GlobalInfo tag, as follows:

<GlobalInfo HTTPWrapSpec="RFC-2047" LegacyVariables="no"
AllowCacheHeaders="no" GetPortFromHeaders="no"
EnableOtherAuthTrans="no" HTTPHeaderEncodingSpec="UTF-8"
HTTPSPorts="80">

Avoid Overriding the HTTPS Ports Configuration

You may have configured the HTTPSPorts parameter to specify ports for SSL connections to the SAML Affiliate Agent. To ensure that the value of the HTTPSPorts parameter is not overwritten, set the NonSSLInterceptorURL parameter.

Set this parameter with a value similar to the SSLInterceptorURL parameter, however, this parameter must start with **http** and the port specified will serve as the non-SSL port, such as port 80.

Example

If the SSLInterceptorURL parameter is set to:

<SSLInterceptorURL>https://2k3.clearcase.com:443/smafa/amts/test1.htm</SSLInterceptorURL>

the NonSSLInterceptorURL parameter should be set to:

<NonSSLInterceptorURL>http://2k3.clearcase.com:80/smafa/amts/test1.htm</NonSSLInterceptorURL>

Important! You must specify a port number with each of these parameters.

Configure HTTP Header Operation

The following sections discuss configuring the HTTP Header Operation.

Set HTTP Header Syntax for Legacy Variables

The LegacyVariables attribute determines the syntax that the SAML Affiliate Agent uses for HTTP header names. The legacy syntax uses an underscore (_) between the SM prefix and the header name, for example SM_*header_name*. For some Sun ONE and IIS servers, the underscore may cause problems with certain applications. This attribute enables you to eliminate the underscore if it causes problems with your applications.

The default value is no, which instructs the Agent not to use the underscore, for example SMHeaderName.

To use the underscore, set this attribute to yes:

LegacyVariables="yes"

The header variables then use the syntax SM_HeaderName.

Allow HTTP Header Caching

The AllowCacheHeaders attribute tells the SAML Affiliate Agent how to handle cache-related request headers. Specifically, this settings tells the Agent whether or not it should remove the if-modified-since or if-none-match request headers before it passes a request to the web server where it is installed. The action taken by the SAML Affiliate Agent affects whether or not a browser uses cached pages.

Note: This attribute does not affect auto-authorized resources. Auto-authorized resources include those matched by the IgnoreExt setting. For these files, the default cache operations are determined by the browser's and web server's own cache settings.

The three options for this attribute are:

Yes

If you enter this option, the SAML Affiliate Agent ignores cache-related headers and takes no action. The default cache operations are determined by the browser's and web server's own cache settings.

Important! Setting AllowCacheHeaders to Yes introduces security issues. Pages which are personalized by an application on the web server but do not have the appropriate cache control headers set may become cached in the browser or any HTTP intermediary. This can introduce unexpected behavior and allow a browser to save sensitive data to the disk.

Additionally, setting this parameter Yes prevents the SAML Affiliate Agent from enforcing session timeouts across all resources. If a resource is served from browser cache, the Agent has no chance to get the SMSESSION cookie and validate the session. Therefore, be sure to evaluate your session security needs before enabling this setting.

No (default)

If you choose this option, the SAML Affiliate Agent removes the cache-related headers from requests for protected resources. The web server now treats the request as an unconditional request, and will not perform any cache validation operations.

Note: If you are configuring the LogOffURI, we recommend accepting this default value. Otherwise, the browser will deliver a cached version of the LogOffURI resource and the user session will not be terminated.

None

If you specify this option, the SAML Affiliate Agent tells the web server to remove all cache-related headers for protected and unprotected resources.

If the session has been terminated, the browser will not use what is in cache, regardless of this setting.

Note: See RFC 2616, Section 13 "Caching in HTTP" for more information about HTTP/1.1 caching mechanisms"

Encode and Wrap HTTP Headers

The following sections discuss encoding and wrapping HTTP headers.

HTTP Header Encoding

The HTTPHeaderEncodingSpec attribute dictates how all HTTP header values and all custom HTTP-COOKIE responses are encoded. The default value is UTF-8 with no wrapping.

To set the Encoding to...	Set the Attribute To...
UTF-8 (default)	HTTPHeaderEncodingSpec="UTF-8"
Shift-JIS	HTTPHeaderEncodingSpec="Shift-JIS"
EUC-J	HTTPHeaderEncodingSpec="EUC-J"
ISO-2022-JP	HTTPHeaderEncodingSpec="ISO-2022-JP"

HTTP Header Wrapping

The HTTPWrapSpec lets you further encode characters that are not legal in HTTP traffic, the protocol used to pass cookies back and forth between the browser and the consumer. For example, some Shift-JIS characters will cause undesirable results if they are not further encoded by RFC-2047.

When you set HTTPWrapSpec, the data is first encoded according to the configured encoding specification, then the data is further encoded following the RFC-2047 specification.

HTTPWrapSpec is an optional attribute and can only accept the value RFC-2047 with no spaces, as follows:

HTTPWrapSpec="RFC-2047"

Log Error Messages

The LogSettings element of the configuration file defines how error messages are displayed.

Log messages can be:

- recorded in a log file
- displayed in an NT console window

You determine the type of messages logged by setting the log level.

Note the following:

- You cannot log message to a console on UNIX platforms.
- The LogLevel attribute is the only logging attribute in the AffiliateConfig.xml file used by the SAML Affiliate Agent installed on an IIS 6.0 or Apache 1.x Web server. These Web servers also use the log settings in the affiliateserverconf.properties file.

More Information

Configure the Affiliate Server (see page 55)

Record Messages in a Log File

To write error messages to a log file, set the LogToFile attribute to Yes and enter a file name in the LogFile attribute. For example:

```
LogToFile="yes"
<LogFile>/opt/netegrity/affiliateagent/log/affiliate.log</LogFile>
```

Note: This parameter is *not* supported for SAML Affiliate Agents configured with IIS 6.0 and Apache 1.x web servers. When used with these web servers, the Affiliate Server and the SAML Affiliate Agent both use a single log file. The logfilename directive specifies the location of the log file.

Specify Log Levels

You can configure the SAML Affiliate Agent to generate different levels of log messages. Choosing a log level can facilitate troubleshooting by dictating the severity and extent of the logged messages.

The following is an example of a log message that includes the log level. The log level is in bold.

3:49:57/94468/238/**2**>Process-User 'jstar' accessing resource /realmA/page1.html

The following table lists the log levels that you can select.

Log Level	Type of Messages
0	Critical error messages only with the least amount of detail. For example, the inability of the SAML Affiliate Agent to communicate with the SAML Affiliate Agent. This is the default log level.

Log Level	Type of Messages
1	Warning error messages, trace messages, and flow state messages. These messages provide information about processes without a lot of detail.
2	Various data messages, including those of the least importance, such as header details and cookie variables, with the highest level of detail.

For the chosen log level, the SAML Affiliate Agent prints messages for that level and messages from any lower level. For example, if you choose level 2, the SAML Affiliate Agent prints all messages from levels 0 through 2.

To specify the log level

1. Set the LogLevel element to 0, 1, or 2.

   For example: <LogLevel>2</LogLevel>

   Note: If the SAML Affiliate Agent uses SMConAPI to produce tracing messages, the following message may be included in the log: SMConAPI - Trace:at least one of the NETE_PS_RPPT or NETE_WA_ROOT has to be defined. You can ignore this message. When the Agent does not find either variable, it uses build-in variable values.

2. If you are using any of the following web servers, you must restart the web server to apply the changes:

   - IIS 5.0
   - Apache 1.x, 2.x (*except* those on UNIX platforms)
   - Sun Java System (UNIX platforms only)

Append Messages to Log Files

To add messages to an existing log file instead of rewriting the entire file each time logging is invoked, enable the Append attribute, as follows:

Append="yes"

Note: This parameter is *not* supported for SAML Affiliate Agents configured with IIS 6.0 and Apache 1.x web servers. When used with these web servers, the Affiliate Server and the SAML Affiliate Agent both use a single log file. The logfilename directive specifies the location of the log file.

How to Display Log Messages in a Console Window (Windows Only)

Before you can log messages to a console, you must do the following:

1. Enable the World Wide Web Publishing service or the Sun ONE Web server to interact with the desktop.
2. Set the LogtoConsole parameter.

Logging messages to a console is not supported for IIS 5 on Windows 2000 and 2003, or for Apache 2.0.43 on any Windows platform.

Configure the Web Service to Interact with the Desktop

These are general procedures. For specific instructions for your Windows platform, refer to Windows documentation.

To configure the web service to interact with the desktop

1. Open up the Services Control Panel.
2. Double click on the Sun ONE server.

   The server properties dialog box opens.
3. Locate the log on properties for the local account and select the Allow Service to Interact with Desktop option.
4. Close the Control Panel.

Enable Console Logging

After you enable the Windows service to interact with the desktop, you can enable the setting for console logging.

To display log messages in a console window

1. In the AffiliateConfig.xml file, set the LogToConsole attribute to Yes. For example:

   LogToConsole="yes"

   Note: This parameter is *not* supported for SAML Affiliate Agents configured with IIS 6.0 and Apache 1.x web servers. When used with these web servers, the Affiliate Server and the SAML Affiliate Agent both use a single log file. The logfilename directive specifies the location of the log file.
2. Restart the Web server.

   When the Web server restarts, the command prompt window opens and the SAML Affiliate Agent starts logging information in the window.

To stop displaying messages in the console window, set LogToConsole to no.

Return Users to the Affiliate after Obtaining a SAML Artifact

When a user visits a consumer and requests a protected consumer resource, they are redirected to the producer. The redirect URL contains the SSLInterceptorURL, which the SAML Affiliate Agent adds as a query parameter. At the producer site, the producer adds a SAML artifact as an additional query parameter, then sends the user back to the SSLInterceptorURL at the consumer. Using SSL ensures that the artifact is not sent in clear text.

No content should be bound to the SSLInterceptorURL because the request is never passed to the Web server. Instead, the SAML Affiliate Agent examines the URL, extracts the information in the query parameters, and processes the request. The request will fail if the query parameters on the URL are invalid.

Note: We strongly recommend that the URL be sent over an SSL connection to ensure that the SAML artifact is not sent in clear text.

To specify the SSLInterceptorURL, enter a valid URL at the consumer Web server where the producer will redirect users, for example:

<SSLInterceptorURL>https://interceptor.domain.com:90/affiliateagent/afftsite/test1.htm</SSLInterceptorURL>

Important! You must specify a port number for this parameter.

Configure PKI for the SSL Connection to the Producer

The Public Key Infrastructure (PKI) attributes configure the SSL connection to the producer.

Set the Location of the Key Store

The KeyStoreLocation attribute defines the location of the repository where information about the root certificate authorities is stored. It is the full name and path of the key store location.

To set the location of the key store, enter a path name on the affiliate web server, such as:

<KeyStoreLocation>/export/usr/netegrity/affiliateagent/AM.keystore</KeyStoreLocation>

If the key store has been modified or replaced with a new store, you must restart the web server plug-in for the SAML Affiliate Agent, and restart the Affiliate Server, which is a UNIX daemon or NT service. To restart the web server plug-in, stop and restart the web server.

More Information

Restart the Affiliate Server (see page 59)

Modify the Key Store Password

The KeystorePassword attribute specifies the password required to retrieve the root certificates from the key store. The consumer needs these root certificates so that it knows to trust the SSL server at the producer.

For instructions on modifying the KeystorePassword attribute, contact Customer Support.

Define the Key Model for Encrypting Cookies

A KeyConfig section defines the encryption of consumer cookies between the browser and the consumer.

Important! The SAML Affiliate Agent is *not* compatible with the FIPS 140-2 encryption standards.

Specify the Key Model

The KeyModel attribute determines how cookies sent between the browser and the SAML Affiliate Agent are encrypted and decrypted.

You can select one of the following key models:

- SharedSecret
- KeyProviderLibrary

In most cases, using the shared secret method is sufficient. Choose the key provider library method if you want an even higher level of security.

To select a key model

1. Specify one of the following in the AffiliateConfig.xml file:
   - <KeyModel>SharedSecret</KeyModel>
   - <KeyModel>KeyProviderLibrary</KeyModel>
2. Depending on which key model you choose, configure the key attribute.

More Information

Configure the Shared Secret (see page 78)
Configure the Key Provider Library (see page 79)

Configure the Shared Secret

The shared secret is an encryption key used for encrypting consumer cookies between the browser and the consumer. By default, the value for this attribute is the key that you entered when installing the SAML Affiliate Agent. For security reasons, this value is masked.

Shared secret is a valid key attribute only when KeyModel=SharedSecret, which is the default KeyModel setting.

The following is an example of the shared secret attribute:

<SharedSecret>yGT1uXzscC2E3yGPxak2txVl4q+eg2bu</SharedSecret>

Note: The SAML Affiliate Agent's shared secret is independent of any shared secret set at the producer-side Web Agent and Policy Server. The SAML Affiliate Agent's shared secret does not have to match the producer's shared secret. However, if more than one SAML Affiliate Agent is installed, for example, in a clustered web server environment, the shared secret for each web server must match.

More Information

Extend a SAML Affiliate Agent (see page 139)

Configure the Key Provider Library

The key provider is an alternative to the shared secret for encrypting cookies. It offers additional security by using a key that can be configured to "rollover," or change at a user-determined time interval, whereas the encrypted shared secret is a static key attribute.

If you set the KeyModel element to KeyProviderLibrary, you must configure the KeyProviderLibrary and KeyRolloverInterval attributes. These two attributes are valid only when KeyModel=KeyProviderLibrary.

The library used by the key provider must be provided by the consumer site. This library implements the Key Provider API, which provides the keys that encrypt the data in the consumer cookies. To direct the SAML Affiliate Agent to the key provider library, enter the full library path name. For example:

<KeyProviderLibrary>/export/affiliateagent/keylibrary.so
</KeyProviderLibrary>

Set the KeyRolloverInterval Time

If you select KeyProviderLibrary as the KeyModel, you need to specify the frequency at which the existing key is changed.

The KeyRolloverInterval attribute specifies the time, in seconds before the current key expires and a new one replaces it.

Set a time for the consumer's encrypted keys to be changed by entering a value in seconds. For example:

<KeyRolloverInterval>900</KeyRolloverInterval>

When a key expires at the specified time, it rolls over and a new key replaces it.

Modify the Shared Secret with Encryptkey

If you need to modify the shared secret after installing the SAML Affiliate Agent, run the EncryptKey utility. Do not modify the value in the AffiliateConfig.xml file. The shared secret can be between 1 and 255 characters with no embedded spaces. Note also that it is case-sensitive.

Important! The SAML Affiliate Agent is *not* compatible with the FIPS 140-2 encryption standards.

Run encryptkey using the following arguments:

encryptkey -path *path_of_AffiliateConfig.xml* -sharedSecret *new_secret*

path_to_AffiliateConfig.xml must contain the file name of the configuration file. If any value in the path contains spaces, the entire path must be surrounded by quotation marks.

For example:

encryptkey -path "C:\Program Files\netegrity\affiliateagent\config\AffiliateConfig.xml" -sharedSecret mysecret

After running the encryptkey utility, the new secret is inserted into the AffiliateConfig.xml file.

Handle Security Issues in Request URLs

The following sections discuss handling security issues in Request URLs.

Specify Ignored File Extensions

The IgnoreExtensions element lists the file extensions that the SAML Affiliate Agent ignores when a request is received. The default values (.gif, .jpeg, .png, .ccc) are the most common file extensions for the SAML Affiliate Agent to ignore. This setting instructs the Agent to pass requests for files with these extensions directly to the web server. Extensions are included in the IgnoreExtensions element because they specify types of files that do not require as much security as other resources.

The SAML Affiliate Agent ignores extensions if the URI of the protected resource contains only one period (.), for example, /image.gif. In this example, the SAML Affiliate Agent passes the request directly to the web server.

If the URI has two or more periods, the SAML Affiliate Agent does *not* ignore the extension. The SAML Affiliate Agent cannot determine how to interpret the two periods so it cannot figure out which part of the URI represents the resource that the server delivers. For example, if the URI is /dir1/app.pl/file1.new.gif, the SAML Affiliate Agent thinks the resource is protected and will not pass the request to the web server even though .gif is specified in the IgnoreExtensions element.

Note the following when adding extensions to the IgnoreExtensions element:

- extensions are not case sensitive
- extensions are separated by commas or spaces

For example:

<IgnoreExtensions>.gif .jpeg .png .ccc</IgnoreExtensions>

Note: To protect URLs that do not have periods, ensure that protected resources do not have extensions in the IgnoreExtensions element.

Secure the Ignore Extensions Feature

An unauthorized user can append a bogus resource to the end of the URL when making a request. Because not all URLs have periods, which is common with servlets—for example, /mydir/myservlet, if a user adds a bogus resource, such as /mydir/myservlet/file.gif the SAML Affiliate Agent is configured to ignore it because of the single period and the .gif extension. Consequently, the unauthorized user gains access to the resource /mydir/myservlet because the servlet engine recognizes the path to myservlet.

Be sure to consider the security and the performance issues of this situation. If you are most concerned about the security risks, you may want to leave the IgnoreExtensions element blank, but be aware of the following consequences:

- Performance may decrease because the SAML Affiliate Agent will evaluate every image URL on a page.
- Behavior of your Web site may change because users may be challenged for resources that formerly did not require authentication.

Designate Bad URL Characters

In the Agent's configuration you can list a set of character sequences that cannot be part of a URL request. These are treated by the Agent as bad URL characters. The SAML Affiliate Agent will refuse URL requests that contain any of the characters or strings of characters that you include in this list. The checking is done on the URL before the "?" character.

By default, the SAML Affiliate Agent rejects URL requests that include these characters:

- a backward slash (\)
- two forward slashes (//)
- period and a forward slash (./)
- forward slash and a period (/.)
- forward slash and an asterisk (/*)
- an asterisk and a period (*.)
- a tilde (~)
- a comma (,)
- a dash (-)
- a percent sign (%)
- %00-%1f
- %7f-%ff
- %25
- %25u
- %25U

These default characters block URLs that might allow a malicious Web client to evade SiteMinder rules.

To specify bad URL characters, add to the list to meet the needs of your applications. Separate the bad URL characters by a comma; do not use spaces.

For example, set BadURLChars to the following:

//,./,/../*,*.,~,\,-,%,space,%00-%1f,%7f-%ff,%25,%25u,%25U

You can use the bad URL characters in CGI parameters if the question mark (?) precedes the bad URL characters.

Designate Bad Query Characters

If you do not wish to allow escaped characters in query data, you can block the specified characters by using the BadQueryChars element. This element is configured in the same manner and works exactly the same as the BadURLChars element, except that it checks characters that follow the query character (?), or "hook," in a URL. For example:

BadQueryChars="%25"

By default, this setting does not function unless specified values are entered for the element.

Protect Web Sites Against Cross-Site Scripting

A cross site scripting (CSS) attack can occur when the input text from the browser (typically, data from a post, or data from query parameters on a URL) is displayed by an application without being filtered for characters that may form a valid, executable script when displayed at the browser.

An attack URL can be presented to unsuspecting users. When the web server receives the request, an application may return to the browser a display that includes the input characters, perhaps along with an error message about bad parameters on the query string. The display of these parameters at the browser can lead to an unwanted script being executed on the browser.

For example, when a user types news into a search engine web page, the application normally might return a blank field, or a response, such as:

Your search for news returned the following:

However, in response to an attack URL, the browser might receive a response, such as:

news< script>BadProgram< /script>

Specify Which Characters to Check for CSS Events

When CSSChecking is set to yes, the SAML Affiliate Agent scans a full URL (including the query string) for the presence of escaped and unescaped versions of the following default character set:

- left and right angle brackets (< and >)
- single quote (’)

For the SAML Affiliate Agent to check for these characters, you must set the BadCSSChars attribute to a character set of your choice.

Include the entire string of characters that you want. For example, if you set BadCSSChars to %3C, %3E (characters for left and right angle brackets), the Agent scans only for these brackets.

Important! If you experience a problem related to these characters, the SAML Affiliate Agent error log shows the following error message: Caught Possible Cross Site Scripting Violation in URL. Exiting with HTTP 403 ACCESS FORBIDDEN.

Some applications require the use of the quote characters in the query string, irrespective of the Web server platform. To use applications such as these, set BadCssChars as follows:

<BadCssChars>%3C,%3E, '</BadCssChars>

If you omit this attribute, the SAML Affiliate Agent will not check for the any characters.

Configure the Error Message Response to a CSS Event

If the SAML Affiliate Agent detects a CSS event, you can configure the error message that the Agent sends to the user's browser by specifying an error file location in the CSSErrorFile attribute.

The CSSErrorFile might be a file path or a URL, for example:

- CSSErrorFile="/export/netegrity/affiliateagent/messages/csserror.htm"
- CSSErrorFile="C:\error\error.txt"
- CSSErrorFile="http://www.mycompany.com/error.jsp"

For more information about cross-site scripting, see CERT Advisory.

More Information

Extend a SAML Affiliate Agent (see page 139)

Preserve Assertion Data

You can store attribute data that the SAML Affiliate Agent retrieves from the assertion sent by the producer. The PersistenceLibrary attribute specifies the full name and path of the shared library that implements the Persistence API.

The SAML Affiliate Agent provides the data to be stored; the library specified in the PersistenceLibrary attribute determines where the data is actually stored, such as a database.

To direct the SAML Affiliate Agent to call the persistence library, enter the library path. For example:

- Windows:

  <PersistenceLibrary>c:\program files\netegrity\affiliateagent\bin\ assertionlib.dll</PersistenceLibrary>

- UNIX:

  <PersistenceLibrary>/opt/netegrity/affiliateagent/bin/ assertionlib.so</PersistenceLibrary>

Note: The persistence library must be provided by the consumer.

Locate a Custom Message File

The CustomMessageFile attribute designates the location of text messages used for the SAML Affiliate Agent's logging feature. There is no need to change this file. The default is as follows:

<CustomMessageFile>*saml_affiliate_agent_home*\affiliateagent\messages\ iso8859-1.msg</CustomMessageFile>

Track User Activity

The SAML Affiliate Agent automatically generates a unique transaction ID for each user authorization request. Each new request generates a new transaction ID. The SAML Affiliate Agent sends this ID to the producer, where it is placed in the producer's Web server log.

After the user is successfully authorized to access a resource, the SAML Affiliate Agent adds this ID to its HTTP header variables and records the ID in its Web Server log. Using this ID, you can follow user activity for a given application.

Note: This feature has no configuration setting.

Log the Transaction ID for an IIS Web Server

To see the transaction ID in a web server log, specifically configure the Web server hosting the SAML Affiliate Agent to log the ID.

For IIS Web servers, the SAML Affiliate Agent logs the transaction ID as a URL element. To see this ID in the Web server log, configure the IIS Web server to log URL query elements. The transaction ID appears in the log as a mock query element.

For example:

123.12.12.1, user1, 2/11/05, 15:30:10, W3SVC, MYSERVER, 11.22.33.44, 26844, 47, 101, 400, 123, GET, /realma/index.html, STATE=MA&**SMTRANSACTIONID=0c01a8c0-01f0-38a47152-01ad-02714ae1**

The transaction ID is appended to the end of an existing query string—in this example, STATE=MA.

If there are no query elements already in the URL, the Agent adds the transaction ID at the end of the log entry. For example:

123.12.12.1, user1, 2/11/00, 15:30:10, W3SVC, MYSERVER, 11.22.33.44, 26844, 47, 101, 400, 123, GET, /realma/index.html, **SMTRANSACTIONID=0c01a8c0-01f0-38a47152-01ad-02714ae1**

Log the Transaction ID for a Sun ONE Web Server

To see the Transaction ID in a web server log, specifically configure the Web server hosting the SAML Affiliate Agent to log the ID.

To see the transaction ID in the Sun ONE Web server log, you have to add the SMTRANSACTIONID header variable to the server's obj.conf file. The variable is %Req->headers.smtransactionid%. Add this variable to the existing list of HTTP server variables that you want to log when the Web server initializes.

For example:

Init fn="flex-init" access="D:/Servers/https-myserver/logs/access" format.access= "%Ses->client.ip% - %Req->vars.auth-user% [%SYSDATE%] \"%Req->srvhdrs.clf-status% %Req->srvhdrs.content-length% **%Req->headers.smtransactionid%"**

In this example, the transaction ID is appended to the end of an existing obj.conf entry, however it can be placed anywhere in the list of variables.

The following is an example of a Web server log entry with the transaction ID in bold:

11.22.33.44 - user1 [27/Jan/2000:16:12:24 -0500] "GET /Anon/index.html HTTP/1.0" 200 748 **3890b4b9-58f8-4a74df53-07f6-0002df8**

Log the Transaction ID for an Apache Web Server

To see the transaction ID in the Apache Web server log, you have to add the SM_TRANSACTIONID header variable to the LogFormat directive in the server's httpd.conf file. For example:

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{SMTRANSACTIONID}i\"" common
```

For more information about the httpd.conf file and the LogFormat directive, refer to the Apache Web server documentation.

Chapter 6: Configure Portal Settings

This section contains the following topics:

Portal Settings

The AffiliateConfig.xml file contains one or more sections that define communication between the producer and the consumer. These portal settings apply on a per producer basis.

To see the portal section, view the AffiliateConfig.xml file located in *saml_affiliate_agent_home*/config. The settings are part of the PerPortalInfo element.

Specify the Number of SSL Connections for Communication

The MaxSSLConnections attribute defines the number of SSL connections available for communication between the SAML Affiliate Agent and the producer. The default number of SSL connections is 5.

Important! The SAML Affiliate Agent is *not* compatible with the FIPS 140-2 encryption standards.

MaxSSLConnections="5"

To use more threads for communication, you can increase this number.

Follow these guidelines when configuring this attribute:

- The minimum value should be 1.
- The maximum value should not exceed the value of the threadpoolsize attribute specified in the affiliateserver.properties file.

  Note: If the AffiliateConfig.xml file has multiple PerPortalInfo sections, the sum of all MaxSSLConnections values should not exceed the threadpoolsize. If it does, the user may see a 500 server error in the consumer log file for some requests.

Specify the Portal Name

The PortalName element identifies the producer (portal) associated with a consumer site. This value is known only to the SAML Affiliate Agent—it is used by the consumer to maintain information about a particular producer.

Specify a name by entering a string up to a maximum 256 characters. For example:

<PortalName>SamplePortal</PortalName>

Redirect Users without Valid Affiliate Cookies to the Producer

When a user makes a request for a protected consumer resource, but that user's browser does not contain valid consumer profile and session cookies, the user is redirected to the producer. The PortalQueryURL is the URL at the producer where the user is sent.

This PortalQueryURL must direct the user to an intersite transfer service, which redirects a user from the producer to the consumer.

Configure the PortalQueryURL by entering a URL to the intersite transfer service at the producer. For example:

<PortalQueryURL>https://secure.portal.domain.com:81/affwebservices/public/intersitetransfer/</PortalQueryURL>

After the user goes to the PortalQueryURL, they are sent back to the SSLInterceptorURL with the SAML artifact. The artifact enables the SAML Affiliate Agent to call the producer and retrieve the SAML assertion. After the assertion is returned to the consumer, the SAML Affiliate Agent uses the assertion data to set the consumer cookies in the user's browser.

Configure the Affiliate to Retrieve a SAML Assertion

For the SAML Affiliate Agent to get information about a user requesting a resource, the Agent must retrieve the SAML assertion stored at the producer.

When a user requests a protected consumer resource, that user is sent to the producer to be authenticated. After the user is authenticated, the producer redirects the user to the SSLInterceptorURL at the consumer. This redirection includes the SAML artifact and the protected URL at the consumer that the user tried to access. This artifact references the actual assertion stored in the session server at the producer, and it enables the SAML Affiliate Agent to retrieve the correct assertion.

To retrieve the assertion, the SAML Affiliate Agent intercepts the redirected request from the producer, and calls the assertion service at the producer over a back channel. It then retrieves the SAML assertion.

The GetAssertionService element defines the URL at the producer where the Affiliate Agent retrieves the SAML assertion.

To specify the assertion service, enter a URL to its location at the producer. This must be a URL over an SSL connection. For example:

<GetAssertionService>https://secure.producer.domain.com:81/affwebservices/assertionsretriever</GetAssertionService>

Note: The affwebservices/assertionretriever portion of the assertion service URL is the default installation directory of the Federation Web Services application, which is installed with the Web Agent Option Pack for the producer-side Web Agent. If you install the Option Pack in a non-default location, be sure the correct location is reflected in this URL.

Specify the Session Provider Service

The session provider service is a service at the producer that the SAML Affiliate Agent uses to synchronize the consumer session with the producer session. If you are using a shared session model for federated communication, you need to provide the SAML Affiliate Agent with the URL for this service.

The SessionProviderService element specifies the URL for the session service at the named producer.

Enter a URL path to specify the service. It must be a URL over an SSL connection. For example:

<SessionProviderService>https://secure.producer.domain.com:81/affwebservices/router/session</SessionProviderService>

The affwebservices/router/session portion of this URL is the default installation directory of the Federation Web Services application, which is installed with the Web Agent Option Pack for the producer-side Web Agent. If you install the Option Pack in a non-default location, be sure the correct location is reflected in this URL.

Specify the Notification Service

The notification service lets the SAML Affiliate Agent notify the producer of user activity at the consumer site. When a user accesses URLs at the consumer, the producer can log this data and use it for reporting purposes.

The notification sent by the SAML Affiliate Agent includes the

- Affiliate name
- URL accessed by the user
- User DN
- User session key, if available
- Time of access

Three settings require configuration for the notification service—they are in different sections of the AffiliateConfig.xml file:

- NotificationService (PerPortalInfo section)—specifies the location of the notification service at the producer.
- NotificationURL (PerAffiliateInfo section)—designates URLs at the consumer that trigger the SAML Affiliate Agent to send notification messages to the producer.
- MatchingRule for the NotificationURL (PerAffiliateInfo section)—indicates how the SAML Affiliate Agent compares each URL for which the user gained access, to the NotificationURL. Required with the NotificationURL.

To configure the notification service

1. Specify a URL path for the NotificationService element. The URL must be over an SSL connection. For example:

   <NotificationService>https://secure.producer.domain.com:81/affwebservices/router/notification</NotificationService>

   The affwebservices/router/notification portion of the notification service URL is the default installation directory of the Federation Web Services application. This application is installed with the Web Agent Option Pack for the producer-side Web Agent. If you install the Option Pack in a non-default location, be sure that the correct location is reflected in the URL.

2. Configure the NotificationURL element and MatchingRule in the PerAffiliateInfo section of the AffiliateConfig.xml file.

More Information

Configure the Notification Service (see page 114)

Enable the Agent to Work with Multiple Producers

You can configure the SAML Affiliate Agent to work with more than one producer by dividing areas of the consumer Web site according to the associated producer. After your consumer Web site is set up to accommodate each producer, add a new PerPortalInfo section to the AffiliateConfig.xml file for each producer.

Note: You must define at least one consumer for each producer.

When you add sections to support multiple producers, each section must indicate a unique producer. This enables the SAML Affiliate Agent to redirect the user to the correct producer for the requested resource.

Chapter 7: Configure Affiliate Settings

This section contains the following topics:

Affiliate Settings

The affiliate settings specify how the consumer communicates with the producer and determines how users are managed. The affiliate settings apply on a per-consumer basis.

To see the affiliate settings, view the AffiliateConfig.xml file located in *saml_affiliate_agent_home*/config. The settings are part of the PerAffiliateInfo element.

If your Web server is divided into several consumer sites and you want to use a single AffiliateConfig.xml file for all sites, you need to add a PerAffiliateInfo section and a corresponding PerPortalInfo section for each consumer managed by a single AffililiateConfig.xml file.

Enable and Disable the SAML Affiliate Agent

Before you enable a SAML Affiliate Agent:

- Identify the consumer for the producer before enabling the SAML Affiliate Agent at the consumer site.

  You add a consumer in the Policy Server User Interface at the designated producer.

  Note: For specific instructions, see the *Federation Security Services Guide*.

- Complete the configuration of the SAML Affiliate Agent.

To enable the SAML Affiliate Agent, set the Enabled attribute to yes. The default is no. For example:

Enabled="yes"

To disable the SAML Affiliate Agent, set the Enabled attribute to no, the default.

Identify the Affiliate and Affiliate Resources

The following sections discuss identifying the Affiliate and Affiliate Resources.

Name the Affiliate

The AffiliateName element specifies the name of the consumer. This name is assigned to the consumer by the administrator at the producer. The name is case-sensitive.

Important! This name must match the Affiliate name configured in the Policy Server User Interface at the producer, which defaults to a lower-case value. Ensure the name entered here is lower-case.

Note: To configure consumers, see the *SiteMinder Federation Security Services Guide*.

To identify the consumer, enter its name, for example:

<AffiliateName>SampleAffiliate</AffiliateName>

Set the Affiliate Password

The AffiliatePassword defines the encrypted password used by the SAML Affiliate Agent to connect to the producer. The password is used by the consumer to identify itself to the producer.

The administrator at the producer should provide the password, and it must match at the consumer and producer sites.

Important! Do not modify the password directly in the AffiliateConfig.xml file.

To set the AffiliatePassword

1. Get the unencrypted password from the administrator at the producer.
2. Use the encryptkey utility to set the password in the AffiliateConfig.xml file.

   Run encryptkey by entering the following command:

   encryptkey -path *AffiliateConfig.xml_path* -affiliate *AffiliateName* -affiliatepassword *new_password*

   AffiliateConfig.xml_path

   Specifies the path, including the file name, of the configuration file. If any value in the path has spaces, the entire path must be enclosed by quotation marks. For example:

   encryptkey -path "C:\Program Files\netegrity\affiliateagent\config\AffiliateConfig.xml" -affiliate SampleAffiliate -affiliatepassword mypassword

   AffiliateName

   Specifies the value entered for the AffiliateName attribute. This argument ensures that the password is changed for the correct consumer, especially if the AffiliateConfig.xml file has more than one PerAffiliateInfo section.

   new_password

   Represents the password in clear text.

Designate Affiliate Resources

The AffiliateResource element specifies resources on the Web server protected by the SAML Affiliate Agent. When a user tries to access a protected resource at the consumer site, the SAML Affiliate Agent redirects these requests to the producer site to authenticate the user. Be sure to structure your Web site with this in mind.

You have to use the AffiliateResource element with the MatchingRule attribute. This attribute defines the specific resources at the affiliate.

More Information

Define the Matching Rule for the Affiliate Resource (see page 98)

Define the Matching Rule for the Affiliate Resource

The MatchingRule attribute used with the AffiliateResource element specifies how the SAML Affiliate Agent compares each requested URL to a affiliate resource. This attribute is required.

The two possible values for the matching rule are:

- StrictPrefix
- Substring

Using a StrictPrefix

For the StrictPrefix attribute, the value of the requested URL has to match only the beginning of the URL prefix, before a query string (?). For example, if the entry is:

<AffiliateResource MatchingRule="StrictPrefix">protected</AffiliateResource>

will match: http://www.affiliate.com/protected/...

but will not match: http://www.affiliate.com/basic?protected

Using a Substring Prefix

For the Substring attribute, the value of the requested URL can match before or after the query string (?). For example, if the entry is:

<AffiliateResource MatchingRule="Substring">protected</AffiliateResource>

will match the following:

http://www.affiliate.com/protected/...

http://www.affiliate.com/basic?protected

Specify Affiliate Resources that Use Similar Responses and Headers

If you have consumer resources that have similar response and header requirements from the producer, you do not have to define additional consumers at the producer. Instead, configure only one consumer at the producer, and at the consumer site, use multiple matching rules to define the specific consumer resources that share the same requirements.

Specify Affiliate Resources that Use Unique Responses and Headers

You may have resources at the affiliate Web server that require unique response and header information from the producer. For the SAML Affiliate Agent to manage these requirements, configure the following:

- Multiple PerAffiliateInfo sections in the AffiliateConfig.xml file with unique AffiliateName and AffiliateResource values.

  The unique values enable the Policy Server to call the correct policy at the producer. The AffiliateName enables the SAML Affiliate Agent to distinguish different consumers within the file.

- Unique consumers configured at the producer, each with their own consumer name and password.

  The Policy Server at the producer differentiates the consumer only by the consumer name.

For example, to configure multiple consumers, the AffiliateConfig.xml would include the following sections. Note the lines in bold.

```
<PerAffiliateInfo Enabled="no" AllowPOSTs="no"    AllowUnknownUsers="no"
RequireActivePortalSession="no"><!--     Name of the affiliate asconfigured in the
portal  -->
  <AffiliateName>FirstAffiliate</AffiliateName>
  <CompanySourceID>b818452610a0ea431bff69dd346aeeff83128b6a
     </CompanySourceID>
  <AffiliateResource MatchingRule="StrictPrefix">app1
     </AffiliateResource>
          .
          .
          .
</AffiliateConfig>


<PerAffiliateInfo Enabled="no" AllowPOSTs="no"   AllowUnknownUsers="no"
RequireActivePortalSession="no"><!--  Name    of the affiliate as configured in the
portal  -->
  <AffiliateName>SecondAffiliate</AffiliateName>
    <CompanySourceID>f69dd3460a8ukcxbff69dd346aeeff83128b6a
    </CompanySourceID>
    <AffiliateResource MatchingRule="StrictPrefix">app2
    </AffiliateResource>
          .
          .
          .
</AffiliateConfig>
```

Modify the Consumer Cookie Domain

The CookieDomain element identifies the domain of the affiliate Web server. This is the domain for which the consumer issues cookies and reads cookies during the authorization process. The default value for this element is the domain you specified during the SAML Affiliate Agent installation.

To modify the cookie domain, enter a new domain for the CookieDomain element. The default is .netegrity.com, for example:

<CookieDomain>netegrity.com</CookieDomain>

Identify the Portal

The following sections discuss identifying the portal.

Name the Portal

The PortalName element specifies the name of the producer, or producer site associated with the consumer. The name must match the other PortalName entry in the PerPortalInfo section of the AffiliateConfig.xml file.

To modify the portal name, enter a value for the PortalName element. For example:

<PortalName>SamplePortal</PortalName>

More Information

Specify the Portal Name (see page 90)

Set the Company Source ID

The **CompanySourceID** attribute specifies a producer's source ID. A source ID is defined by the SAML specification standard as a 20-byte binary, hex-encoded number that identifies the producer. This ID, which is used by the SAML Affiliate Agent to identify an assertion issuer, must match the Source ID that the producer uses.

At the producer, the source ID is located in the SAML assertion generator properties file, AMAssertionGenerator.properties.

The default value of the CompanySourceID attribute is the source ID of Netegrity, Inc, as follows:

<CompanySourceID>b818452610a0ea431bff69dd346aeeff83128b6a</CompanySourceID>

Change this value manually after installing the SAML Affiliate Agent.

Use OpenSSL Toolkit to Modify the Company Source ID

You can use the OpenSSL cryptography tool kit to generate a 20-byte hashed Company Source ID value for use with SiteMinder.

To generate a 20-byte hashed Company Source ID

1. Download the OpenSSL from the Open SSL Web Site.

   Only the binary is needed. If the binary release is not available from the site, download the source and compile it locally.

2. Put the source text string, such as mycompanyname into a file, such as id.txt.

3. Run the message digest command:

   openssl dgst -sha1 *file.txt*

   file.txt is the file you created in Step 2.

   The output string is the hexidecimal value of the 20-byte message digest for the input file.

   For example:

   $ openssl dgst -sha1 id.txt

   SHA1(id.txt)= e4eabc78894e2c204d788521812497e021f45c08

   For more information about the dgst command, go to Open SSL.

4. Use the hexidecimal string after the equals sign (see the example in Step) as the CompanySourceID parameter value. Place this value in the following places:

   - The Agent's AffiliateConfig.xml file at the affiliate.
   - The AMAssertionGenerator.properties file at the producer. This file is in the directory *policy_server_home*/config/properties.

Manage User Access to Consumers

The following sections discuss managing user access to consumers.

Permit Unknown Users Access to the Consumer

The AllowUnknownUsers attribute determines whether or not the SAML Affiliate Agent allows unknown users access to the consumer site. Unknown users are users who do not have a session at the producer site before coming to the consumer.

- To deny unknown users access to a consumer resource, accept the default value, which is no.
- To permit unknown users access to a resource, set the attribute to yes:

  AllowUnknownUsers="yes"

Note: If you set this attribute to yes, leave the RequireActivePortalSession element set to no. Unknown users do not get a session cookie at the producer, so you cannot require a producer session.

Refuse Anonymous User Access to the Consumer

Anonymous users cannot access resources that are protected by the SAML Affiliate Agent. Anonymous users are users who have visited the producer site but have not registered there.

Deny a User Access to an Consumer Resource

If the SAML Affiliate Agent denies a user access to a consumer resource, the SAML Affiliate Agent can redirect the user to the URL specified by the NoAccessURL element.

The user can be redirected to a URL at the consumer or the producer. For example, the SAML Affiliate Agent may deny users access if they have not yet been identified at the producer. In this case, the NoAccessURL might redirect the user to a login or registration page at the producer site.

To redirect a user who is denied access to a resource, enter a URL for the NoAccessURL element. For example:

<NoAccessURL>http://122.123.1.1:70/badaccess.html</NoAccessURL>

Note: If you do not specify a valid URL, the user sees a browser error stating that the page cannot be displayed.

Append the Original Affiliate Destination to the NoAccessURL

The AppendTarget attribute instructs the SAML Affiliate Agent to add the original target URL at the consumer to the NoAccessURL as a query parameter.

To add the target URL to the NoAccessURL element, specify the following:

<NoAccessURL AppendTarget="yes">

By setting AppendTarget to yes, the SAML Affiliate Agent appends the word Target followed by the target URL to the NoAccessURL value. For example:

http://122.123.1.1:70/badaccess.html?Target=http://www.affiliate.com/realmf/page1.html%3fparam=one%26param2=two

In the target URL, the SAML Affiliate Agent encodes the hook (?) character as %3f and the ampersand (&) character as %26. This is standard URL encoding, and it ensures that the Web browser interprets the target URL as one value. If the characters are not encoded, the browser may parse the elements.

Communication Across a SiteMinder Federated Network

The following sections discuss communication across a SiteMinder Federated Network.

Determine How the Consumer Authenticates to the Producer

The AuthenticationScheme element determines how the consumer authenticates itself to the producer-side Web server that has Federation Web Services installed. The communication between the Affiliate Server component of the SAML Affiliate Agent can occur over an HTTP or SSL connection.

Important! The SAML Affiliate Agent is *not* compatible with the FIPS 140-2 encryption standards.

Specify one of the following for the AuthenticationScheme element.

- Basic— In this mode, the Affiliate Server communicates with the Federation Web Services application over an SSL back-channel connection. This is the default mode. For example:

  <AuthenticationScheme>Basic</AuthenticationScheme>

- Basic_Unsecure—In this mode, the Affiliate Server communicates with the Federation Web Services application over an HTTP connection. You might choose Basic_Unsecure if, for example, your producer and consumer sites are operating in a secure virtual private network (VPN). For example:

  <AuthenticationScheme>Basic_Unsecure</AuthenticationScheme>

A mix of SSL and non-SSL connections with Federation Web Services is not allowed. If you use Basic for the AuthenticationScheme element, all connections should be secure (SSL); however, if you choose Basic_Unsecure, all connections should be HTTP connections (non-SSL).

Note: If you change the AuthenticationScheme element, re-start the Affiliate Server (Smaffdaemon for UNIX or Affiliate Minder Service for Windows) and restart the Web server.

Designate the Assertion Issuer and Audience

For assertion generation, you have to specify the entity that issues assertions and the assertion audience.

Specify the Assertion Issuer

The AssertionIssuer element references the site that issues assertions for specific affiliates. The consumer accepts assertions from only this issuer. The issuer is determined by the administrator at the producer. Only one assertion issuer is permitted for each PerAffiliateInfo section in the file.

To specify an AssertionIssuer enter either a string value or a URL. For example, if you enter a URL:

<AssertionIssuer>http://www.example.com/SiteMinder</AssertionIssuer>

This element is case-sensitive.

The value that you enter for the AssertionIssuer in the AffiliateConfig.xml file must match the value of the AssertionIssuerID at the producer site. This value is specified in the AMAssertionGenerator.properties file:

siteminder_home/Config/properties/AMAssertionGenerator.properites

Specify the Assertion Audience (Optional)

The AssertionAudience element identifies the location of a document that describes the terms and conditions of the business agreement between the producer and the consumer. The audience is determined by the administrator at the producer.

The AssertionAudience value should not exceed 1K; the entire assertion, which includes user profile data, the SAML audience information, and other standard SAML data should not exceed 4K.

To specify an AssertionAudience, enter a URL. This element is case-sensitive. For example:

<AssertionAudience>http://www.example.com/SampleAudience</AssertionAudience>

You are not required to set this attribute at the consumer or the producer. However, if you do specify an audience at both sites, the values that you enter at the consumer and the producer site must match. At the producer, you specify the audience via the Policy Server User Interface.

Note: For more information on configuring consumers, see the *SiteMinder Federation Security Services Guide*.

Allow Post Actions

The AllowPOSTs attribute tells the SAML Affiliate Agent how to handle initial POST action requests to a protected resource for a user that does not have a SiteMinder session.

These requests can come from users or applications posting data to the consumer on an initial request. For example, a user might fill out a form at one site. The data from the form is then posted to the consumer. Because the user is unknown to the consumer, the SAML Affiliate Agent needs to permit the POST action without redirecting the request to the producer.

Note: Although the SAML Affiliate Agent may be configured to permit POST actions, the POST data is not preserved for future use. The POST data is simply passed through the SAML Affiliate Agent without a redirect to the producer.

For users that already have active sessions, the Agent ignores this setting.

To allow POST actions at the consumer, set the attribute to yes:

AllowPOSTs="yes"

Setting AllowPOSTs to yes allows access for initial posts. After the initial POST request, the Agent allows subsequent POST requests.

By default, this element is set to no so the SAML Affiliate Agent does not allow POSTs to the requested page.

If you do not want unknown users to post data to the URLs protected by the SAML Affiliate Agent, you can allow posts to a less secure area of your Web server that the SAML Affiliate Agent does not monitor.

Note: If the target resource where the user is trying to POST is unprotected, then the SAML Affiliate Agent will allow POSTs any time.

Configure SiteMinder Sessions for Federated Single Sign-on

To provide single sign-on across partners in a trusted relationship, there are three types of session models for federated communication:

- Default
- Active Portal
- Shared

An administrator can assign different models to different affiliated content at their site, depending on the security requirements for that content.

Use the following table as a guide for choosing the appropriate session model.

Session Model	Best Used For
Default	Affiliated content that requires minimal security.
Active Portal	Affiliated content that requires a moderate level of security. This model gives you a low-cost tie to the producer, or producer site The control of the session is essentially one-sided, with the session controlled by the producer. Minimum bandwidth is taken up exchanging session information between the consumer and the producer.

Session Model	Best Used For
Shared	Affiliate content that requires the tightest level of security, such as payment data for a business transaction. This model lets you tie in the actions at the consumer with the session at the producer.

For sessioning, Federation Security Services do not require the use of persistent cookies at the producer; the service can operate with the session or identity cookie. The type of cookie required at the producer depends on how the consumer and producer are sharing information.

The following table describes the different cookies needed for each model:

Session Model	Cookies Required at the Producer	Side that Manages the Session
Default	Identity or session cookie used to establish user profile	SAML Affiliate Agent configuration defines session parameters. A producer session is not required. The identity cookie generated during the authentication process can be used.
Active Portal	Session cookie	Producer defines session parameters, but the consumer (consumer site) manages the session, taking the session configuration from the producer.
Shared	Session cookie	Producer defines session; the producer and consumer sessions are synchronized.

More Information

Default Session Overview (see page 108)
Active Portal Session Overview (see page 109)
Shared Sessions (see page 110)

Session Server Overview

The session server is a module of the Policy Server that maintains information about SiteMinder sessions. The session server enables the Policy Server to store a user's session information in a common database called the session store. The session store serves as a temporary container for the SAML assertion that is eventually sent to the consumer. Sessions created in the session server are persistent.

Default Session Overview

The default session model is defined by separate sessions at the producer and the consumer. Each side maintains session information locally, relying on timeout settings to terminate a session. When the producer and consumer timeout values are reached, a user must log in again to access the site.

The timeout settings for the Default session model are:

- SessionIdleTimeout
- SessionMaxTimeout

You can set the SessionIdleTimeout attribute to allow the maximum number of seconds that a user can wait between consumer requests. Set the SessionMaxTimeout attribute to allow the maximum number of seconds that a default session is valid. For example:

<SessionIdleTimeout>300</SessionIdleTimeout>

<SessionMaxTimeout>600</SessionMaxTimeout>

The Default model is the default value for the SessionModel attribute. This type of model is appropriate for general information where a less strict level of security may be warranted.

Note: If you configure a Default session model and specify yes for the RequireActivePortalSession element, you are essentially configuring an active portal session.

Configure a Default Session

To configure a default session at the producer:

There is no specific setting to configure a default session.

To configure a default session at the consumer:

Set the attributes in the AffiliateConfig.xml as follows:

<SessionModel>Default</SessionModel>

<SessionIdleTimeout>300</SessionIdleTimeout>

<SessionMaxTimeout>600</SessionMaxTimeout>

Active Portal Session Overview

To provide an additional level of security at the consumer site, the consumer can require that users have an active session at the producer, or producer site. This ensures that the user is currently authenticated and active at the producer. If the producer session ends, the user is prompted to authenticate and start a new session the next time the user accesses resources at the producer or consumer.

During an active portal session, the user logs in at the producer, and the producer keeps track of the session, storing session information with the session server. After a producer session is established, the producer's maximum and idle timeout settings, configured in the Policy Server, determine the session duration.

Though the producer dictates the session duration, the consumer receives the session information from the producer in the SAML assertion and sets a local cookie containing this information so it can maintain an independent session from the producer.

It is possible for the session to expire at the consumer but remain active at the producer. If the session expires at the consumer, the consumer automatically redirects the user's browser back to the producer site to update the user's session information.

To enable an active portal session, you must enable the RequireActivePortalSession element. With this element enabled, the SAML Affiliate Agent collects the session information returned by the producer in the assertion, and uses this information to issue local user profile and session cookies.

The session information in the assertion includes:

- Starting time of the session—when the user first authenticated at the portal.
- Session ID
- Session idle timeout—the maximum time that can elapse between requests before the session is invalid.
- The total number of seconds a session is valid.

Configure an Active Portal Session

To configure an active portal session at the producer

There is no specific setting to configure an active portal session model. You configure assertions as you would normally.

To configure an active portal session at the consumer

1. Modify the AffiliateConfig.xml configuration file:
   - <SessionModel>Default</SessionModel>
   - RequireActivePortalSession="yes"
   - AllowUnknownUsers="no"
2. Check that the AllowUnknownUsers element is set to no. This setting must be no because the RequireActivePortalSession set to yes in the previous step.

   Unknown users do not get a session cookie at the producer.

Shared Sessions

The shared session model lets a consumer to share a single session with the producer. This provides users with a seamless session when they access resources at the producer and consumer because information about the users' activities and log outs is transmitted between the producer and the consumer.

With a shared session model, a central session for the user is created and stored at the producer. The consumer interacts with the producer to determine whether the session is valid or whether it has to be terminated. For the consumer and the producer to share a session, the consumer must periodically go back to the producer to check that the session is active. The frequency that the consumer checks for an active session is configurable and depends on the security relationship between the producer and consumer.

When shared sessions are used, sessions on the affiliate expire after all of the following time intervals elapse:

- Sync Interval
- Idle timeout

Shared sessions offer the following benefits:

Distributed logout

Allows users to logout from any consumer or the producer. If a user logs out at one consumer but is still valid at other consumers, the user's session at the producer is terminated. When the user visits other consumers, the user is rechallenged for credentials. This establishes an additional level of security across domains in a federated network.

Distributed timeout

Allows access to a resource at the consumer or the producer and the session remains active until you exceed the MaxTimeoutEnabled value set at the producer. This value is set in a realm's session properties.

Each time the consumer checks for an active session, SiteMinder updates the time stamp at the producer so the producer does not log the user out. The session server checks the timestamp and terminates any sessions that have expired. This improves the user experience because they do not have to reauthenticate often.

If the producer tells the consumer that the session is not valid, then the user must log back in and be reauthenticated.

Persistent sessions are validated by the Policy Server only when the Validation Period (Drift) value is greater than zero but less than the value of the idle timeout period. The Web Agent will not redirect the user to the idle timeout URL. The user will be re-challenged without being re-directed to the idle timeout URL.

The Role of the Session Server

When a user accesses an consumer resource during a shared session, authoritative session state for the user is maintained in the session server, provided that the realm where the user logged in at the producer is configured for persistent sessions.

Note: Successful federated communication requires that realms at the producer be configured for persistent sessions.

If the user does not log in at the producer prior to visiting the consumer, they are directed to the AuthenticationURL at the producer, which should always be part of a realm configured for a persistent session. If a realm is not set up for a persistent session, session information is stored in a session cookie instead of the session server.

Note: For protecting the Authentication URL, see the *SiteMinder Federation Security Services Guide*

In addition to the session server, the producer and consumer continue to issue and use session cookies to maintain local state. This avoids checking with the session server every time a user accesses a resource; however, the session information between the consumer cookies and the session server needs to be synchronized.

The sync interval defines the frequency, while the user is active at the consumer, at which the SAML Affiliate Agent contacts the producer to validate session status. This setting, configured at the producer, synchronizes the information in the session server and the information in the consumer cookies.

Set the Producer's Sync Interval for Shared Sessions

The sync interval defines the frequency, while the user is active at the consumer, with which the SAML Affiliate Agent contacts the producer to validate session status. This setting is configured at the producer; the default sync interval is 30 seconds.

For example, if the sync interval is 50 seconds, the session cookies at the consumer are out of sync with the producer session server by 50 seconds. If the user logs out at the producer at 4:00 PM, the consumer waits until 4:00:50 PM until it checks the producer site again to see if the user has an active session.

Three considerations affect the value of Sync Interval:

- If the value of Sync Interval is too small, the SAML Affiliate Agent keeps calling the session server, which slows down SiteMinder's performance when processing requests.
- The value must not larger than half the lowest Idle Timeout Enabled value that is set for the realm where the user logs in at the producer before going to the consumer. The Idle Timeout Enabled field is part of a realm's session configuration parameters.
- No session at the producer times out early when the user is still active at the consumer. The user is only logged out if they try to access a protected consumer resource after a session is terminated.

Configure Shared Sessioning

To configure shared sessioning at the producer:

To configure shared sessioning, you have to modify the Session properties for the consumer in the Policy Server User Interface.

To configure shared sessioning at the consumer:

Modify the AffiliateConfig.xml configuration file as follows:

<SessionModel>Shared</SessionModel>

AllowUnknownUsers="no"

Note: If you configure a Shared session model, the SAML Affiliate Agent ignores the RequireActivePortalSession attribute.

Response Attributes and Default HTTP Headers Overview

Response attributes and default HTTP headers instruct applications how to collect user data and apply that information to display personalized content for each user.

The following mechanisms are available for developing Web applications:

- Configurable response attributes

  Response attributes are sent by the Policy Server in SAML assertions to the SAML Affiliate Agent to be included in HTTP headers or cookies. These are configurable by an administrator using the Policy Server User Interface.

- Default HTTP headers

  The SAML Affiliate Agent has a set of default HTTP headers that it passes to Web applications.

More Information

Response Attributes to Personalize Content (see page 117)
Use SAML Affiliate Agent Default HTTP Headers (see page 119)

Log Users Out from a Session

The LogOffURI attribute enables the SAML Affiliate Agent to log a user out of a session. This attribute must be set to a single URI that is present on the affiliate Web server. When a user logs out at the consumer, the SAML Affiliate Agent removes the consumer profile and session cookies and, in the case of a shared session, sends a logoff notification to the producer session server indicating that the session has ended at the consumer.

If you are using this feature for a Default session model, be aware of the following:

- The LogoffURI should be treated as a protected resource. To do this, include the URI in the AffiliateResource attribute.
- Because the consumer and producer maintain separate sessions in the Default model, users may be unaware that they have been logged out of an consumer session if the producer session is still active. If a user logs off at the consumer site, the SAML Affiliate Agent invalidates only the consumer session cookie. At the next request for a protected resource at the consumer, the user is sent back to the producer. If the producer session is still valid, the user receives new consumer cookies without being rechallenged by the producer.

  Comprehensive information about redirects between the consumer and producer are included in the SAML Affiliate Agent's log file.

To specify a LogOffURI, enter a URI. For example:

<LogoffURI>/protected/logoff.html</LogoffURI>

Note: This element can only be used with a default or shared session.

Configure the Notification Service

The notification service lets the SAML Affiliate Agent notify the producer of user activity at the consumer site. This service applies only to protected resources.

The notification service will be called when a user accesses a URL specified in the NotificationURL element. Event details are logged at the producer in the NotificationMessages.log file.

Three settings are required for the notification service—they are in different sections of the AffiliateConfig.xml file:

- NotificationService (PerPortalInfo section)—specifies the location of the notification service at the producer.
- NotificationURL (PerAffiliateInfo section)—designates URLs at the consumer that trigger notification messages to be sent to the producer. You can have one or more of these entries in the configuration file.
- MatchingRule for the NotificationURL (PerAffiliateInfo section)—indicates how the SAML Affiliate Agent compares each URL for which the user gained access to the NotificationURL. Required with the NotificationURL.

To configure the notification service

1. Specify a URL path for the NotificationService element. It must be a URL over an SSL connection. For example:

   <NotificationService>https://secure.producer.domain.com:81/affwebservices/router/notification</NotificationService>

2. Configure the NotificationURL element together with the MatchingRule attribute. For example:

   <NotificationURL MatchingRule="StrictPrefix">protected</NotificationURL>

3. Ensure that the resources you want to designate for notification service are added to the AffiliateResource setting.

More Information

Specify the Notification Service (see page 92)
Define the MatchingRule for the Notification Service (see page 115)
Designate Affiliate Resources (see page 97)

Define the MatchingRule for the Notification Service

To configure the notification service, you must indicate which URLs trigger a notification message to the producer when accessed by a user.

The NotificationURL is not a complete URL; it is only the resource portion of the URL after the server name. The SAML Affiliate Agent checks each consumer URL request against the NotificationURL.

The MatchingRule attribute, required as part of the NotificationURL, indicates how the SAML Affiliate Agent compares each URL, for which the user gained access, to the NotificationURL.

The two possible values for the matching rule are:

- StrictPrefix
- Substring

Use a StrictPrefix

For the StrictPrefix attribute, the value of the requested URL has to match only the beginning of the URL prefix, before the query string (?). For example, the entry:

<NotificationURL MatchingRule="StrictPrefix">protected</NotificationURL>

will match: http://www.affiliate.com/protected/...

but will not match: http://www.affiliate.com/basic?protected

Use a Substring Prefix

For the Substring attribute, the value of the requested URL can match before or after the query string (?). For example, the entry:

<NotificationURL MatchingRule="Substring">protected</NotificationURL>

will match the following:

http://www.affiliate.com/protected/...

http://www.affiliate.com/basic?protected

Chapter 8: Response Headers for Customizing Web Applications

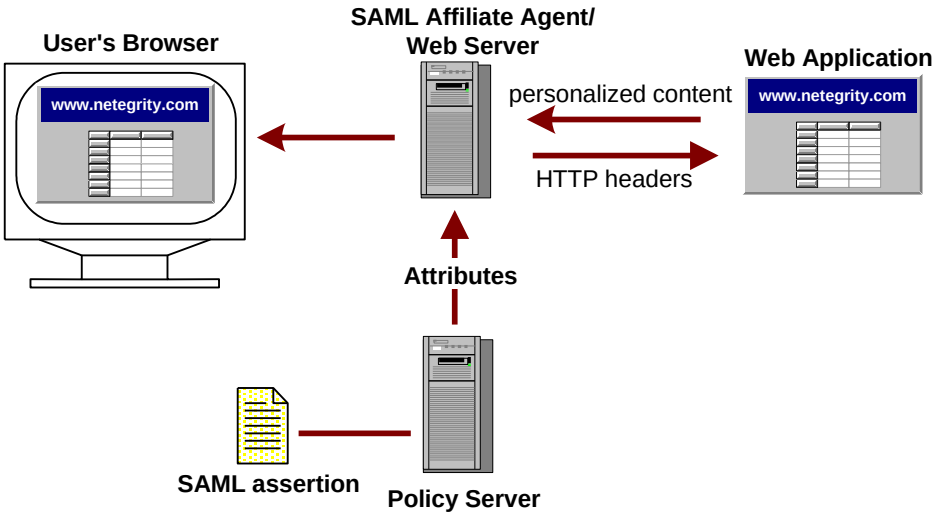
This section contains the following topics:

Response Attributes to Personalize Content

SiteMinder provides configurable response attributes as a means of delivering data to applications and customizing the user experience.

You configure responses using the Policy Server User Interface, and then associate them with a specific consumer. When the Policy Server sends an assertion to the SAML Affiliate Agent, it includes the response data. The SAML Affiliate Agent interprets the information and makes it available to Web applications.

The following illustration shows how responses personalize content.



The following table lists the SAML Affiliate Agent response attributes.

Affiliate Attribute Name	Description
Affiliate-HTTP-Cookie-Variable	Generates a SetCookie header, which then sets a non-persistent cookie in a Web browser. The cookies only exist in the cookie domain where the SAML Affiliate Agent is configured. You can enter multiple Affiliate-HTTP-Cookie-Variable attributes.
Affiliate-HTTP-Header Variable	Allows you to specify an arbitrary dynamic name/value pair for use by a Web application. You can enter multiple Affiliate-HTTP-Header-Variable attributes.

The HTTP-Header-Variable and HTTP-Cookie-Variable attributes enable a SAML Affiliate Agent to pass a static or dynamic list of name/value pairs to a Web application. The name/value pairs are specific to the user requesting a resource, which enables the application to customize what the user sees.

For example, an administrator configures the Affiliate-HTTP-Header-Variable response attribute to store the full name of the user. When the user is authorized to access the protected resource, the SAML Affiliate Agent passes the user's full name to the Web application. The user's name is then displayed by the application, which helps to establish a relationship with the customer.

Be aware that in a Web application environment, the HTTP-Header-Variable response attribute appears as an HTTP_*attribute_name* variable, where *attribute_name* is the name of the HTTP variable, for example USERFULLNAME. You do not have to have an underscore in the name as the underscores cause problems with some application servers.

Note: The server may convert any dash (-) in the attribute name to an underscore (_), and all alphabetic characters to uppercase.

Configure Affiliate Response Attributes

You configure affiliate response attributes in the Policy Server User Interface. Response attributes are one of the properties that you configure as part of the Affiliate Properties that you define when configuring an consumer in a consumer domain.

Note: For detailed information about configuring response attributes, see *SiteMinder Federation Security Services Guide*.

When the Policy Server issues a SAML assertion as part of the process to authenticate a user requesting a resource at the consumer, it includes the response attributes in the assertion it sends to the SAML Affiliate Agent.

Note: When configuring response attributes, note that the maximum buffer size for the assertion if 4KB.

Use SAML Affiliate Agent Default HTTP Headers

As part of the Web application environment, the SAML Affiliate Agent submits default HTTP headers to the Web server, which in turn, makes them available for Web applications. You can use these headers to include functions and enable your Web applications to personalize content. Headers can store information such as a user's name and the type of action a user is authorized to perform.

The Agent sends these headers regardless of whether or not they are called from a Web application.

The following table lists the default HTTP headers issued by the SAML Affiliate Agent.

Default HTTP Header	Description
HTTP_SMCOMPANYSOURCEID	The company source ID specified in the AffiliateConfig.xml file, such as b818452610a0ea431bff69dd346aeeff83.
HTTP_SMAUTHORIZED	Confirms that the user is authorized. The Agent includes this header only if the user is authorized. If the user is not authorized or if the resource is unprotected, this header is not returned.
HTTP_SMUSER	Login name of the authenticated user, such as user1. If a user does not provide a user name at log in, such as certificate-based authentication, then this header is not set.
HTTP_SMUSERDN	The user's distinguished name recognized by SiteMinder. For example, uid=user1,ou=People,o=netegrity.com

Chapter 9: Upgrade to SAML Affiliate Agent v6.x QMR 6

This section contains the following topics:

Upgrade a 5.x or 6.0 SAML Affiliate Agent to 6.x QMR 6

The following are Affiliate Agents you can upgrade:

- 4.x Affiliate Agent
- 5.5 SAML Affiliate Agent
- 5.5 QMR 1, QMR 2, QMR 3, QMR 4 SAML Affiliate Agent
- 6.0 SAML Affiliate Agent
- 6.x QMR 1, QMR 2, QMR 3, QMR 4, QMR 5 SAML Affiliate Agent

You can upgrade these releases even if you have applied a hotfix.

When you upgrade from a pre-6.x QMR 6 SAML Affiliate Agent, the program lays out the SAML Affiliate Agent files again without altering your existing configuration.

When the upgrade is complete, only one AffiliateConfig.xml file exists, located in the directory *saml_affiliate_agent_home*/config. The AffiliateConfig.xml file is a shared configuration file for all web servers on the system. The SAML Affiliate Agent can operate with this one file; however, to establish independent AffiliateConfig.xml files for different web servers, run the Configuration Wizard. The Configuration Wizard enables you to modify the configuration and generate a configuration file on a per-web server basis.

To upgrade the SAML Affiliate Agent, the Agent must be compatible with the Policy Server and Web Agent at the producer site. The 6.x SAML Affiliate Agent is compatible with the 6.x Policy Server and 6.x Web Agent. Therefore, an upgrade from a 5.x SAML Affiliate Agent to 6.x also requires an upgrade of the 5.x Policy Server and Web Agent to 6.x.

Note: SAML Affiliate Agent v6.x QMR 6 is not compatible with a SiteMinder 5.x producer.

The following are the main steps for upgrading any 5.5, 5.x QMR x, and r6.x SAML Affiliate Agent to a r6.x QMR 6 SAML Affiliate Agent:

1. Plan a recovery strategy.
2. Uninstall the 5.x/6.x SAML Affiliate Agent.
3. Upgrade to the 6.x QMR 6 SAML Affiliate Agent.
4. Modify the configuration of the upgraded Agent.

These procedures refer to the latest versions of SiteMinder products. For other compatible versions, go to Technical Support and search for SiteMinder Platform Matrix.

Complete the following prerequisites for upgrading a 5.x/6.x SAML Affiliate Agent to 6.x QMR 6.

- On the system that you are upgrading, prepare two web server instances:
  - one non-SSL server
  - one SSL server (for back channel communication)

  Note: Instead of using two server instances, you can use one server instance with two ports (non-SSL and SSL). IIS and Sun ONE 6.0 servers support this type of configuration.
- Verify that a JRE is installed. To verify the supported version, go to Technical Support and search for SiteMinder Platform Matrix.
- Be sure that you have access to the 6.x SiteMinder documentation. You can refer to these guides for additional information.
- Make copies of the AffiliateConfig.xml file, the affiliateserverconf.properties file, and the AM.keystore. Information from these files can be used to configure the upgraded SAML Affiliate Agent.
- As part of the upgrade procedure, supply URL and domain values for the producer and consumer. Use the values from the existing AffiliateConfig.xml file to respond to these prompts. The following information is required:
  - Affiliate Cookie Domain—domain at the consumer for the local server where the SAML Affiliate Agent is installed.
  - SSL Interceptor Root URL—consumer-site URL that enables the SAML Affiliate Agent to receive the SAML artifact. This URL is the secure web server at the consumer site where the SAML Affiliate Agent is installed.
  - Federation Web Services Root URL—secure URL to the producer web server where the Web Agent Option Pack is installed.

  Note: You are prompted for two producer-side URLs, but you are not required to change the configuration at the producer when performing the upgrade.

Step 1: Plan a Recovery Strategy

Before upgrading the SAML Affiliate Agent, implement a recovery plan that lets you return to your original configuration if the upgrade fails.

The best strategy is to make copies of the AffiliateConfig.xml file, the affiliateserverconf.properties file, and the AM.keystore to have as a back up.

Note: For more information, see the *SiteMinder Upgrade Guide*.

Step 2: Uninstall the 5.x/6.0 SAML Affiliate Agent

Important! Keep copies of the older configuration files for reference during the upgrade.

Uninstall the existing SAML Affiliate Agent before installing the newer version.

More Information

Uninstall the SAML Affiliate Agent (see page 51)

Step 3: Upgrade to the 6.x QMR 6 SAML Affiliate Agent

When you upgrade from a pre-6.x QMR 5 SAML Affiliate Agent, the program lays out the SAML Affiliate Agent files again without altering your existing configuration.

When the upgrade is complete, only one AffiliateConfig.xml file exists, located in the directory *saml_affiliate_agent_home*/config. The AffiliateConfig.xml file is a shared configuration file for all web servers on the system. The SAML Affiliate Agent can operate with this one file; however, to establish independent AffiliateConfig.xml files for different web servers, run the Configuration Wizard. The Configuration Wizard enables you to modify the configuration and generate a configuration file on a per-Web server basis.

The upgrade instructions that follow reflect the GUI mode procedures. For UNIX systems, you can upgrade using Console mode by executing the SAML Affiliate Agent binary file with the -i console command argument, for example, nete-af-6xqmr6-*operating_system*.bin -i console. The command-line upgrade prompts are similar to GUI mode prompts.

To upgrade to the 6.x QMR 6 SAML Affiliate Agent

1. Stop all web servers and web server applications.
2. Download the software from the CA Technical Support site.

3. Complete one of the following steps:

   Windows: Navigate to the win32 folder then double-click nete-af-*version*-win32.exe.

   Solaris: From the solaris folder, copy nete-af-*version*-sol.bin to a local directory, navigate to that directory and enter the following command in a console window:

   ./nete-af-*version*-sol.bin

   Linux 2.1 systems: From the linux folder, copy nete-af-*version*-linux.bin to a local directory, navigate to that directory and enter the following command in a console window:

   ./nete-af-6qmr6-linux.bin

   The setup program prepares the files for installation.

4. In the Introduction dialog, read the information then click Next.

5. Read the License Agreement and select the option to accept the agreement. Click Next.

   If you do not accept the agreement, the installation terminates.

6. Read the Release Notes, then click Next.

7. In the Choose Install Folder dialog, accept the default installation location or use the Browse button to select a different location. Click Next.

8. In the Web Server dialog, select two web server instances:

   - one non-SSL server
   - one SSL server

   You can use one web server with two ports (non-SSL and SSL). IIS and Sun ONE 6.0 support this type of configuration.

   Follow the steps for your web server type:

   IIS Web Server

   Select the IIS Web server and click Next.

   Apache Web Server

   Select the Apache Web server and enter the full path to the server location, for example, /usr/local/apache2. Click Next.

Sun ONE Web Server

a. Do one of the following:

   For Windows systems, the installation detects the Sun ONE web server automatically.

   For UNIX systems, select the Sun ONE Web server and enter the full path to the root directory of the server, for example, usr/sunone/servers.

b. Click Next.

c. Select the instance of the Sun ONE Web server on which the SAML Affiliate Agent will run.

d. Click Next.

9. If prompted, specify the location of the JRE by accepting the default location or by using the Browse button to select a different location.

   Note: For the supported JRE version, search the SiteMinder Platform Matrix on the Technical Support site.

10. Respond to the following configuration prompts then click Next:

    Important! When prompted to enter a root URL, enter it in the form http://address.domain.com:port—do not enter any additional text.

    When you specify a value for a root URL, the installation script appends additional information to it in the AffiliateConfig.xml file. For example, if you enter https://interceptor.domain.com:90 for the SSL Interceptor Root URL, the script appends /smafa/amts/test1.htm to it.

    Affiliate Cookie Domain

    Enter the domain for the local server where the SAML Affiliate Agent is installed, such as .partner.com.

    SSL Interceptor Root URL

    Enter the URL at the consumer site where the producer redirects users during consumer requests. The URL points to the secure web server at the consumer where the SAML Affiliate Agent is installed. We recommend that you use an SSL connection, and begin the URL with https://, such as

    https://affiliatesslserver.partner.com:90

    The SSLInterceptorURL enables the SAML Affiliate Agent to obtain the SAML artifact. The artifact identifies the SAML assertion stored at the producer. The assertion contains user profile and session information. After the Affiliate Agent gets the artifact, it calls the producer over the SSL back channel to retrieve the assertion.

Federation Web Services Root URL

Enter the URL for the web server at the producer where the Web Agent Option Pack is installed. The URL must be a secure URL that begins in the form https://, such as

https://myproducer.ca.com:81

11. In the Passwords dialog, complete the following:
    a. Enter the Shared Secret twice. The SAML Affiliate Agent uses this secret to encrypt consumer cookies.

       This secret is used locally to encrypt consumer cookies. You do not have to specify a corresponding secret at the Policy Server.

    b. Enter the Affiliate Password twice. The SAML Affiliate Agent uses the password to communicate with the Policy Server at the producer site. The password must match the password for a consumer defined in the Policy Server User Interface.

       Note: For information about configuring a consumer, see the *Federation Security Services Guide*.
12. Optionally, respond to the prompt about optional UNIX configuration. If you are using the Bourne shell, include the nete-af-env.sh environment variable in the .profile file.
13. In the Pre-Installation Summary dialog, confirm the configuration settings then select Install.

    The setup program copies files to the specified location. Afterward, the Setup Complete dialog is displayed.
14. In the Installation Complete dialog, click Finish.
15. Restart the web server.

Step 4: Modify the Configuration of the Upgraded Agent

In addition to the configuration settings that you specify during the upgrade, you can transfer other settings from the 5.x/6.0 SAML Affiliate Agent to the 6.x QMR 6 Agent.

Use the 5.x AffiliateConfig.xml file as a reference to modify the 6.x QMR 6 file manually.

Important! Do not copy the 5.x AffiliateConfig.xml file and use that file for the 6.x QMR 6 SAML Affiliate Agent. The file structures are different.

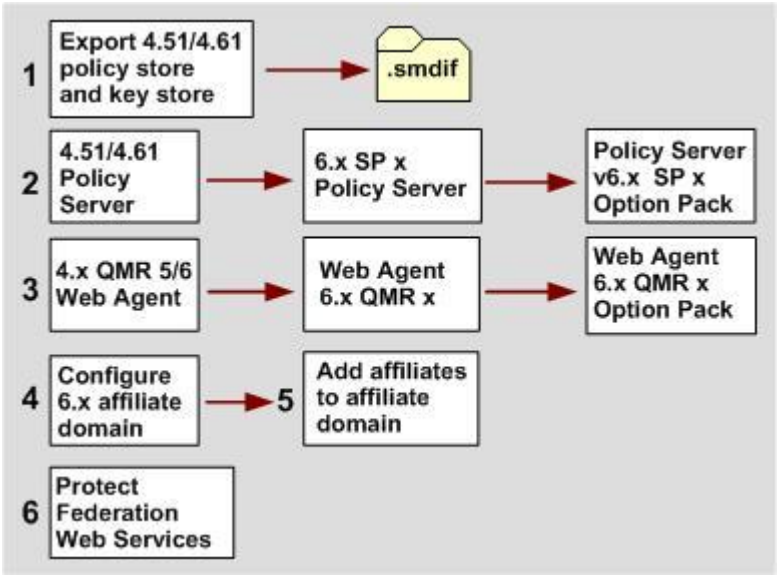
Upgrade a 4.51/4.61 Affiliate Agent to a 6.x SAML Affiliate Agent

Migrating a 4.51/4.61 federated environment to a 6.x environment involves tasks at the producer site, where the Policy Server and Web Agent reside, and at the consumer site, where the SAML Affiliate Agent resides.

No script automatically upgrades a 4.51/4.61 Affiliate Agent to the 6.x QMR 6 SAML Affiliate Agent—you must follow the upgrade procedure.

Note: You can only upgrade a 4.51/4.61 Affiliate Agent to a SAML Affiliate Agent on web server platforms that support the SAML Affiliate Agent. For example, if the 4.x Affiliate Agent is installed on an iPlanet 4.1 server running Solaris, it cannot be upgraded to a SAML Affiliate Agent on that same platform.

The following illustration shows the upgrade tasks required at the producer site:

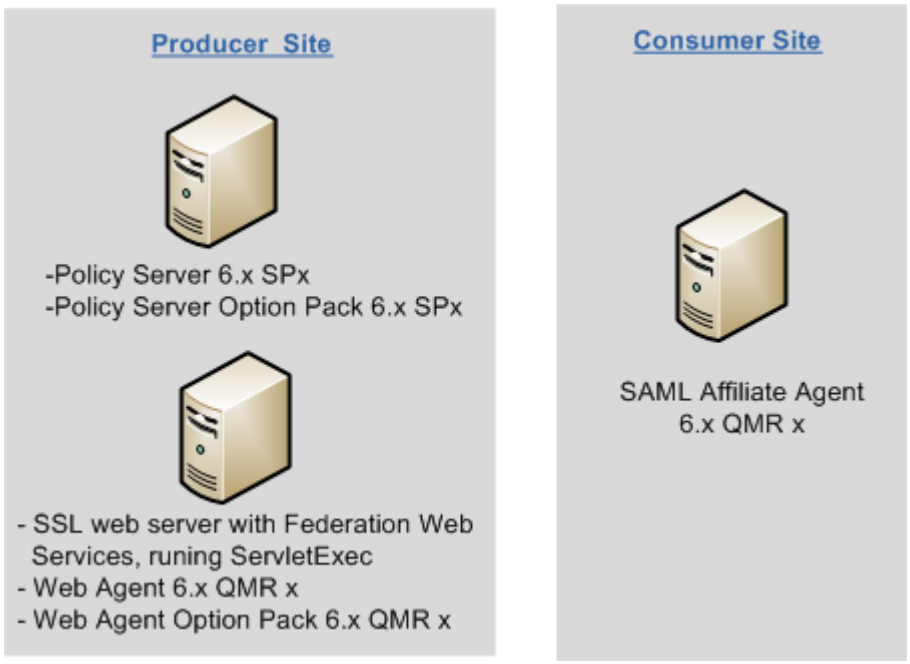


The following upgrade tasks are required at the consumer site:

1. Copy the 4.x Affiliate.conf file.
2. Install the SAML Affiliate Agent v6.x QMR x.
3. Transfer the 4.x affiliate configuration to the SAML Affiliate Agent.
4. Enable the SAML Affiliate Agent.

During an upgrade you must take each component offline, perform the upgrade, then bring it back online. Each component is unavailable to the others during the migration.

When the upgrade is complete, the components of the federated network are similar to the following illustration.



Important! We strongly recommend reading through the entire upgrade procedure before you upgrade.

Plan a Recovery Strategy

Before upgrading the Policy Server and Agents, implement a recovery plan that lets you return to your original configuration if the migration fails.

The most complete recovery plan is to back up each machine's entire image—Policy Servers, Web Agents, and Affiliate Agents. We strongly recommend this method.

Note: For more information, see the *SiteMinder Upgrade Guide*.

Perform an Upgrade at the Producer Site

At the producer site, upgrading a 4.51/4.61 Affiliate Agent to a 6.x SAML Affiliate Agent involves the following tasks:

1. Exporting the 4.51/4.61 policy store and key store.
2. Upgrading the 4.51/4.61 Policy Server and Option Pack to 6.0 SP6.
3. Upgrading the 4.x QMR 5/6 Web Agent to 6.x QMR 6.

4. Defining Affiliate domains.
5. Adding an affiliate to an affiliate domain.
6. Protecting Federation Web Services.
7. (Optional) Deleting the 4.x policy domains.

Step 1: Export the 4.51/4.61 Policy Store and Key Store

At the producer site, export the policy store to an .smdif file using the 4.51/4.61 version of smobjexport. The policy store includes all affiliate policy domain and key store information. Later, you import this .smdif file to the 6.x policy store.

You can export data in encrypted or clear text. If the 6.x Policy Server uses the same encryption key as the 4.51/4.61 Policy Server, do not export the data stores in clear-text. Using clear-text is necessary only if the 4.51/4.61 and 6.x Policy Servers use different encryption keys and it gives you a record of encrypted information, such as shared secrets.

Important! Be aware that if you export data in clear-text, you may expose sensitive information, such as passwords or shared secrets.

If your key store resides in the policy store, use the -k option with smobjexport because keys are not included in the export by default. To preserve your policy store, enter the following command:

smobjexport -o*file_name* -d*admin_name* -w*admin_pw* -k -c -v -t

Argument	Meaning
file_name	4.51/4.61 smdif file
admin_name	SiteMinder administrator's user name
admin_pw	SiteMinder administrator's password
-k	Exports Agent keys stored in the policy store. By default, keys are not included in the export.
-c (optional)	Exports sensitive data as clear-text. Using clear-text is necessary only if the 4.51/4.61 and 6.x Policy Servers use different encryption keys
-v	Enables verbose mode
-t	Enables low level tracing mode to troubleshoot the import process

If an argument contains spaces, use double quotes around the entire argument. For example, if the SiteMinder administrator is SM Admin, the argument would be -d"SM Admin". For example:

smobjexport -opstore.smdif -d"SM Admin" -wpassword -k -c -v -t

Note: For more information on using the 4.51/4.61 version of smobjexport, see the *Policy Server Installation Guide* for v4.5 or v4.6.

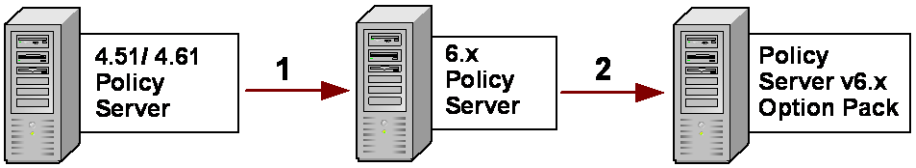
Step 2: Upgrade v4.51/4.61 Policy Server and Option Pack to v6.0 SP 6

At the producer site, upgrade a 4.51/4.61 Policy Server and its Option Pack to version 6.0 SP 6.

Upgrade the Policy Server first followed by the Web Agent to maintain single sign-on and failover.

Important: Upgrade the Policy Server before upgrading the Web Agent. For instructions, see the *SiteMinder Upgrade Guide*.

The following illustration shows the upgrade path.



1. Upgrade the 4.51/4.61 Policy Server to 6.x.

   As part of the Policy Server upgrade procedure, import the .smdif file, which was generated when you exported the 4.51/4.61 policy store, to the 6.x policy and key store.

   Use the smobjimport utility to import the file. The command syntax is:

   smobjimport -i*file_name* -d*admin_name* -w*admin_pw* -4 -k -c -v -t

   Example:

   smobjimport -ipstore.smdif -d"SM Admin" -wpassword -k -c -v -t

   Note: For instructions on upgrading the 4.51/4.61 Policy Server to 6.x and using the smobjimport utility, see the *SiteMinder Upgrade Guide*.

2. Install the Policy Server Option Pack for 6.x.

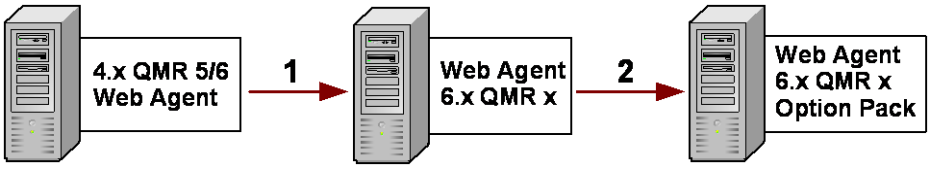
   The Policy Server Option Pack installs federation features on the Policy Server.

   Note: For instructions, see the *Policy Server and Web Agent Option Pack Guide*.

Step 3: Upgrade the 4.x QMR 5/6 Web Agent to 6.x QMR 6

At the producer site, upgrade a 4.x QMR 5/6 Web Agent to 6.x QMR 6.

The following figure shows the upgrade path.



1. Upgrade the 4.x QMR 5/6 Web Agent to 6.x QMR 6.

   Note: For instructions, see the *Web Agent Installation Guide.*

2. Install the Web Agent Option Pack, v6.x QMR 6.

   The Web Agent Option Pack installs the Federation Web Services application on the Web Agent.

   Note: For instructions, see the *Policy Server and Web Agent Option Pack Guide.*

Step 4: Define Affiliate Domains at the Policy Server

At the producer, transfer the 4.51/4.61 affiliate information into 6.x affiliate policy objects for the Policy Server using the Policy Server User Interface.

When you import 4.51/4.61 affiliates into the 6.x policy store, they appear as a policy domain. You should see a 4.51/4.61 Affiliate Agent policy domain object in the System tab of the Policy Server User Interface.

In the 6.x affiliate model, an affiliate is not associated with a policy domain—it must be associated with an affiliate domain, so you need to create a new affiliate domain for each 4.51/4.61 policy domain.

Note: If you had multiple affiliates within multiple 4.51/4.61 policy domains, you may want to create multiple affiliate domains that correspond to each policy domain.

In the Policy Server User Interface:

1. Create a new affiliate domain.
2. Enter the relevant 4.51/4.61 information in the appropriate fields in the affiliate domain properties, as shown in the following table.

   Note: For step-by-step instructions on creating affiliate domains, see the *Federation Security Services Guide*.

4.51/4.61 Affiliate Property	Sample Value	Where to Set Value in the 6.x Policy Server User Interface
Affiliate Name	testagent	Create a domain and enter the following: Name: testagent Description: Affiliate Agent in test environment. Select Affiliate Domain radio button.
Description	Affiliate Agent in test environment	
User Directories	userdir-test1	In the Domain Properties for the affiliate domain: Select the User Directories tab and add the userdir-test1 directory.
Administrators	SiteMinder	In the Domain Properties for the affiliate domain: Select the Administrators tab and ensure SiteMinder is added.

Step 6: Protect Federation Web Services

The Federation Web Services application installed at the producer-side Web Agent is protected by SiteMinder policies. When you import the ampolicy.smdif file during the installation of the Policy Server Option Pack, most of these policies and related policy objects are automatically created.

For these policies, you must:

- Add a Web Agent to the appropriate Agent Group to enforce protection.
- Allow affiliates access to Federation Web Services by including the affiliates as users within each federation policy.

Federation Web Services policies do not protect resources at the consumer site.

Note: For information about protecting Federation Web Services, see the *Federation Security Services Guide*.

Step 7: Delete 4.x Policy Domains (Optional)

When you have tested the 6.x affiliate configuration, you can delete the old 4.51/4.61 policy domains.

Upgrade the 4.51/4.61 Affiliate at the Consumer Site

Important! Upgrade the Affiliate Agent in a test environment first and ensure the configuration is tested before putting the new SAML Affiliate Agent into a production environment.

At the consumer site, upgrade a 4.51/4.61Affiliate Agent by completing the following tasks:

1. Make a copy of the 4.51/4.61 Affiliate Agent configuration file.
2. Install the 6.x QMR 6 SAML Affiliate Agent.
3. Transfer the 4.51/4.61 Affiliate Agent configuration to 6.x QMR 6 SAML Affiliate Agent.
4. Enable the 6.x QMR 6 SAML Affiliate Agent.

Step 1: Copy the 4.51/4.61 Affiliate Agent Configuration File

The Affiliate.conf file is the configuration file for the 4.51/4.61 Affiliate Agent. You need the configuration information from this file to configure the SAML Affiliate Agent.

Important! Copy of the configuration file to use as a reference.

The Affiliate.conf file resides in the following default locations:

- IIS:

  C:\Program Files\Netegrity\SiteMinder Affiliate Agent\Bin

- Sun ONE:
  - UNIX: *Sun_Java_System_home*/https-hostname/config/
  - NT: *Sun_Java_System_home*\https-hostname\config\
- Apache:

  /*apache_home*/conf

Step 2: Install the SAML Affiliate Agent v6.x QMR 6

To install the SAML Affiliate Agent 6.x QMR 6

1. Stop the web server where the 4.51/4.61 Affiliate Agent resides.
2. Uninstall the 4.51/4.61 Affiliate Agent.

   Note: For instructions, see the *4.x SiteMinder Installation Guide*.
3. Ensure the web server where the SAML Affiliate Agent will be installed accept HTTPS and HTTP connections.
4. Install the SAML Affiliate Agent.

   You are prompted for certain URLs and an affiliate domain during the installation. For two of the settings, use existing 4.x affiliate data, as shown in the table following these steps. Note that the values are examples.
5. Optionally, if you are using a root certificate authority that is not listed in your AM.keystore file, you need to add it to the key store. Use the Java keytool utility included with the JDK.

4.x Affiliate.conf Setting	**6.x AffiliateConfig.xml Setting**
portal="http://www.netegrity.com/siteminderagent/smProfile.ccc"	<PortalQueryURL>https://secure.portal.domain.com:81/affwebservices/public/intersitetransfer/</PortalQueryURL>
cookiedomain=".netegrity.com"	<CookieDomain>.netegrity.com</CookieDomain>
Settings Exclusive to 6.x	
Federation Services Root URL These parameters use this URL: ■ <GetAssertionService> ■ <SessionProviderService> ■ <NotificationService>	https://myserver.netegrity.com:81
SSL Interceptor URL (specify a port for this parameter)	<SSLInterceptorURL>https://interceptor.domain.com:90/affiliateagent/afftsite/test1.htm</SSLInterceptorURL>

More Information

Install a SAML Affiliate Agent (see page 27)
Add Certificate Authorities to the Affiliate Key Store (see page 161)

Step 3: Transfer the 4.51/4.61 Affiliate Configuration

Transfer the configuration settings from the 4.51/4.61 Affiliate.conf file to the 6.x AffiliateConfig.xml file according to the following table.

4.x Affiliate.conf Setting	**AffiliateConfig.xml Setting**
	GlobalInfo Section
logconsole="NO"	LogToConsole="no"
loglevel="0"	<LogLevel>0</LogLevel>
logfilename="../logs/affiliate.log"	<LogFile>C:\Program Files\netegrity\affiliateagent\log\ affiliate.log</LogFile>
logfile="NO"	LogToFile="no"
logappend="NO"	LogSettings Append="no"
ignoreexts=".class, .gif, .jpg, .jpeg, .png, .fcc, .scc, .ccc"	<IgnoreExtensions>.gif .jpeg .png .ccc</IgnoreExtensions>
	PerPortalInfo Section
portal="http://www.netegrity.com/siteMinderagent/smProfile.ccc"	<PortalQueryURL>https://secure.portal.domain.com:81/ affwebservices/public/intersitetransfer/</PortalQueryURL>
	PerAffiliateInfo Section
affiliateresource="/realmA"	<AffiliateResource MatchingRule="StrictPrefix">/realmA </AffiliateResource>
noaccessurl="http://localhost/unprotected/bad.htm"	<NoAccessURL AppendTarget="yes">http://localhost/ unprotected/bad.htm</NoAccessURL>
allowunknownusers="NO"	AllowUnknownUsers="no"
noaccessurl="http://localhost/noaccess/noaccess.html"	<NoAccessURL AppendTarget="no">http://localhost/ unprotected/bad.htm</NoAccessURL>
affiliatetimeout="0"	<SessionMaxTimeout>0</SessionMaxTimeout>
noaccesstarget="NO"	<NoAccessURL AppendTarget="no">
cookiedomain=".netegrity.com"	<CookieDomain>.netegrity.com</CookieDomain>
affiliatename="sampleaffiliate"	<AffiliateName>SampleAffiliate</AffiliateName>

4.x Affiliate.conf Setting	AffiliateConfig.xml Setting
requireactiveportalsession="NO"	RequireActivePortalSession="no"

More Information

4.x Affiliate.conf Parameter Mappings (see page 177)

Step 4: Enable the SAML Affiliate Agent

1. Open the AffiliateConfig.xml file.
2. Go to the PerAffiliateInfo section for the affiliate.
3. Set the Enabled attribute to yes.

More Information

Enable and Disable the SAML Affiliate Agent (see page 95)

Step 5: Add an Affiliate to an Affiliate Domain

The producer must be able to identify each affiliate, so you have to add each affiliate to an affiliate domain.

To add an affiliate to a domain

1. Log on to the Policy Server User Interface.
2. Add an affiliate to the affiliate domain.

   Assign an affiliate name and password, and an AuthenticationURL at the minimum for a 6.x QMR 6 affiliate model. Note also that there are additional items to configure, such as notifications, assertions and sessions.

   Note: For step-by-step instructions on configuring affiliates, see the *Federation Security Services Guide*.
3. Transfer users from the 4.x configuration that have access to affiliate domains to users with access to affiliate domains for the 6.x QMR 6 configuration as follows:

   From the 6.x Policy Server User Interface:

   a. Select an affiliate

      The Affiliate Properties dialog displays.

   b. Select the Users tab.

   c. Click Add/Remove.

   d. Select the appropriate users and add them.

For example, for the 4.51/4.61 Affiliate Property **Users**, sample values would be:

ou=People

o=test.com

ou=JPeople

ou=People

o=test.com

4. Transfer the 4.x affiliate attributes to the 6.x QMR 6 affiliate configuration. The information you have to enter is listed in the following table.

4.51/4.61 Affiliate Property	**Sample Value**	**Where to Set Value in the 6.x Policy Server User Interface**
Response Attribute	Affiliate-HTTP-Header-Variable	Select the affiliate. In the Affiliate Properties dialog: 1. Select the Attributes tab. 2. Click Create. The Affiliate Attribute dialog opens. 3. In the Attribute field, select Affiliate-HTTP-Header-Variable 4. In the Attribute Fields group box, enter: ■ Variable Name: HEADER_DATA ■ Variable Value: affiliate1 5. In the Attribute Kind group box, select Static. SiteMinder administrator user name SiteMinder administrator password
Response Attribute Variable Name	HEADER_DATA	
Response Attribute Variable Value	affiliate1	
Response Attribute Kind	static	

Chapter 10: Extend a SAML Affiliate Agent

This section contains the following topics:

SAML Affiliate Agent Extensions Overview

The capabilities of the SAML Affiliate Agent can be extended using the two C language APIs included with Federation Services installed at the producer.

The APIs include the following:

- Persistence API—Integrates custom assertion management routines with SAML Affiliate Agent operations.

  Use this API to implement custom persistence libraries.

- Key Provider API—Provides the SAML Affiliate Agent with different encryption keys on a periodic basis. The Agent uses these keys to encrypt and decrypt its cookies.

  Use this API to implement custom key provider libraries.

You can compile your custom libraries to Solaris .so files or Windows .dll files. The libraries are dynamically linked to the SAML Affiliate Agent at runtime.

Persistence API

When a user logs into the producer site and wants to access a protected resource on the consumer site, the SAML Affiliate Agent "pulls" the user's entitlement information from the producer in the form of an XML assertion.

Typically, the Agent is responsible for parsing the raw XML assertion data into entitlement name/value pairs and storing this information in the SMPROFILE cookie. The SAML Affiliate Agent then makes these name/value pairs available to the user's application via HTTP headers or cookies.

However, if you want to control the way XML assertions are stored, parsed, and retrieved, you can write a library that performs these custom operations, and then use the Persistence API to integrate your custom assertion management routines with Agent operations.

Sequence of Events

The following events occur when a persistence library is used to perform custom assertion management:

1. A user makes an initial request for a resource at the consumer site.
2. The SAML Affiliate Agent pulls the raw XML assertion data from the producer, and then passes the unparsed XML assertion to the persistence library in a call to PersistAssertion().
3. Only profile information is sent. The SAML Affiliate Agent has removed the security information.
4. The persistence library parses the assertion into entitlement name/value pairs and stores the entitlement data.
5. The library also sends a lookup key to the Affiliate Agent in the output parameter of PersistAssertion(). Applications use the lookup key to retrieve the entitlement data.
6. When the user makes resource requests during the session, the SAML Affiliate Agent sends the lookup key to the requesting application in an SMPERSISTENCEKEY HTTP header.
7. The application uses the key to retrieve the entitlement data that the library has extracted from the assertion and stored.

Lifetime of Entitlement Data

Since the entitlement data is under the control of the application and is stored outside of a browser session cookie, the data can exist beyond the duration of the session. As a result, after the session ends, the information is still available to the application for operations such as data analysis and reporting.

Note: The SAML Affiliate Agent will make an AssertionExpired call to this library to notify it that the user session has ended. The library will determine how it responds.

Entitlement Payload

The entitlement portion of the raw XML assertion is bound by the <SMprofile> tag—for example:

```
<SM:SMprofile xmlns:AM="http://www.netegrity.com/affiliateMinder">
    <SM:NVpair>cookie:AuthorizedAmount=500</SM:NVpair>
    <SM:NVpair>cookie:name=tking</SM:NVpair>
    <SM:NVpair>cookie:email=tking@netegrity.com</SM:NVpair>
    <SM:NVpair>cookie:usertype=premium</SM:NVpair>
    <SM:NVpair>header:usertype=premium</SM:NVpair>
</SM:SMprofile>
```

Persistence Library Path

Specify the full path and name of the persistence library in the configuration file AffiliateConfig.xml, as follows:

<PersistenceLibrary>*library_path_and_name*</PersistenceLibrary>

Note: Persistence library file names must be less than 256 characters.

More Information

Configuring the Affiliate Web Server Plug-in (see page 60)

Persistence API Functions

The functions in the Persistence API are:

- AssertionExpired()
- InitPersistenceLib()
- PersistAssertion()

The functions appear below in alphabetical order.

More Information

AssertionExpired() (see page 141)
InitPersistenceLib() (see page 142)
PersistAssertion() (see page 142)

AssertionExpired()

Notifies the persistence library that the user session has ended. It is up to the library to determine how to respond to this notification—for example, whether to delete the stored assertion associated with the key.

AssertionExpired() Syntax

int AssertionExpired(const char* key);

Parameter	I/O	Description
key	I	The key that applications use to retrieve the parsed entitlement data. The persistence library previously returned this key to the Affiliate Agent as an output parameter to PersistAssertion().

AssertionExpired() Returns

0 on success, or non-zero on failure.

InitPersistenceLib()

Notifies the persistence library that the Affiliate Agent has started up and that the library should execute any necessary initialization routines.

InitPersistenceLib() Syntax

int InitPersistenceLib();

InitPersistenceLib() Returns

0 on success, or non-zero on failure.

PersistAssertion()

Notifies the persistence library that the SAML Affiliate Agent has verified the assertion and is about to write the SMPROFILE cookie at the consumer site. The Agent passes in the raw XML assertion data and expects the library to provide the key that applications can use to retrieve the parsed entitlement data.

The SAML Affiliate Agent will free the key when it is done with it.

PersistAssertion() Syntax

```
int PersistAssertion
(
const char* assertionPayload
char* key
);
```

Parameter	I/O	Description
assertionPayload	I	The raw (unparsed) XML assertion. If shared sessions are used, the SAML Affiliate Agent removes session-related information from the raw XML data before calling PersistAssertion().
key	O	A key that applications use to retrieve the name/value pairs that the persistence library extracted from *assertionPayload*. The SAML Affiliate Agent stores this key value in the SMPERSISTENCEKEY field of the cookie SMPROFILE.

PersistAssertion() Returns

0 on success, or non-zero on failure.

Key Provider API

The SAML Affiliate Agent encrypts and decrypts cookies before passing the cookies to a user's browser. When an Agent receives a SiteMinder cookie, the agent key enables the Agent to decrypt the contents of the cookie. Because cookies may contain sensitive data, these keys secure this information.

For cookie encryption, the SAML Affiliate Agent can use either:

- A static shared secret. The shared secret is defined during installation of the SAML Affiliate Agent.
- Keys that are periodically changed. These dynamic key values are supplied to the SAML Affiliate Agent through the Key Provider API.

  The SAML Affiliate Agent polls the Policy Server for key updates at a regular interval. If keys have been updated, Agents pick up the changes during polling. When a Web Agent detects a key rollover has taken place, the Agent retrieves new values for the keys.

You specify the type of encryption key to use through the KeyModel element of the AffiliateConfig.xml file.

To supply dynamic encryption keys through a key provider library, define the KeyProviderLibrary and KeyRolloverInterval attributes in KeyModel:

- KeyProviderLibrary. The full path and file name of the key provider library.

  Note: Key provider library file names must be less than 256 characters.
- KeyRolloverInterval. The rollover time, in seconds, when the current key expires and a new one must be supplied to the SAML Affiliate Agent.

When a key provider library is used to supply dynamic keys, the KeyModel elements looks like this:

```
<KeyModel>KeyProviderLibrary</KeyModel>
 <KeyAttributes>
  <KeyProviderLibrary>library_path_and_name</KeyProviderLibrary>
  <KeyRolloverInterval>time_in_seconds</KeyRolloverInterval>
 </KeyAttributes>
</KeyModel>
```

More Information

Configuring the Affiliate Web Server Plug-in (see page 60)

Key Provider API Functions

The functions in the Key Provider API are:

- GetNextKey()
- InitKeyProvider()

The functions appear below in alphabetical order.

More Information

GetNextKey() (see page 145)
InitKeyProvider() (see page 145)

GetNextKey()

Passes the next encryption key to the SAML Affiliate Agent when the current key expires. The SAML Affiliate Agent will free the outBuffer when it is done with it.

The expiration time is specified in the KeyRolloverInterval element of the AffiliateConfig.xml file.

GetNextKey() Syntax

```
int GetNextKey(char* outBuffer);
```

Parameter	I/O	Description
outBuffer	O	The new encryption key to send to the SAML Affiliate Agent.

GetNextKey() Returns

0 on success, or non-zero on failure.

InitKeyProvider()

Notifies the key provider library that the SAML Affiliate Agent has started up and that the library should execute any necessary initialization routines.

InitKeyProvider() Syntax

```
int InitKeyProvider();
```

InitKeyProvider() Returns

0 on success, or non-zero on failure.

Chapter 11: SAML Affiliate Agent Troubleshooting

This section contains the following topics:

Configuration Changes Are Not Taking Effect

Symptom:

You have made configuration changes to the SAML Affiliate Agent, the Web Agent at the producer or the Affiliate Service and the changes are not taking effect.

Solution:

If you modify a configuration parameter, restart the associated service.

SAML Affiliate Agent Not Starting

Symptom:

The SAML Affiliate Agent is not starting.

Solution:

If there is a typographical error in the AffiliateConfig.xml file, this may cause parsing errors and prevent the SAML Affiliate Agent from starting.

If you are seeing this type of problem, do one of the following:

- Refer to the panic.log to review the error messages.

  The panic.log is used only on Solaris and Linux platforms and its location is dependent on the Web server, as follows:

  - Sun ONE Sun Java Systems—*web_server_home*/config
  - Apache—*web_server_home*/conf

  where *web_server_home* is the installed location of the Web server.

- For Windows systems, refer to the system's Event Log.

The httpd.conf File Shows a Log Level of "warn"

Symptom:

The httpd.conf file shows a log level of "warn" which is a greater severity than "info" yet the Apache 1.3.33 error log shows entries related to the Affiliate Agent with the level [Information].

Solution:

This is the expected behavior. It has been defined in the code that the Affiliate Agent logs error messages to the Apache error log with the hard-coded log level [Information]. So, irrespective of what is mentioned as the log level in the httpd.conf file, the Affiliate messages always have the log level of [Information].

IIS 6.0 Wildcard Mappings Required to Protect Resources

Symptom:

The SAML Affiliate Agent appears not to protect resources on an IIS 6.0 web server installed on a Windows system.

Solution:

If the SAML Affiliate Agent is properly installed, the appropriate DLL files are automatically added to the wildcard application mappings for the IIS 6.0 web server.

The two mapped files are:

- C:\Program Files\netegrity\affiliateagent\bin\plugins\ISAPI60AffiliateAgent.dll
- C:\Program Files\netegrity\webagent\bin\plugins\ISAPI6WebAgent.dll

You can see these mappings by selecting the properties for the Default Web Site and selecting the Home Directory tab. On the Home Directory tab, select Configuration. The Wildcard application maps section displays the files.

If the SAML Affiliate Agent is not properly installed and these mappings are not present, add the mappings manually.

Appendix A: Properties File for Unattended Installations

The nete-af-installer.properties file is generated during a SAML Affiliate Agent installation and contains all of the parameters, paths, and passwords entered during the installation.

During an unattended installation, the nete-af-installer.properties file provides the installation settings that would be entered by an end-user in a GUI or Console installation.

By default, the nete-af-installer.properties file contains the settings from the initial installation. You can use the default nete-af-installer.properties to create installations with the same settings as the initial installation, or use the nete-af-installer.properties file as a template that you modify to suit your environment.

To view the nete-af-installer.properties file, go to *saml_affiliate_agent_home*/install_config_info

This section contains the following topics:

General Information

In the General Information section, you can set the properties in the following table.

General Information Parameter	Description and Sample Value
DEFAULT_INSTALL_DIR	The location where the unattended installation places the SAML Affiliate Agent files. For example, Windows: C:\\Program Files\\netegrity\\affiliateagent UNIX: /opt/netegrity/affiliateagent
DEFAULT_JRE_ROOT	The full path to Java Runtime Environment (JRE). For example, Windows: C:\\Program Files\\java\\j2re1.4.2 UNIX: /opt/netegrity/java/j2re1.4.2

General Information Parameter	Description and Sample Value
DEFAULT_UPDATE_USER_PROFILE (UNIX only)	Configures required environment variables by adding the nete-af-env.sh script to the .profile file. To add the nete-af-env.sh script to the .profile file, set DEFAULT_UPDATE_USER_PROFILE=true If you do not want the installation to add the nete-af-env.sh script to the .profile file, set DEFAULT_UPDATE_USER_PROFILE=false

Web Server Descriptions

In the Web Server section, you can configure a web server for the SAML Affiliate Agent. The following table lists the parameters.

Web Server Parameter	Description and Sample Value
DEFAULT_WEBSERVER_INFO	Identifies the default web server. **For IIS web servers specify:** *type*, *root_folder* *type*=IIS *root_folder* =*saml_affiliate_agent_home*\bin\IIS folder. Example: Microsoft IIS,C:\\Program Files\\netegrity\\siteminder\\bin\\IIS; **For iPlanet web servers, specify:** *instance_name*, *instance_config_folder* instance_name=iPlanet web server instance name *instance_config_folder* =the location of the instance's configuration file directory. Example: https-instance1,/usr/iplanet/servers/ https-instance1/config;https-instance2,/usr/iplanet/servers/https- instance1/config **Note:** Use a semicolon (;) to separate multiple instances, as shown in the example. **For Apache Web servers:** This parameter does not apply. Leave the value blank. However, specify values for the DEFAULT_WEBSERVER_TYPE and the DEFAULT_WEBSERVER_ROOT parameters.

Web Server Parameter	Description and Sample Value
DEFAULT_WEBSERVER_TYPE (For Solaris installations only)	The type of web server on which the SAML Affiliate Agent runs. Specify IPLANET or APACHE. **Note:** Use capital letters.
DEFAULT_WEBSERVER_ROOT (For Solaris and Linux only)	The location of the web server root. For example: ■ iPlanet web servers: /export/iplanet/servers ■ Apache web servers: /usr/local/apache2

URL Information

You can specify URL and domain settings using the parameters in the following table.

URL Parameters	Description and Sample Value
DEFAULT_COOKIE_DOMAIN	The domain for the local server where the SAML Affiliate Agent is installed, such as .netegrity.com
DEFAULT_SSLIR_URL	The URL at the consumer site where the producer redirects users during requests. This is a URL to the consumer's secure Web server where the SAML Affiliate Agent is installed. We recommend that you use an SSL connection and that the URL begin with https://, such as https://mysslserver.netegrity.com:90 The SSLInterceptorURL enables the SAML Affiliate Agent to obtain the SAML artifact, which identifies the SAML assertion stored at the producer. The assertion contains user profile and session information. After the SAML Affiliate Agent gets the artifact, it makes a call on the SSL back channel to the producer to retrieve the actual assertion.
DEFAULT_PQR_URL	The URL at the producer where the Web Agent is installed. The SAML Affiliate Agent sends the user to this URL if they do not have the consumer profile and session cookies that are required to access a resource at the consumer. For example: https://myserver.netegrity.com:80
DEFAULT_FWSR_URL	The URL to the producer Web server where the Web Agent Option Pack is installed. This must be a secure URL that begins in the form https://, such as https://myserver.netegrity.com:81

Passwords and Shared Secrets

When you initially install the SAML Affiliate Agent, you specify a shared secret and a consumer (affiliate) password. The installer encrypts these passwords and enters them in the nete-af-installer.properties file.

You can use the encrypted shared secret and consumer password in subsequent installations.

If you want to specify a new shared secret or consumer password, you can enter new values in the nete-af-installer.properties file before you run an unattended installation; however, you must specify the new shared secret or consumer password in clear text.

The following table lists the password parameters you can modify.

Password Parameters	**Description and Sample Value**
DEFAULT_SHARED_SECRET	Allows you to specify a new shared secret, which the SAML Affiliate Agent uses to encrypt consumer cookies. Enter the shared secret in clear text. **Note:** By default, the DEFAULT_SHARED_SECRET is commented out and the unattended installer uses the encrypted secret in the ENCRYPTED_SHARED_SECRET parameter to encrypt consumer cookies. To specify a new shared secret: 1. Uncomment the DEFAULT_SHARED_SECRET parameter, and specify the new secret. For example: DEFAULT_SHARED_SECRET =newsecret 2. Comment out the ENCRYPTED_SHARED_SECRET parameter. For example: # ENCRYPTED_SHARED_SECRET = ENC:DVFrPCj3RcsRg1nxNpEODA== The shared secret is case-sensitive.
ENCRYPTED_SHARED_SECRET	The encrypted secret that the SAML Affiliate Agent uses to encrypt consumer cookies. You entered this shared secret during the initial SAML Affiliate Agent installation and cannot change it. Important: Changing the encrypted secret will cause communication between the SAML Affiliate Agent and the Policy Server to fail. To specify a new shared secret, uncomment the DEFAULT_SHARED_SECRET parameter and specify a new secret in clear text.

Password Parameters	Description and Sample Value
DEFAULT_AF_PW	Allows you to change the password that the SAML Affiliate Agent uses to communicate with the Policy Server at the producer site. The consumer password that you specify must be in clear text. **Note:** By default, the DEFAULT_AF_PW is commented out and the unattended installer uses the encrypted password in the ENCRYPTED_AF_PW parameter to communicate with the Policy Server. To specify a new password: 1. Uncomment the DEFAULT_AF_PW parameter, and specify the new password. For example: DEFAULT_AF_PW=newpassword 2. Comment out the ENCRYPTED_AF_PWparameter. For example: # ENCRYPTED_AF_PW = ENC:DVFrPCj3RcsRg1nxNpEODA== The consumer password is case-sensitive.
ENCRYPTED_AF_PW	Shows the encrypted password that the SAML Affiliate Agent uses to communicate with the Policy Server at the producer site. You entered this password during the initial SAML Affiliate Agent installation. Important: Changing the encrypted consumer password will cause communication between the SAML Affiliate Agent and the Policy Server to fail. To change the shared secret, uncomment the DEFAULT_AF_PW parameter and specify a new consumer password in clear text.

Appendix B: Parameter Descriptions for Agent Configuration

The following sections discuss parameter descriptions.

This section contains the following topics:

Alphabetical List of Elements and Attributes

The following table lists the elements and attributes that you can use to configure a SAML Affiliate Agent.

Element/Attribute	Default Value	Link to Description
AffiliateName	SampleAffiliate	Naming the Affiliate (see page 96)
AffiliatePassword	No default	Setting the Affiliate Password (see page 96)
AffiliateResource	transpolar examples	Designating Affiliate Resources (see page 97)
AllowCacheHeaders	no	Allowing HTTP Header Caching (see page 71)
AllowPOSTs	no	Allowing Post Actions (see page 105)
AllowUnknownUsers	no	Permitting Unknown Users Access to the Consumer (see page 102)
Append	no	Appending Messages to Log Files (see page 74)
AppendTarget	no	Appending the Original Affiliate Destination to the NoAccessURL (see page 103)
AssertionAudience	http://www.netegrity.com/SampleAudience	Designating the Assertion Issuer and Audience (see page 104)
AssertionIssuer	http://www.netegrity.com/SiteMinder	Designating the Assertion Issuer and Audience (see page 104)

Element/Attribute	Default Value	Link to Description
AuthenticationScheme	Basic	Determining How the Consumer Authenticates to the Producer
BadQueryChars	No default	Designating Bad Query Characters (see page 83)
BadURLChars	//,./,/../*,*.,~,\,%00-%1f,%7f-%ff,%25	Designating Bad URL Characters (see page 82)
BadCSSChars	No default	Protecting Web Sites Against Cross-Site Scripting (see page 83)
CookieDomain	No default	Specifying Affiliate Resources that Use Similar Responses and Headers (see page 98)
CompanySourceID	Hex value—Netegrity source ID	Setting the Company Source ID (see page 100)
CSSChecking	yes	Protecting Web Sites Against Cross-Site Scripting (see page 83)
CSSErrorFile	No default	Protecting Web Sites Against Cross-Site Scripting (see page 83)
CustomMessageFile	*saml_affiliate agent_home*\messages\iso-8859-1.msg	Locating a Custom Message File (see page 85)
Enabled	no	Enabling and Disabling the SAML Affiliate Agent (see page 95)
EnableOtherAuthTrans	no	Enabling the Agent to Work with Multiple AuthTrans Functions (see page 68)
GetAssertionService	No default	Configuring the Affiliate to Retrieve a SAML Assertion (see page 90)
GetPortFromHeaders	no	Using the HTTP HOST Request for the Port Number (see page 68)
HTTPHeaderEncodingSpec	UTF-8, no wrapping	Encoding and Wrapping HTTP Headers (see page 72)
HTTPSPorts	443	Defining HTTPS Ports (see page 69)
HTTPWrapSpec	RFC-2047	Encoding and Wrapping HTTP Headers (see page 72)
IgnoreExtensions	.gif, .jpeg, .png, .ccc	Specifying Ignored File Extensions (see page 80)

Element/Attribute	Default Value	Link to Description
KeyRolloverInterval	900 seconds	Configuring the Key Provider Library (see page 79)
KeystoreLocation	No default	Setting the Location of the Key Store (see page 76)
KeyModel	SharedSecret	Defining the Key Model for Encrypting Cookies
KeystorePassword	netegrity	Modifying the Key Store Password (see page 77)
LegacyVariables	no	Setting HTTP Header Syntax for Legacy Variables (see page 70)
LogLevel	0	Specifying Log Levels (see page 73)
LogFile	*saml_affiliate_agent_home*/log/affiliate.log	Recording Messages in a Log File (see page 73)
LogOffUri	logout.htm	Logging Users Out from a Session
LogToFile	yes	Recording Messages in a Log File (see page 73)
LogToConsole	yes	Displaying Log Messages in a Console (Windows only) (see page 75)
MaxSSLConnections	5	Specifying the Number of SSL Connections for Communication
MatchingRule	StrictPrefix	Defining the Matching Rule for the Affiliate Resource (see page 98) The MatchingRule attribute, required as part of the NotificationURL, indicates how the SAML Affiliate Agent compares each URL, for which the user gained access, to the NotificationURL.
NoAccessURL	No default	Denying a User Access to an Consumer Resource (see page 102)
NotificationService	No default	Specifying the Notification Service (see page 92)
NotificationURL	No default	Specifying the Notification Service (see page 92)
PersistenceLibrary	No default	Preserving Assertion Data (see page 84)
PortalQueryURL	No default	Redirecting Users without Valid Affiliate Cookies to the Producer (see page 90)

Element/Attribute	Default Value	Link to Description
PortalName	No default	Specifying the Portal Name (see page 90)
RequireActivePortalSession	no	Configuring an Active Portal Session (see page 110)
SessionModel	Default	Configuring SiteMinder Sessions for Federated Single Sign-on (see page 106)
SessionIdleTimeout	300 seconds	Default Session Overview (see page 108)
SessionMaxTimeout	600 seconds	Default Session Overview (see page 108)
SocketTimeout	300,000 milliseconds	Closing the Connection After an Assertion is Retrieved for Artifact SSO (see page 59)
SSLInterceptorURL	No default	Returning Users to the Affiliate after Obtaining a SAML Artifact (see page 76)
SharedSecret	No default	Specifying the Key Model (see page 77)
SessionProviderService	No default	Specifying the Session Provider Service (see page 91)

Set the Shared Secret and Affiliate Password

The SharedSecret and AffiliatePassword attributes in the AffiliateConfig.xml file cannot be modified directly in the file. You have to use the encryptkey utility to modify these values.

Important! The SAML Affiliate Agent is *not* compatible with the FIPS 140-2 encryption standards.

Change the SharedSecret and AffiliatePassword as follows:

- SharedSecret

  This value is set during the installation of the SAML Affiliate Agent. Use encryptkey to modify it. The shared secret can be between 1 and 255 characters with no embedded spaces. Note also that it is case-sensitive.

  To modify this value using encryptkey, use the following command syntax:

  encryptkey -path *path_to_AffiliateConfig.xml* -sharedSecret *new_secret*

- AffiliatePassword

  This value must be set after installation of the SAML Affiliate Agent. The administrator at the producer should provide the unencrypted password, and it must match at the consumer and producer sites.

  To modify this value using encryptkey, use the following command syntax:

  encryptkey -path *path_to_AffiliateConfig.xml* -affiliate *AffiliateName* -affiliatepassword *new_password*

The variables in the encrypt key commands are as follows:

path_to_AffiliateConfig.xml

Specifies the path, including the file name, of the configuration file. If any value in the path has spaces, the entire path must be enclosed by quotation marks. For example:

encryptkey -path "C:\Program Files\netegrity\affiliateagent\config\AffiliateConfig.xml" -sharedSecret nete

new_secret

Specifies the shared secret in clear text.

AffiliateName

Indicates the value entered for the AffiliateName attribute. This argument ensures that the password is changed for the correct consumer, especially if the AffiliateConfig.xml file has more than one PerAffiliateInfo section.

new_password

Specifies the password in clear text.

After running the encryptkey utility, the new secret or password is inserted into the AffiliateConfig.xml file.

Note: To see the all the arguments for use with encryptkey, open a command window and enter encryptkey followed by a carriage return.

Appendix C: Add Certificate Authorities to the Affiliate Key Store

The following sections discuss how to add certificate authorities to the AM.keystore.

This section contains the following topics:

Use the AM.keystore Database

For SAML 1.x artifact single sign-on, the SAML Affiliate Agent, sends a request for the assertion to the Assertion Retrieval Service. This service retrieves the assertion from the producer and then returns the assertion to the consumer over a back channel.

We recommend that the Assertion Retrieval Service be protected from unauthorized access. You secure this service by protecting the realm where this service resides.

The two authentication schemes you can use for protection are:

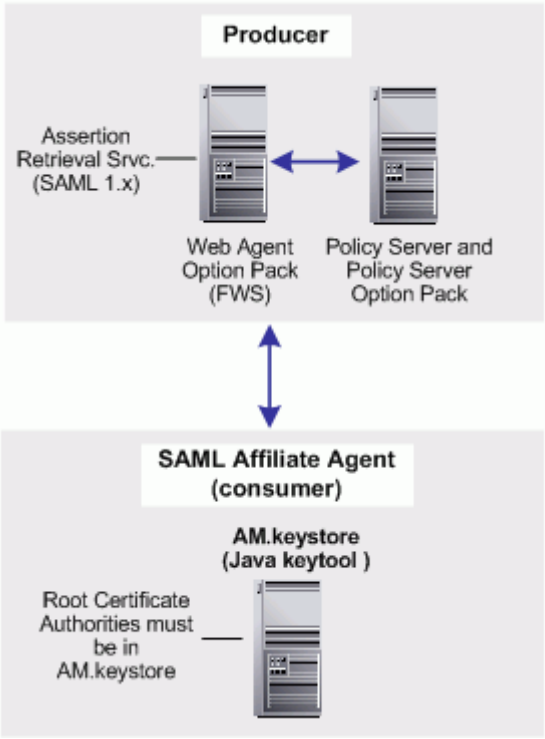
- Basic
- Basic over SSL

If you are using Basic over SSL, you must:

- Enable SSL at the producer-side web server
- Configure a database called the AM.keystore.

  This database must be set up correctly to allow the SAML Affiliate Agent to communicate with the Assertion Retrieval Service at the producer in a secure manner.

The following illustration shows where the AM.keystore resides in a SiteMinder federated network.



Information Stored in the AM.keystore Database

The Am.keystore holds Certificate Authority (CA) certificates. CA certificates are used for establishing an SSL connection from a consumer to the web server at a producer.

A set of common root CA certificates are shipped with the default AM.keystore database. To use a certificate for a CA that is not already in the key store, you must import the certificate into the AM.keystore database.

Formats Supported by AM.keystore

The AM.keystore database supports V1, V2, V3 of X.509 certificate format for root CAs. It also supports DER and base 64 encoding.

Modify the AM.keystore Database

If you are using a root or chain Certificate Authority (CA) to secure the back channel and it is not listed in the AM.keystore file, you need to add it to the file using the Java keytool utility. The keytool utility is part of the JRE included with the JDK required by the Web Agent Option Pack.

For example, a signed Verisign CA server-side certificate is used to SSL-enable the producer-side web server installed with the Web Agent Option Pack. To use this certificate for Basic over SSL authentication, add the Verisign certificate to the AM.keystore. This ensures that the consumer is communicating with a producer that can present a server-side certificate that has been verified by a trusted CA.

Note: You do not need to add the root certificate to the Web Agent key store.

For complete instructions for using the Java keytool utility, go to Sun Microsystems.

Import Root Certificate Authorities for Basic over SSL Authentication

Do one of the following:

- If you select the Basic over SSL authentication scheme and you are using one of the root certificates already in the AM.keystore file, then no additional configuration is required.
- If you are using a root certificate that is not in the AM.keystore, then the certificate has to be imported into the key store.

To set up the AM.keystore for Basic over SSL authentication:

1. Obtain a root certificate.
2. Check whether the CA is in the database by running the following Java keytool command:

   keytool -list -v -keystore "*path_to_AM.keystore*"

   If it is in the key store, configuration is complete. If it is not in the key store, import it as instructed in the next step.

The output of the keytool -list command is similar to the following:

Your keystore contains 12 entries:

Alias name: verisignclass3ca

Creation date: Aug 1, 2005

Entry type: trustedCertEntry

Owner: OU=Class 3 Public Primary Certification Authority, O="VeriSign, Inc.", C=US

Issuer: OU=Class 3 Public Primary Certification Authority, O="VeriSign, Inc.", C=US

Serial number: e49efdf33ae80ecfa5113e19a4240232

Valid from: Mon Aug 1 19:00:00 EST 2005 until: Wed Aug 3 18:59:59 EST 2005

Certificate fingerprints:

MD5:78:2A:02:DF:DB:2E:14:D5:A7:5F:0A:DF:B6:8E:9C:5D

SHA1:4F:65:56:63:36:DB:65:98:58:1D:58:4A:59:6C:87:93:4D:5F:2A:B4

3. To import a new CA certificate, start at a command prompt and enter the following:

   keytool -import -alias *key_alias* -file *cert_file* -trustcertst -keystore *key_store*

4. At the prompt, enter the key store password.

5. When asked if you trust the certificate, enter YES.

   The certificate is added to the AM.keystore.

Appendix D: Modifications to Sun Java System Files During Agent Installation

When you install the SAML Affiliate Agent on a Sun Java System Web server, configuration settings are added to the magnus.conf file and the obj.conf file. These files are loaded automatically when the web server is started, and the added settings are used to initialize the SAML Affiliate Agent.

Note: This information is primarily for reference; you do not have to make the modifications documented in this chapter after you install the Agent. However, if you remove the Affiliate Agent from your web server, you should remove the Agent-related entries.

When the SAML Affiliate Agent installation program adds information to the Web server's configuration, it divides this information differently, depending on the version of the Sun Java System Web Server.

Note that the first two Init fn lines are added to the magnus.conf file. All the remaining lines are added to the obj.conf file.

This section contains the following topics:

Modifications Made to Sun Java System/Windows Platforms

The obj.conf and magnus.conf files are located as follows:

Sun_Java_System_home\https-hostname\config\

Sun_Java_System_home

Specifies the location where the Sun Java System server is installed on your system.

hostname

Indicates the name of the server instance.

Code Added to the magnus.conf File (Sun Java System/Windows Platform)

The following lines of code are added by the SAML Affiliate Agent installation program. Some of the entries may differ slightly from your files.

```
Init fn="load-modules" shlib="C:/Program
Files/netegrity/affiliategent/bin/plugins/NSAPIAffiliateAgent.dll"
funcs="SmInitAffiliate,SiteMinderAffiliate,SmRequireAuth"
Init fn="SmInitAffiliate" config="C:/Program
Files/netegrity/affiliateagent/config/AffiliateConfig.xml"
logfile="C:/iPlanet/Servers/https-myserver/logs/affiliate.log"
```

The first two lines beginning with Init fn instruct the web server to load the SAML Affiliate Agent with three NSAPI functions: SmInitAffiliate, SiteMinderAffiliate, and SmRequireAuth.

Code Added to the obj.conf File (Sun Java System/Windows Platform)

The following lines of code are added by the SAML Affiliate Agent installation program:

```
AuthTrans fn="SiteMinderAffiliate"
PathCheck fn="SmRequireAuth"
```

The line that reads AuthTrans fn="SiteMinderAffiliate" is added to the default object (<Object name="default">). It sets up the SAML Affiliate Agent as the Authorization method, or AuthTrans function, for the web server.

The lines that reads PathCheck fn="SmRequireAuth" is added to any existing PathCheck lines in the default object. It verifies that the user is authorized to perform the requested action on the requested resource.

Modifications Made to Sun Java System/UNIX Platforms

The obj.conf and magnus.conf files are located as follows:

/export/smuser/*Sun_Java_System_home*/server6/https-*hostname*/config/

Sun_Java_System_home

Specifies the location where the Sun Java System server is installed on your system.

hostname

Indicates the name of the server.

Code Added to the magnus.conf File (Sun Java System/UNIX Platform)

The following lines of code are added by the SAML Affiliate Agent installation program:

```
Init fn="load-modules"
shlib="/export/smuser/netegrity/affiliateagent/bin/plugins/libNSAPIAffiliateAgent
.so" funcs="SmInitAffiliate,SiteMinderAffiliate,SmRequireAuth"


Init fn="SmInitAffiliate"
config="/export/smuser/servers/https-server.netegrity.com/config/AffiliateConfig.
xml"
logfile="/export/smuser/server6/https-myserver.mysite.com/logs/affiliate.log"
```

Code Added to the obj.conf File (Sun Java System/UNIX Platform)

The following lines of code are added by the SAML Affiliate Agent installation program:

AuthTrans fn="SiteMinderAffiliate"

The first two lines beginning with Init fn instruct the web server to load the SAML Affiliate Agent with three NSAPI functions:

- SmInitAffiliate
- SiteMinderAffiliate
- SmRequireAuth

The line that reads AuthTrans fn="SiteMinderAffiliate" is added to the default object (<Object name="default">). It sets up the SAML Affiliate Agent as the Authorization method, or AuthTrans function, for the web server.

The lines that reads PathCheck fn="SmRequireAuth" is added to any existing PathCheck lines in the default object. It verifies that the user is authorized to perform the requested action on the requested resource.

Appendix E: Federation Web Services Messages

This section contains the following topics:

Sample Affiliate Agent Request Messages

The SAML Affiliate Agent interacts with Federation Web Services using the HTTP POST method. Some sample request messages appear below so you can see the type of data that is exchanged. Each sub-section contains a message that is specific to a particular service.

SAML Request for Assertion Retrieval

An actual schema for the SAML Request element can be found at Oasis.

Look for the document cs-sst-schema-protocol-01.xsd.

A SAML artifact is an 42-byte number that uniquely identifies an assertion. The artifact contains these elements:

- Type code
- Source ID
- Assertion Handle

This is a sample request for assertion retrieval:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'><SOAP-ENV:Body>
<samlp:Request xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"  MajorVersion="1" MinorVersion="0"
RequestID="112.112.1.12.1028670652906" IssueInstant="2002-08-06T21:50:52.906Z" ><
samlp:AssertionArtifact>AAG4GEUmEKDqQxv/ad00au7/gxKLakNjY1IwSW1FT3VVMVhvSjhEd0hBd2xZSTRRTT06YTI0N
WMwZmQyYzJlNTU2Mw==< /samlp:AssertionArtifact>
</samlp:Request>
</SOAP-ENV:Body></SOAP-ENV:Envelope>
```

Notification Request Message

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<SOAP-ENV:Body>
<ns1:notifyAccess xmlns:ns1="urn:com-netegrity-affiliate-webservice" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<affiliateName xsi:type="xsd:string">…</affiliateName >
<sessionId xsi:type="xsd:string">…</sessionId >
<userDN xsi:type="xsd:string">…</userDN>
<accessedURL xsi:type="xsd:string">…</accessedURL >
<accessTimestamp xsi:type="xsd:string">…</accessTimestamp >
</ns1:notifyAccess>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Session Logout Request Message

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<SOAP-ENV:Body>
<ns1:logout xmlns:ns1="urn:com-netegrity-affiliate-webservice" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<sessionId xsi:type="xsd:string">…</sessionId>
</ns1:logout>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Session Validate Request Message

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<SOAP-ENV:Body>
<ns1:validateSession xmlns:ns1="urn:com-netegrity-affiliate-webservice" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<sessionId xsi:type="xsd:string">…</sessionId>
</ns1:validateSession>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Sample Response Messages

The federation web services receive the message from the client, process it, and return a SOAP-encoded XML "response" message.

Some sample response messages appear below. Each sub-section contains a message that is specific to a particular service.

SAML Response with an Embedded Assertion

An actual schema for the SAML response element can be found at OASIS.

Look for the document cs-sst-schema-protocol-01.xsd.

Below are two samples of SAML responses with embedded assertion.

Sample 1

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
<saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns:SM="http://www.netegrity.com/SiteMinder" MajorVersion="1" MinorVersion="0"
AssertionID="111.123.1.1.1029438859437"
Issuer="http://www.netegrity.com/SiteMinder"
IssueInstant="2002-08-15T19:14:19.437Z">
<saml:Conditions NotBefore="2002-08-15T19:13:49.125Z"
NotOnOrAfter="2002-08-15T19:15:49.125Z">
<saml:AudienceRestrictionCondition>
<saml:Audience>http://jsmith.netegrity.com/rd/redirect.jsp</saml:Audience>
</saml:AudienceRestrictionCondition>
</saml:Conditions>
<saml:AttributeStatement>
<saml:Subject>
<saml:NameIdentifier NameQualifier="www.netegrity.com"
Format="urn:oasis:names:tc:SAML:1.0:assertion">uid=tking,o=netegrity.com<
/saml:NameIdentifier>
</saml:Subject>
```

```
<saml:Attribute AttributeName="SMContent"
AttributeNamespace="http://www.netegrity.com/SiteMinder">
<saml:AttributeValue>
<SM:SMContent>
<SM:SMsession>
<SM:SessionID>pZKZpWxyAUip18QlVIGBMaNeyiQ=</SM:SessionID>
<SM:startTime>1029438853</SM:startTime>
<SM:idleTimeout>3600</SM:idleTimeout>
<SM:maxTimeout>7200</SM:maxTimeout>
<SM:timeIn>30</SM:timeIn>
</SM:SMsession>
<SM:SMlogin>
<SM:UserDN>uid=tking,o=netegrity.com</SM:UserDN>
<SM:Username>tking</SM:Username>
</SM:SMlogin>
<SM:SMprofile>
<SM:NVpair>header:name=smith</SM:NVpair>
</SM:SMprofile>
</SM:SMContent>
</saml:AttributeValue>
</saml:Attribute>
</saml:AttributeStatement>
<saml:AuthenticationStatement AuthenticationMethod="Unspecified"
AuthenticationInstant="2002-08-15T19:14:13.000Z">
<saml:Subject>
<saml:NameIdentifier NameQualifier="www.netegrity.com"
Format="urn:oasis:names:tc:SAML:1.0:assertion">uid=tking,o=netegrity.com</saml:Na
meIdentifier>
</saml:Subject>
</saml:AuthenticationStatement>
</saml:Assertion>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Sample 2

```
:<SOAP-ENV:Envelope xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'><
SOAP-ENV:Body>
<samlp:Response xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"  MajorVersion="1"
MinorVersion="0" ResponseID="112.123.1.123.1028670653171"
InResponseTo="112.123.1.123.1028670652906"
IssueInstant="2002-08-06T21:50:53.171Z" ><samlp:Status
xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion" ><samlp:StatusCode
Value="samlp:Success" ></samlp:StatusCode>
</samlp:Status>
<saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
MajorVersion="1" MinorVersion="0" AssertionID="112.123.1.123.1028670646671"
Issuer="http://www.netegrity.com/AffMinder"
IssueInstant="2002-08-06T21:50:46.671Z" ><saml:Conditions
NotBefore="2002-08-06T21:50:15.703Z" NotOnOrAfter="2002-08-06T22:01:15.703Z" ><
saml:AudienceRestrictionCondition><saml:Audience>http://jsmith.netegrity.com</sam
l:Audience>
</saml:AudienceRestrictionCondition>
</saml:Conditions>
<saml:AttributeStatement><saml:Subject><saml:NameIdentifier
NameQualifier="www.netegrity.com" Format="urn:oasis:names:tc:SAML:1.0:assertion"
>uid=user1,ou=people,o=security.com</saml:NameIdentifier>
</saml:Subject>
<saml:Attribute AttributeName="SMContent"
AttributeNamespace="http://www.netegrity.com/affiliateMinder" ><
saml:AttributeValue><AM:SMContent>
    <AM:SMsession>
        <AM:SessionID>CccR0ImEOuU1XoJ8DwHAwlYI4QM=</AM:SessionID>
        <AM:startTime>1028670640</AM:startTime>
        <AM:idleTimeout>3600</AM:idleTimeout>
        <AM:maxTimeout>7200</AM:maxTimeout>
        <AM:timeIn>30</AM:timeIn>
    </AM:SMsession>
    <AM:SMlogin>
        <AM:UserDN>uid=user1,ou=people,o=security.com</AM:UserDN>
        <AM:Username>user1</AM:Username>
    </AM:SMlogin>
    <AM:SMprofile>
        <AM:NVpair>header:AffID=affiliateA0001</AM:NVpair>
    </AM:SMprofile>
</AM:SMContent>
</saml:AttributeValue>
</saml:Attribute>
</saml:AttributeStatement>
```

```
<saml:AuthenticationStatement AuthenticationMethod="Unspecified"
AuthenticationInstant="2002-08-06T21:50:40.000Z" ><
saml:Subject><saml:NameIdentifier NameQualifier="www.netegrity.com"
Format="urn:oasis:names:tc:SAML:1.0:assertion"
>uid=user1,ou=people,o=security.com</saml:NameIdentifier>
</saml:Subject>
</saml:AuthenticationStatement>
</saml:Assertion>
</samlp:Response>
</SOAP-ENV:Body></SOAP-ENV:Envelope>
```

Notification Response Message

The following is a sample notification response message

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<SOAP-ENV:Body>
<ns1:notifyAccessResponse xmlns:ns1="urn:com-netegrity-affiliate-webservice"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<return xsi:type="xsd:int">1</return>
</ns1:notifyAccessResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Session Logout Response Message

The following is a sample session logout response message.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<SOAP-ENV:Body>
<ns1:logoutResponse xmlns:ns1="urn:com-netegrity-affiliate-webservice"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<return xsi:type="xsd:int">1</return>
</ns1:logoutResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Session Validate Response Message

The following is a sample session validate response message.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<SOAP-ENV:Body>
<ns1:validateSessionResponse xmlns:ns1="urn:com-netegrity-affiliate-webservice"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<return xsi:type="xsd:int">1</return>
</ns1: validateSessionResponse>
< /SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Appendix F: 4.x Affiliate.conf Parameter Mappings

This section contains the following topics:

Map 4.x Affiliate Parameters to 6.x Parameters

The following table lists the parameter mappings from the 4.x Affiliate.conf file to the settings in the AffiliateConfig.xml file.

Note: The entry n/a (not applicable) means that there is no equivalent parameter to map between the 4.x and 6.x QMR x settings.

4.x Affiliate.conf Settings	6.x QMR x AffiliateConfig.xml Settings
loglevel	LogLevel
logfilename	LogFile
logfile	LogToFile
logappend	Append
logconsole	LogToConsole
Enableaffiliate	Enabled
ignoreexts	IgnoreExtensions
companyname	n/
affiliateresource	AffiliateResource
portal	PortalQueryURL
allowsiteminderanonymous	n/a
allowunknownusers	AllowUnknownUsers
cookiedomain	CookieDomain
affiliatename	AffiliateName
requireactiveportalsession	RequireActivePortalSession
affiliatetimeout	SessionMaxTimeout

4.x Affiliate.conf Settings	6.x QMR x AffiliateConfig.xml Settings
noaccessurl	NoAccessURL
noaccesstarget	AppendTarget
sharedsecret	n/a SharedSecret setting exists, but it functions differently from the 4.x setting
n/a	HTTPWrapSpec
n/a	LegacyVariables
n/a	AllowCacheHeaders
n/a	GetPortFromHeaders
n/a	EnableOtherAuthTrans
n/a	HTTPHeaderEncodingSpec
n/a	SSLInterceptorURL
n/a	KeystoreLocation
n/a	KeystorePassword
n/a	KeyProviderLibrary
n/a	KeyRolloverInterval
n/a	BadQueryChars
n/a	BadURLChars
n/a	CSSChecking
n/a	BadCSSChars
n/a	CSSErrorFile
n/a	CSSChecking
n/a	CustomMessageFile
n/a	MaxSSLConnections
n/a	GetAssertionService
n/a	SessionProviderService
n/a	NotificationService
Allowposts	AllowPOSTs
n/a	CompanySourceID
n/a	MatchingRule

4.x Affiliate.conf Settings	6.x QMR x AffiliateConfig.xml Settings
n/a	AffiliatePassword
n/a	AuthenticationScheme
n/a	AssertionAudience
n/a	AssertionIssuer
n/a	SessionIdleTimeout
n/a	LogoffURI
n/a	NotificationURL
n/a	MatchingRule
n/a	SocketTimeout

Appendix G: Environment Variables Modified by the Agent Installation

This section contains the following topics:

Added or Modified Environment Variables

The following environment variables are added or modified by the SAML Affiliate Agent installation:

- NETE_AM_ROOT = $USER_INSTALL_DIR$
- NETE_AM_PATH = $USER_INSTALL_DIR$$/bin/$plugins

Index

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

W