CA™ SOA Security Manager

Agent Configuration Guide

r12.1

Second Edition

CA Product References

This document references the following CA products:

- CA SOA Security Manager
- CA SiteMinder® Web Access Manager

Contact CA

Contact Technical 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 also available on the CA Support website, found at http://ca.com/docs.

Contents

Chapter 1: SOA Agent for Web Servers

This section contains the following topics:

Overview

The SOA Agent for Web Servers is an XML-enabled version of the CA SiteMinder Web Agent that operates with a Web server to handle XML messages sent to Web service implementations.

When a Web consumer (client) application sends an XML message to a URL that is bound to a Web service, the SOA Agent for Web Servers intercepts these messages and communicates with the Policy Server to process authentication and authorization requests before the XML message is passed on to the Web service. In addition, the Policy Server can provide information that the SOA Agent for Web Servers adds to the XML message, such as a SAML assertion based on the originating client application's identity.

Note: If you have purchased CA SiteMinder, you can also use the core Web Agent functionality of the SOA Agent for Web Servers to protect other resources on a Web server. For more information about this functionality, see the *CA SiteMinder Agent Guide*—the remainder of this chapter deals specifically with use of the SOA Agent for Web Servers to protect Web services.

SOA Agent for Web Servers Functions

The SOA Agent for Web Servers performs the following tasks:

- Intercept posted XML messages to protected Web services and work with the Policy Server to determine whether or not a client application should have access.
- Ensure a client application's ability to access Web services quickly and securely. The SOA Agent for Web Servers stores contextual information about client application access privileges in a session cache. You can optimize performance by modifying the cache configuration settings.
- Support multistep and chain authentication service models by generating and consuming SAML Session Tickets and WS-Security tokens.

The SOA Agent for Web Servers and the Policy Server

To enforce Web service access control, the SOA Agent for Web Servers interacts with the Policy Server, where all authentication and authorization decisions are made.

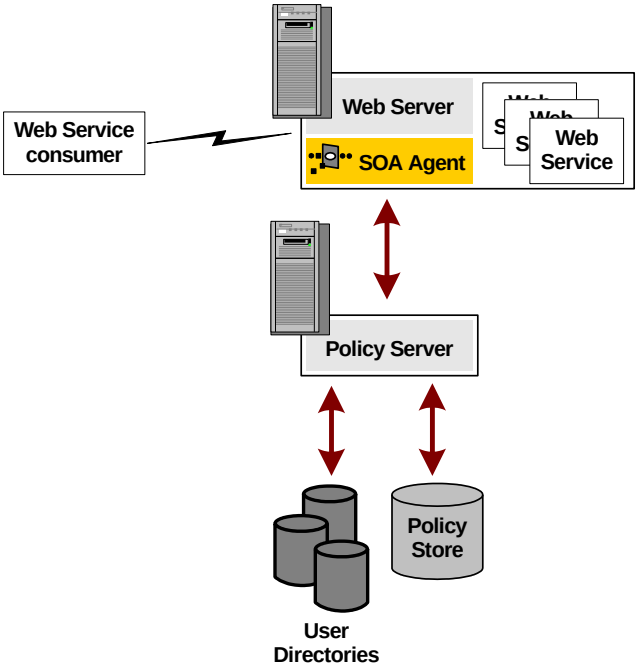
The SOA Agent for Web Servers intercepts XML messages posted to a Web server and checks with the Policy Server to see if the requested resource is protected. If the resource is unprotected, the access request proceeds directly to the Web server. If the resource is protected, the following occurs:

- The SOA Agent for Web Servers checks which authentication method is required for this resource. Typical credentials are a name and password, but other credentials, such as a certificate or SAML assertion, may be required.
- The SOA Agent for Web Servers obtains credentials from the transport, header, or body of the XML message.
- The SOA Agent for Web Servers passes the credentials to the Policy Server, which determines if the credentials are sufficient for the authentication method.
- If the posted XML message passes the authentication phase, the Policy Server determines if the message is authorized to access the resource. If a policy uses policy expressions as part of the authorization process, the SOA Agent for Web Servers may need to resolve the variables used in these expressions if the Policy Server cannot resolve them.
- Once the Policy Server grants access, the SOA Agent for Web Servers allows the access request to proceed to the Web service.

The SOA Agent for Web Servers can also receive message-specific attributes, in the form of *responses,* to be passed on to the Web service. A response is a personalized message or other message-specific information returned to the SOA Agent for Web Servers from the Policy Server after authorizing the message. A response consists of name-value attribute pairs that instruct the SOA Agent for Web Servers to generate SAML Session Tickets and WS-Security tokens.

SOA Agent for Web Servers Support for Web Servers

To protect Web services hosted on a Web server, you deploy a SOA Agent for Web Servers on that Web server (as shown in the following illustration). You then configure authentication and authorization policies for the Web service resources hosted on that Web server.



For a list of Web server platforms on which the SOA Agent for Web Servers is supported, see the SOA Security Manager Platform Support matrix on the Technical Support site at http://ca.com/support.

Configure the SOA Agent for Web Servers

Configure SOA Agent for Web Servers objects as you would a SiteMinder Web Agent. The SOA Agent for Web Servers fully conforms to the CA SiteMinder central Agent management model.

You need to configure:

- Host Configuration Object (one for each Web server)
- Agent Configuration Object (one for each SOA Agent for Web Servers)
- Agent identity (one for each SOA Agent for Web Servers)

Note: For detailed information about how to configure Agent-related objects, see *the SOA Security Manager Policy Configuration Guide* and the *SOA Security Manager Implementation Guide*.

To configure the SOA Agent for Web Servers

1. On the Policy Server:
   a. Duplicate or create a Host Configuration Object, which holds initialization parameters for a Trusted Host.

      The Trusted Host is a server that hosts one or more Agents and handles their connection to the Policy Server.

   b. As necessary, add or edit Trusted Host parameters in the Host Configuration Object that you just created.

   c. Duplicate or create an Agent Configuration Object, which holds Agent configuration parameters and can be used to centrally configure a group of Agents.

   d. Add or edit required Agent parameters in the Agent Configuration Object.

      The configuration object should include the standard Web Agent parameters—DefaultAgentName or AgentName to specify the Agent identity, and for IIS Agents, DefaultUserName and DefaultPassword for the IIS proxy account.

   e. Create an Agent identity for the SOA Agent for Web Servers. You must select Web Agent as the Agent type for an SOA Agent for Web Servers.
2. On the system where the SOA Agent for Web Servers is installed, configure the underlying Web Agent to work with the web server.

More information:

SOA Agent for Web Servers Parameters (see page 13)
Configure the SOA Agent to Work With IIS (see page 16)
Configure the SOA Agent to Work With Sun Java System Web Server (see page 24)
Configure the SOA Agent to Work With the Apache HTTP Server (see page 31)
Configure the SOA Agent to Work With the Domino Web Server (see page 41)
Configuration Procedures That Apply for All Web Agent Types (see page 47)

SOA Agent for Web Servers Parameters

The following table lists configuration parameters for the SOA Agent for Web Servers.

Parameter Name	Value	Description
IgnoreXMLSDK	yes or no	If this parameter is added to the Agent Configuration Object and is set to Yes, the SOA Agent for Web Servers is disabled. This means that the Agent behaves as a Web Agent for all incoming requests. If added to the Agent Configuration Object and set to No (or not added to to the Agent Configuration Object at all), the SOA Agent for Web Servers is enabled. That is, the Agent uses the XML SDK to process incoming HTTP requests under these conditions: ■ HTTP action is POST ■ HTTP MIME type is "text/xml" or, if the XMLSDKMimeTypes parameter is configured, any one of the MIME types specified by that parameter. ■ HTTP content is an XML document

Parameter Name	Value	Description
XMLSDKMimeTypes	String	A comma-delimited list of MIME types that the SOA Agent for Web Servers will accept for processing by SOA Security Manager. All POSTed requests having one of the listed MIME types are processed. Examples: ■ text/xml ■ application/octet-stream ■ text/xml,multipart/related If you do not add this parameter to the Agent Configuration Object, the SOA Agent for Web Servers defaults to accepting only the text/xml MIME type.
ServerProductName	String	Description of the product name—for example, iPlanet Web Server. Provides a value for the SOA Agent for Web Servers variable property Server Product Name. **Note:** For more information about setting this variable, see the *SOA Security Manager Policy Configuration Guide*.
ServerVendor	String	Description of the Web Server vendor—for example, Sun. Provides a value for the SOA Agent for Web Servers Variable property Server Vendor. **Note:** For more information about setting this variable, see the *SOA Security Manager Policy Configuration Guide*.
ServerVersion	String	Description of the product version—for example, 6.0 SP2). Provides a value for the SOA Agent for Web Servers Variable property Server Version. **Note:** For more information about setting this variable, see the *Policy Configuration Guide*.

Parameter Name	Value	Description
MaxXmlSdkRetries	Number	Defines the number of times the SOA Agent for Web Servers tries to contact the XML SDK server when it receives requests. The default is 3. The Agent does not continually retry the server for the same request. If a request comes in and the Agent cannot contact the SDK server, that request is dropped and the Agent tries again when a subsequent request is made. The Agent attempts to connect for each new request until it reaches the number specified in this parameter. If the SOA Agent does not connect to the XML SDK server, the Agent assumes the server is not running and stops trying to process SOA Security Manager-specific requests. **Note:** For the SOA Agent for Web Servers on Apache Web servers, this value applies to each process.
XMLSDKResourceIdentification	yes or no	Determines if the SOA Agent should identify the web service operation being requested by an incoming XML message as well as the resource identifier (that is, perform fine-grain resource identification). The default is No. **Note:** You must set this option to Yes if you want to use the Administrative UI to configure policies to protect resources with the SOA Agent for Web Servers.
XMLSDKAcceptSMSessionCookie	yes or no	Determines whether or not the SOA Agent for Web Servers accepts an CA SiteMinder session cookie to authenticate a client. The default is no. If set to yes, the SOA Agent for Web Servers uses information in a session cookie sent as an HTTP header in the request as a means of authenticating the client. If set to no, session cookies are ignored and the SOA Agent for Web Servers requests credentials required by the

Parameter Name	Value	Description
		configured authentication scheme.
SAMLSessionTicketLogoffi	yes or no	Determines whether the SOA Agent for Web Servers should attempt to log off session tickets in SAML assertions. The default is yes.
XMLAgentSoapFaultDetails	yes or no	Determines whether or not the SOA Agent for Web Servers should insert the authentication/authorization rejection reason (if provided by the Policy Server) into the SOAP fault response sent to the Web service consumer. The default is no.

Configure the Underlying Web Agent to Work With the Web Server

The SOA Agent for Web Servers is an XML-enabled version of the CA SiteMinder Web Agent. Unless you have installed the SOA Agent software on top of a previously installed and configured Web Agent, you must configure the underlying Web Agent to work with the host web server before you configure the SOA Agent itself.

Configure the SOA Agent to Work With IIS

Unless you installed the SOA Agent over a previously installed and configured IIS Web Agent, you must configure the underlying Web Agent to work with the IIS web server.

How to Configure a SiteMinder Web Access Manager Web Agent on IIS 6.0

Before you can use the Web Agent on an IIS 6.0 web server, you must complete the prerequisites using the following process:

1. Assign read permissions to samples and error files directories.
2. Allow IIS to execute Web Agent ISAPI and CGI extensions.
3. (Optional) Increase the Web Agent's size limit for uploaded files.
4. Gather the Web Agent information.
5. Run the Configuration Wizard for an IIS Web Agent.
6. Put the Agent filter and extension before other third-party filters.

Assign Read Permissions to Samples and Error Files Directories

The Network Service account must have Read permissions to any directory where the Web Agent reads forms credential collector (FCC) files and to any directory where the Web Agent reads Web Agent custom error files.

To Assign Read Permissions to the Samples and Error Files Directories

1. Open Windows Explorer and go to the appropriate directory:
   - samples: *web_agent_home*/samples
   - custom error file: the location or your custom error files. There is no default location.
2. Right-click the directory and select Sharing and Security.
3. Select the Security tab.
4. Click Add.

   The Select Users, Computers, or Groups dialog box opens.
5. Do one of the following:
   a. Accept the defaults for the Select this object type and From this Location fields.
   b. In the Enter the object names to select field, enter Network Service and click OK.

   You return to the Properties dialog box for the directory.
6. In the Permissions for Network Service scroll-box, allow Read permissions.
7. Click OK to finish.
8. Repeat this procedure for each directory.

Allow IIS to Execute the Agent ISAPI and CGI Extensions

You must add certain ISAPI and CGI extentions to the IIS 6.0 web server and grant the server permission to execute them before configuring the SiteMinder Web Access Manager Web Agent. These extensions will execute the Web Agent ISAPI and CGI scripts and other files.

To add the extensions and permissions

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. To add the ISAPI Web Agent extension, do the following:
   a. Click the Add a new Web service extension link.

      The New Web Service Extension dialog box opens.
   b. In the Extension name field, enter ISAPI6WebAgentDLL, and then click Add.

      The Add File dialog box opens.
   c. Click the Browse button, and then navigate to the ISAPI6WebAgent.dll file in the *web_agent_home*/bin directory. If the proper file does not appear, click the Files of type drop-down list and select either ISAPI dll files (for the .dll files) or CGI exe files (for .exe files).

      web_agent_home

      Indicates the directory where the Web Agent is installed.

      Default (Windows installations): [set Access Path variable]\webagent

      Default (UNIX installations): [set the Installation Path variable]/webagent
   d. Click Open

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

      You return to the New Web Service Extension dialog box.
   f. Select the Set extension status to allowed check box.
   g. Click OK.

      The New Web Service Extension dialog box closes.
4. Repeat Step 3 and add each of the following Web Agent files. Even though both files use the same name, you must add a separate extension for each because they are in different directories.

- *web_agent_home*/pw/smpwservicescgi.exe (suggested extension name: Password Services CGI)
- *web_agent_home*/pw_default/smpwservicescgi.exe (suggested extension name: PW Default CGI)

IIS 6.0 Web Agents and Third-Party Software on the Same Server

The IIS 6.0 Web Agent consists of an ISAPI filter and an ISAPI extension. The majority of Web Agent processing occurs in the extension.

When the Web Agent is installed on an IIS 6.0 Web Server with other third-party software, such as WebSphere or ServletExec, the Agent has the following restrictions:

- The Web Agent filter and Web Agent extension must be configured to run before other third-party filters installed on the web server.
- The Web Agent must be configured as the first wildcard application map if it is going to protect applications running as or spawned by an ISAPI extension.
- The IIS 6.0 web server does not enforce how third-party filters and extensions behave. IIS 6.0 processes ISAPI filters before calling ISAPI extensions, including the Web Agent extension. Therefore, the SiteMinder Web Access Manager Web Agent for IIS 6.0 is unable to authenticate or authorize access to applications implemented as pure ISAPI filters. This limitation impacts Web Agent integration with other third-party offerings for the IIS 6.0 web server, if those offerings are implemented as ISAPI filters that process and/or redirect the request before ISAPI extensions are called.

Increase the Agent's Size Limit for Uploaded Files

The Web Agent installed on an IIS 6.0 web server has a size limit of 2.5 MB for uploading files. If you want to increase this size limit, you can add a new key to the Windows registry on your web server.

To upload files that are larger than this limit

1. Open the registry editor.

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

2. Navigate to the following location:

   HKEY_LOCAL_MACHINE\SOFTWARE\[set the alternate Access Path variable]\SiteMinder Web Agent\Microsoft IIS

3. Create a new DWORD registry key in the previous location using the following name:

   MaxRequestAllowed

4. Set this value of the key to the number of bytes that corresponds to the size limit you want.

   The value of this key overrides the default limit. If the value of this key is less than or equal to 0, than the default of 2.5 MB (2,500,000 B) is used. This key accepts decimal values from 0 to 4294967295.

   Note: The IIS 6.0 web server has its own size limit. Changing the Web Agent's limit will not affect the IIS 6.0 limit. If you want to change the IIS 6.0 server's limit, see the Microsoft IIS 6.0 documentation or online help.

5. Close the registry editor.

   The size limit is changed.

Run the Configuration Wizard for an IIS Web Agent

Before you configure the Agent, you may want to register the system where the Agent is installed as a trusted host; however, you can do this at a later time.

To configure an IIS Web Agent

1. If necessary, start the Web Agent Configuration Wizard.

   The default method is to select Start, Programs, SiteMinder Web Access Manager, Web Agent Configuration Wizard. If you have placed the Wizard shortcut in a non-default location, the procedure will be different.

   Important! If you are running this wizard on Windows Server 2008, run the executable file with Administrator permissions, even if you are logged into the system as an Administrator. For more information, see the release notes for your SiteMinder Web Access Manager component.

   Note: If you chose to configure the Web Agent immediately after the installation, SiteMinder Web Access Manager automatically starts the wizard automatically.

2. If you have registered the trusted host, skip to the next step. If not, select No in the Host Registration dialog box to skip registration, then click Next.

3. Select the web server instances that you want to configure with Web Agents.

   If you have already configured a server with a Web Agent and you are running the Configuration Wizard to configure additional web servers instances, the Wizard displays the Select One or More Instances to Overwrite dialog box. This dialog box lists the web servers that you have previously configured.

   a. Select one of the following:

      Overwrite—to overwrite the server instance configuration.

      Preserve—to preserve the web servers configuration.

      Important! If you uncheck a previously configured server, the Web Agent will be removed from this server.

   b. Click Next.

4. In the Agent Configuration Object field, enter the name of the Agent Configuration Object for this web server instance, then click Next.

   This name must match an Agent Configuration Object already defined at the Policy Server. For example, enter IISDefaultSettings to use the default.

5. If you want to configure Registration Services for DMS2, select Yes. If not, select No.

   A servlet engine is required to run Self Registration. If the Web Agent Configuration Wizard does not detect a servlet engine, the Select Servlet Engine for Registration dialog box is not displayed.

   If you selected Yes to configure Registration Services:

a. Select a servlet engine to be configured for the web server. If you do not see your engine displayed, select Other Advanced server. Click Next.

b. In the Self Registration Services Admin Account dialog box, identify the the DMS Administrator by provide values for the Admin User Name, Admin Password and Admin Confirm Password fields and click Next.

   The user name and password that you enter here must match the DMS Admin values you set at the Policy Server.

   The DMS Administrator account secures DMS requests that are performed outside of the scope of a DMS administrator, such as self-registration. The user name and encrypted password for the account are stored in the dms.properties file on the Web Agent.

6. In the Web Server Configuration Summary dialog box, confirm that the configuration settings are correct, then click Install.

   The Web Agent files are installed.

7. Click Done when the installation is complete.

8. Enable the Web Agent:

   a. Open the WebAgent.conf file located in *web_agent_home*\bin\IIS

      Example: [set Access Path variable]\webagent\bin\IIS

   b. Set the value of the EnableWebAgent parameter to yes.

   c. Save the file and restart the web server.

   Note: You need to reboot the machine once the Agent is configured to ensure proper logging of Agent and trace messages.

Put the Agent Filter and Extension Before Other Third-Party Filters

The IIS 6.0 Web Agent consists of an ISAPI filter and an ISAPI extension. The majority of Web Agent processing occurs in the extension.

When the Web Agent is installed on an IIS 6.0 Web Server with other third-party software, such as WebSphere or ServletExec, the Agent has the following restrictions:

- The Web Agent filter and Web Agent extension must be configured to run before other third-party filters installed on the web server.
- The Web Agent must be configured as the first wildcard application map if it is going to protect applications running as or spawned by an ISAPI extension.
- The IIS 6.0 web server does not enforce how third-party filters and extensions behave. IIS 6.0 processes ISAPI filters before calling ISAPI extensions, including the Web Agent extension. Therefore, the SiteMinder Web Access Manager Web Agent for IIS 6.0 is unable to authenticate or authorize access to applications implemented as pure ISAPI filters. This limitation impacts Web Agent integration with other third-party offerings for the IIS 6.0 web server, if those offerings are implemented as ISAPI filters that process and/or redirect the request before ISAPI extensions are called.

When you install the Web Agent on an IIS 6.0 web server, the Agent's filter is automatically placed at the top of the ISAPI filters list. However, if you install any other third-party plugins after installing the Web Agent, those filters may take precedence.

After you install and configure an IIS 6.0 Web Agent, you must ensure that the siteminderagent ISAPI filter and extension is listed before any third-party filter or extension. This enables the Web Agent to process requests before a third-party.

To put the agent filter and extension before other third-party filters

1. Check the ISAPI filter by doing the following steps:
   a. Open the IIS Manager.
   b. Select Web Sites then right-click and select Properties.
   c. Select the ISAPI Filters tab.
   d. Check the list of filters and ensure that siteminderagent is the first entry in the list. If it is not, use the Move Up button to place it at the top of the list.
   e. Click OK.
   f. Exit the IIS Manager.
2. Check the ISAPI extensions by doing the following steps:

a. Open the IIS Manager, and then expand the web server.

b. Right-click the Default Web Site folder, and select Properties.

c. Click the Home Directory tab, and then click Configuration.

d. The following file should be at the top of the Wildcard application maps (order of implementation) field:

web_agent_home\bin\ISAPI6WebAgent.dll

web_agent_home

Indicates the directory where the Web Agent is installed.

Default (Windows installations): [set Access Path variable]\webagent

Default (UNIX installations): [set the Installation Path variable]/webagent

Configure the SOA Agent to Work With Sun Java System Web Server

Unless you installed the SOA Agent over a previously installed and configured Sun Java System Web Agent, you must configure the underlying Web Agent to work with the Sun Java System Web Server.

Configuration Methods for Sun Java System Web Agents on Windows Systems

The following configuration methods are available for Web Agents on Windows systems:

- GUI mode
- Unattended mode

More Information

Run the Configuration Wizard on Windows (see page 25)

Run the Configuration Wizard on Windows

To configure the Web Agent on an Sun Java System web server

1. If necessary, start the Web Agent Configuration Wizard.

   The default method is to select Start, Programs, SiteMinder Web Access Manager, Web Agent Configuration Wizard. If you have placed the Wizard shortcut in a non-default location, the procedure will be different.

   Important! If you are running this wizard on Windows Server 2008, run the executable file with Administrator permissions, even if you are logged into the system as an Administrator. For more information, see the release notes for your SiteMinder Web Access Manager component.

   Note: If you chose to configure the Web Agent immediately after the installation, SiteMinder Web Access Manager automatically starts the wizard automatically.

2. If you have already done host registration, skip to the next step. If not, select No in the Host Registration dialog box to skip registration, then click Next.

   To register a trusted host, go to the installation chapter for your platform.

3. Select the web server instances that you want to configure with Web Agents.

   If you have already configured a server with a Web Agent and you are running the Configuration Wizard to configure additional web servers instances, the Wizard displays the Select One or More Instances to Overwrite dialog box. This dialog box lists the web servers that you have previously configured.

   a. Select one of the following:

      Overwrite—to overwrite the server instance configuration.

      Preserve—to preserve the web servers configuration.

      Important! If you uncheck a previously configured server, the Web Agent will be removed from this server.

   b. Click Next.

4. In the Agent Configuration Object field, enter the name of the Agent Configuration Object for this web server instance, then click Next.

   This name must match an Agent Configuration Object already defined at the Policy Server. For example, to use the default enter iPlanetDefaultSettings.

5. If applicable, select one of the advanced SSL authentication schemes listed in the SSL Authentication dialog box. If the Agent is not providing advanced authentication, select No advanced authentication. Click Next.

   The selections are:

- HTTP Basic over SSL—identifies a user based on a user name and password. The credential delivery is always done over an encrypted Secure Sockets Layer (SSL) connection.
- X509 Client Certificate—identifies a user based on X.509 V3 client certificates. Digital certificates act as a signature for a user. Certificate authentication uses SSL communication.
- X509 Client Cert and HTTP Basic—combines X.509 Client Certificate and Basic authentication. The user's X.509 client certificate must be verified **and** he or she must provide a valid user name and password.
- X509 Client Cert or HTTP Basic—combines X.509 Client Certificate and Basic authentication. The user's X.509 client certificate must be verified, or he or she must provide a valid user name and password.
- X509 Client Cert or Form—The X.509 Client Certificate or HTML Forms authentication scheme combines the use of X.509 Client Certificates and the use of customized HTML forms to collect authentication information. Using this scheme, the user's X.509 client certificate must be verified **or** the user must provide the credentials requested by an HTML form.
- X509 Client Cert and Form—The X.509 Client Certificate and HTML Forms authentication scheme combines the use of X.509 Client Certificates and the use of customized HTML forms to collect authentication information. Using this scheme, the user's X.509 client certificate must be verified **and** the user must provide the credentials requested by an HTML form.

Note: For additional information about advanced authentication schemes, see the *Policy Server Configuration* Guide.

6. If you want to configure Self Registration for DMS2, select Yes. If not, select No.

   A servlet engine is required to run Self Registration. If the Web Agent Configuration Wizard does not detect a servlet engine, the Select Servlet Engine for Registration dialog box is not displayed.

   If you selected Yes to configure Self Registration:

   a. Select a servlet engine to be set up for the web server. If you do not see your engine displayed, select Other Advanced server. Click Next.

   b. In the Self Registration Services Admin Account dialog box, identify the the DMS Administrator by provide values for the Admin User Name, Admin Password and Admin Confirm Password fields and click Next.

      The user name and password that you enter here must match the DMS Admin values you set at the Policy Server.

      The DMS Administrator account secures DMS requests that are performed outside of the scope of a DMS administrator, such as self-registration. The user name and encrypted password for the account are stored in the dms.properties file on the Web Agent.

7. In the Web Server Configuration Summary dialog box. Confirm that the configuration settings are correct, then click Install.

   The Web Agent files are installed and the Configuration Complete dialog box displays.

8. Click Done to exit the Configuration Wizard.
9. Enable the Web Agent:
   a. Open the WebAgent.conf file, located in:

      Sun_Java_System_server_home\servers\https-*hostname*\config

   b. Set the EnableWebAgent parameter to Yes.
   c. Save the file.
10. Apply changes to Sun Java System Web Server files. This is required for the Agent's configuration to take effect.

More Information

Apply Changes to Sun Java System Web Server Files (see page 31)

Configuration Methods for Sun Java System Web Agents on UNIX Systems

The following configuration methods are available for Web Agents on UNIX systems:

- GUI mode
- Console mode
- Unattended mode

Configure Sun Java System Web Agents Using GUI or Console Mode

These instructions are for GUI and Console Mode configuration. The steps for the two modes are the same, with these exceptions for Console Mode:

- You may be instructed to select an option by entering a corresponding number. For example, to select the Sun Java System Web Server, you enter a 3, which corresponds to this server.
- Press ENTER after each step to proceed through the process instead of "clicking Next," as stated in the following procedure.
- All passwords that you enter are displayed in clear text. To workaround this issue, run the installation in GUI or unattended mode.

The prompts for each mode will help guide you through the process.

web_agent_home

Indicates the directory where the Web Agent is installed.

Default (Windows installations): [set Access Path variable]\webagent

Default (UNIX installations): [set the Installation Path variable]/webagent

To configure the Web Agent on a Sun Java System Web Server

1. If necessary, start the Configuration Wizard.
   a. Open a console window.
   b. Navigate to *web_agent_home*/install_config_info
   c. Enter one of the following commands:

      GUI mode: **.**/ca-pep-config.bin

      Console mode: ./ca-pep-config.bin -i console
2. If you have already done host registration, skip to the next step. Otherwise, select the option to skip host registration, then click Next.

   To register the trusted host, go to the installation chapter for your platform.
3. In the Select Web Server(s) dialog box, select the option for the iPlanet or Sun ONE Web Server and click Next.
4. Specify the root path where the Sun Java System web server is installed and click Next. For example, /opt/iPlanet/servers.

   You can click Choose to locate the root directory.
5. Select the web server instances that you want to configure with Web Agents.

If you have already configured a server with a Web Agent and you are running the Configuration Wizard to configure additional web servers instances, the Wizard displays the Select One or More Instances to Overwrite dialog box. This dialog box lists the web servers that you have previously configured.

a. Select one of the following:

   Overwrite—to overwrite the server instance configuration.

   Preserve—to preserve the web server's configuration.

   Important! If you uncheck a previously configured server, the Web Agent will be removed from this server.

b. Click Next.

6. In the Agent Configuration Object field, enter the name of the Agent Configuration Object for this web server instance.

   This name must match an Agent Configuration Object already defined at the Policy Server. For example, to use the default enter iPlanetDefaultSettings.

7. If applicable, select one of the advanced SSL authentication schemes listed in the SSL Authentication dialog box. If the Agent is not providing advanced authentication, select No advanced authentication. Click Next following your choice.

   The selections are:

   - HTTP Basic over SSL—identifies a user based on a user name and password. The credential delivery is always done over an encrypted Secure Sockets Layer (SSL) connection.
   - X509 Client Certificate—identifies a user based on X.509 V3 client certificates. Digital certificates act as a signature for a user. Certificate authentication uses SSL communication.
   - X509 Client Cert and HTTP Basic—combines X.509 Client Certificate and Basic authentication. The user's X.509 client certificate must be verified **and** he or she must provide a valid user name and password.
   - X509 Client Cert or HTTP Basic—combines X.509 Client Certificate and Basic authentication. The user's X.509 client certificate must be verified, or he or she must provide a valid user name and password.
   - X509 Client Cert or Form—The X.509 Client Certificate or HTML Forms authentication scheme combines the use of X.509 Client Certificates and the use of customized HTML forms to collect authentication information. Using this scheme, the user's X.509 client certificate must be verified **or** the user must provide the credentials requested by an HTML form.

- X509 Client Cert and Form—The X.509 Client Certificate and HTML Forms authentication scheme combines the use of X.509 Client Certificates and the use of customized HTML forms to collect authentication information. Using this scheme, the user's X.509 client certificate must be verified **and** the user must provide the credentials requested by an HTML form.

  Note: For more information, see the Policy Server documentation.

8. If you want to configure Self Registration for DMS2, select Yes. If not, select No.

   A servlet engine is required to run Self Registration. If the Web Agent Configuration Wizard does not detect a servlet engine, the Select Servlet Engine for Registration dialog box is not displayed.

   If you selected Yes to configure Self Registration:

   a. Select a servlet engine to be configured for the web server. If you do not see your engine displayed, select Other Advanced server. Click Next.

   b. In the Self Registration Services Admin Account dialog box, identify the the DMS Administrator by provide values for the Admin User Name, Admin Password and Admin Confirm Password fields and click Next.

      The user name and password that you enter here must match the DMS Admin values you set at the Policy Server.

      The DMS Administrator account secures DMS requests that are performed outside of the scope of a DMS administrator, such as self-registration. The user name and encrypted password for the account are stored in the dms.properties file on the Web Agent.

9. In the Web Server Configuration Summary dialog box. Confirm that the configuration settings are correct, then click Install.

   The Web Agent files are installed and the Configuration Complete message is displayed.

10. Click Done when the installation is complete.

11. Enable the Web Agent:

    a. Open the WebAgent.conf file, located in

       Sun_Java_System_server/servers/https-*hostname*/config

    b. Set the value of the EnableWebAgent parameter to Yes.

    c. Save the file.

    d. Restart the web server.

12. Apply changes to the Sun Java System Web Server files. This is required for the Agent's configuration to take effect.

More Information

Apply Changes to Sun Java System Web Server Files (see page 31)

Apply Changes to Sun Java System Web Server Files

The Web Agent Configuration Wizard makes changes to the Sun Java System server's magnus.conf, obj.conf, and mime.types files. If you plan to use the Sun Java System Administration console, you must apply the changes to these files *before* making any modifications with the console or the Web Agent configuration may be lost. If you lose your configuration, use the Configuration Wizard to reconfigure your Web Agent.

Note: The Web Agent adds settings to the Sun Java System's obj.conf file when the Agent is configured to support an advanced authentication scheme. SiteMinder Web Access Manager does not remove these settings later if the Agent is reconfigured to support a different advanced authentication scheme. Administrators must edit the obj.conf file manually to remove the settings that are no longer relevant.

To apply changes to the Sun Java System configuration files

1. Log on to the Sun Java System Administration Server console.
2. From the Servers tab, select the web server with the Web Agent installed and click Manage.
3. In the right corner of the dialog box, click Apply.

   You will see a warning message about loading the modified configuration files.
4. Click Load Configuration Files.
5. Exit the console.
6. Restart the web server.
7. Optimize the Sun Java System Web Agent by tuning the shared memory segments.

You may be required reboot your machine once the Agent is configured.

Configure the SOA Agent to Work With the Apache HTTP Server

Unless you installed the SOA Agent over a previously installed and configured Apache Web Agent, you must configure the underlying Web Agent to work with the Apache HTTP Server.

Configure an Apache Web Agent on Windows Systems

Before you configure the Agent, you may want to register the system where the Agent is installed as a trusted host; however, you can do this at a later time.

To configure the Apache Web Agent

1. If necessary, start the Web Agent Configuration Wizard.

   The default method is to select Start, Programs, SiteMinder Web Access Manager, Web Agent Configuration Wizard. If you've placed the Wizard shortcut in a non-default location, the procedure will be different.

   Important! If you are running this wizard on Windows Server 2008, run the executable file with Administrator permissions, even if you are logged into the system as an Administrator. For more information, see the release notes for your SiteMinder Web Access Manager component.

2. If you have already done host registration, skip to the next step. Otherwise, select the option to skip host registration, then click Next.

   To register the trusted host, go to the installation chapter for your platform.

3. In the Select Web Server(s) dialog box, select the radio button for the Apache Web Server and click Next.

4. In the Apache Web Server Path dialog box, specify the Apache web server root.

   If you installed the Agent on an Apache-based server, such as an IBM HTTP Server, or Oracle server, the Web Agent may not recognize the path. In this case, the Configuration Wizard displays the Apache Web Server Failure dialog box with the following options:

   - I would like to re-enter the Apache Server Root.

     Select this option for an Apache web server and re-enter the root path.

   - I would like to enter a specific configuration path.

     Select this option if you are using an Apache-based web server (such as, IBM HTTP, HP Apache-based, or Oracle). You are prompted to enter the full configuration path to the web server root.

   - I don't have an Apache web server.

     Choose this option to skip Apache configuration and continue with the Agent configuration.

   Click Next.

5. Following the server root path, specify the version of Apache you are using. Select from these two options:

   - Apache version 1.0
   - Apache version 2.0

6. Select the web server instances that you want to configure with Web Agents.

If you have already configured a server with a Web Agent and you are running the Configuration Wizard to configure additional web servers instances, the Wizard displays the Select One or More Instances to Overwrite dialog box. This dialog box lists the web servers that you have previously configured.

a. Select one of the following:

   Overwrite—to overwrite the server instance configuration.

   Preserve—to preserve the web servers configuration.

   Important! If you uncheck a previously configured server, the Web Agent will be removed from this server.

b. Click Next.

7. In the Agent Configuration Object field, enter the name of the Agent Configuration Object for this web server instance.

   This name must match an Agent Configuration Object already defined at the Policy Server. For example, to use the default enter ApacheDefaultSettings.

8. If you want to configure Self Registration for DMS2, select Yes. If not, select No and go to Step 9.

   A servlet engine is required to run Self Registration. If the Web Agent Configuration Wizard does not detect a servlet engine, the Select Servlet Engine for Registration dialog box is not displayed.

   If you selected Yes to configure Self Registration:

   a. Select a servlet engine to be configured for the web server. If you do not see your engine displayed, select Other Advanced server. Click Next.

   b. In the Self Registration Services Admin Account dialog box, identify the the DMS Administrator by provide values for the Admin User Name, Admin Password and Admin Confirm Password fields and click Next.

      The user name and password that you enter here must match the DMS Admin values you set at the Policy Server.

      The DMS Administrator account secures DMS requests that are performed outside of the scope of a DMS administrator, such as self-registration. The user name and encrypted password for the account are stored in the dms.properties file on the Web Agent.

9. In the Web Server Configuration Summary dialog box. Confirm that the configuration settings are correct, then click Install.

   The Web Agent files are installed.

10. Click Done when the installation is complete.

11. Enable the Web Agent:

    a. Open the WebAgent.conf file, located as follows:

       Apache_home\conf

where *Apache_home* is the installed location of the Apache web server.

b. Set the EnableWebAgent parameter to Yes.

c. Save and close the file.

12. Restart the web server.

When you run the Configuration Wizard for the Apache Web Agent, it makes changes to the Web Server's httpd.conf file and to the library path.

For httpd.conf changes to take effect, you need to restart the web server.

Configuration Methods for Apache Web Agents on UNIX Systems

The following configuration methods are available for Web Agents on UNIX systems:

- GUI mode
- Console mode
- Unattended mode

Notes:

- For the IBM HTTP web server, HP Apache-based web server, and Oracle HTTP web server, the Apache Web Agent is the Agent you should have installed. All the information for the Apache web server applies to those web servers also.
- Before you configure the Agent, you may want to register the system as a trusted host; however, you can do this at a later time.

Configure an Apache Web Agent Using GUI or Console Mode

These instructions are for GUI and Console Mode configuration. The steps for the two modes are the same, with these exceptions for Console Mode:

- You may be instructed to select an option by entering a corresponding number. For example, to select the Apache Web Server, you enter a 1, which corresponds to this server.
- Press ENTER after each step to proceed through the process instead of "clicking Next," as stated in the following procedure.
- All passwords that you enter are displayed in clear text. To workaround this issue, run the installation in GUI or unattended mode.

The prompts for each mode will help guide you through the process.

web_agent_home

Indicates the directory where the Web Agent is installed.

Default (Windows installations): [set Access Path variable]\webagent

Default (UNIX installations): [set the Installation Path variable]/webagent

To configure the Apache Web Agent

1. If necessary, start the Configuration Wizard.
   a. Open a console window.
   b. Navigate to *web_agent_home*/install_config_info
   c. Enter one of the following commands:

      GUI mode: **.**/ca-pep-config.bin

      Console mode: ./ca-pep-config.bin -i console
2. If you have already done host registration, skip to the next step. Otherwise, select the option to skip host registration, then click Next.

   To register the trusted host, go to the installation chapter for your platform.
3. In the Select Web server(s) dialog box, select the option for the Apache Web Sever and click Next.
4. In the Apache Web Server Path dialog box, specify the Apache Web Server root, for example, /opt/apache2. Click Next.

   If you installed the Agent on an Apache-based server, such as Covalent server, IBM HTTP Server, or Stronghold server, the Web Agent may not recognize the path. In this case, the Configuration Wizard displays the Apache Web Server Failure dialog box with the following options:
   - I would like to re-enter the Apache Server Root.

Select this option for an Apache web server and re-enter the root path.

- I would like to enter a specific configuration path.

  Select this option if you are using an Apache-based web server (Covalent, IBM HTTP, HP Apache-based, Oracle, Stronghold). You are prompted to enter the full configuration path to the web server root.

- I don't have an Apache web server.

  Choose this option to skip Apache configuration and continue with the Agent configuration.

Click Next.

5. Following the server root path, specify the version of Apache you are using. Select from these two options:

   - Apache version 1.0
   - Apache version 2.0

6. Select the web server instances that you want to configure with Web Agents.

   If you have already configured a server with a Web Agent and you are running the Configuration Wizard to configure additional web servers instances, the Wizard displays the Select One or More Instances to Overwrite dialog box. This dialog box lists the web servers that you have previously configured.

   a. Select one of the following:

      Overwrite—to overwrite the server instance configuration.

      Preserve—to preserve the web servers configuration.

      Important! If you uncheck a previously configured server, the Web Agent will be removed from this server.

   b. Click Next.

7. In the Agent Configuration Object field, enter the name of the Agent Configuration Object for this web server instance.

   This name must match an Agent Configuration Object already defined at the Policy Server. For example, to use the default enter ApacheDefaultSettings.

8. If you want to configure Self Registration for DMS2, select Yes. If not, select No.

   A servlet engine is required to run Self Registration. If the Web Agent Configuration Wizard does not detect a servlet engine, the Select Servlet Engine for Registration dialog box is not displayed.

   If you selected Yes to configure Self Registration:

   a. Select a servlet engine to be configured for the web server. If you do not see your engine displayed, select Other Advanced server. Click Next.

b. In the Self Registration Services Admin Account dialog box, identify the the DMS Administrator by provide values for the Admin User Name, Admin Password and Admin Confirm Password fields and click Next.

The user name and password that you enter here must match the DMS Admin values you set at the Policy Server.

The DMS Administrator account secures DMS requests that are performed outside of the scope of a DMS administrator, such as self-registration. The user name and encrypted password for the account are stored in the dms.properties file on the Web Agent.

9. In the Web Server Configuration Summary dialog box. Confirm that the configuration settings are correct, then click Install.

The Web Agent files are installed.

10. Click Done when the installation is complete.
11. Enable the Web Agent:
    a. Open the WebAgent.conf file, located as follows:

    Apache_home/conf

    b. Set the EnableWebAgent parameter to yes.
    c. Save and close the file.
12. Restart the web server.

When you run the Configuration Wizard for the Apache Web Agent, it makes changes to the Web Server's httpd.conf file and to the library path.

For httpd.conf changes to take effect, you need to restart the web server.

13. For Apache on UNIX systems, optimize the Apache Web Agent by tuning the shared memory segments.

Set Apache Run As Identity as Owner of /webagent Directory on UNIX Systems

To ensure that the SOA Agent functions correctly with Apache HTTP Server on UNIX systems, you must set the user with which Apache is run (such as root, nobody, and so on) as the owner of the *SOA_HOME*/webagent directory.

SOA_HOME

Specifies the path to the SOA Security Manager installation directory.

Improve Server Performance with Optional httpd.conf File Changes

You can improve server performance by modifying the default configuration settings in the httpd.conf file; however, these changes are *not* required:

To improve server performance with optional httpd.conf file changes

1. For Apache and Sun Java System servers, assign a higher priority level to your Apache20WebAgent.dll file than any other auth or access modules installed in your server's configuration.
2. For low-traffic websites, define the following directives:
   - Set MaxRequestsPerChild>1000 *or* Set MaxRequestsPerChild=0
   - MinSpareServers >5
   - MaxSpareServers>10
   - StartServers=MinSpareServers>5
3. For high-traffic websites, define the following directives:
   - Set MaxRequestsPerChild>3000 *or* Set MaxRequestsPerChild=0
   - MinSpareServers >10
   - MaxSpareServers>15
   - StartServers=MinSpareServers>10

Note: CA Services can provide assistance with performance-tuning for your particular environment.

Set the LD_PRELOAD Variable for Apache Agent Operation

The LD_PRELOAD variable needs to be defined for the Apache Web Agent to operate on different platforms.

More Information

Set the LD_PRELOAD Variable for Apache Web Server on SUSE Linux 98 (see page 39)
Set the LD_PRELOAD Variable for SSL Configuration on an IBM HTTP Server 2.0.47/Linux AS 3.0 System (see page 39)

Set the LD_PRELOAD Variable for Apache Web Server on SUSE Linux 98

After you install the Web Agent [insert SiteMinder version number] on an Apache web server running on SUSE 9, you must set the LD_PRELOAD environment variable as follows:

LD_PRELOAD=*web_agent_home*/bin/libbtunicode.so

Without this setting, two problems may occur:

- The Apache web server may dump core upon shutdown.
- When a SAML Affiliate Agent is running on an Apache Web Server, a load change may spawn multiple processes that eventually consume 100% of the CPU cycles.

Note: Setting this environment variable causes any application executed from that environment to bind with libbtunicode.so. Therefore, only set this variable when starting or stopping a web server that loads the SiteMinder Web Access Manager Web Agent.

Set the LD_PRELOAD Variable for SSL Configuration on an IBM HTTP Server 2.0.47/Linux AS 3.0 System

When configuring SSL on an IBM HTTP Server 2.0.47.1 running SUSE8, ikeyman, the graphical user interface for the IBM key management utility, crashes when it is used to create a key database.

To resolve this issue, set the following environment variable:

export LD_PRELOAD=/usr/lib/libstdc++-libc6.2-2.so.3

After setting this variable, the key database file, key.kdb, is created successfully.

Set the LD_ASSUME_KERNEL for Apache Agent on SuSE Linux 9 for zSeries

After you install the Web Agent on an Apache web server running on SuSE Linux 9 for zSeries, set the LD_ASSUME_KERNEL environment variable as follows:

LD_ASSUME_KERNEL=2.4.21

export LD_ASSUME_KERNEL

Important! You must set this variable to 2.4.21 because it represents the kernel release upon which the Web Agent libraries are built.

Without this setting, the following problems occur:

- The Apache web server will not start properly.
- Host registration dumps core.

Configure Apache for Oracle 9.0.2/9.0.3 HTTP Server

Oracle HTTP Server (OHS) is based on the Apache Web Server and is protected using the Apache Web Agent.

Prerequisites:

- Before you configure the Apache Agent for OHS, you should have already installed the Oracle Server 9.0.2/9.0.3 on a system running Solaris 9 or HP-UX 11i.
- For HP-UX 11i systems, install patch a C++ runtime A.03.30 or later before installing the Oracle server.

To configure the Apache Web Agent for Oracle HTTP Server, you must:

1. Optionally, Install a Server Certificate (Required for SSL Only). For SSL-enabled Oracle HTTP Servers, you must install a server certificate using the Oracle Wallet Manager application. For more information, see the documentation supplied with your Oracle HTTP Server.
2. Configure the Apache Web Agent for the Oracle Server
3. Restart the Web Server as follows:
   a. Stop the Web server from *OHS_home*/dcm/bin by executing the command:

      dcmctl stop

   b. Restart the Web Server by executing the command:

      dcmctl start –ct ohs –v

Note: When you configure an Apache Web Agent on an Oracle HTTP Server, the Configuration Wizard makes changes to the server's httpd.conf file.

Modify the http.conf File for Apache Reverse Proxy Server

A reverse proxy server acts on behalf of incoming requests to a company's internal network as opposed to outgoing requests from a private network to the Internet. When a request for a resource is received by a reverse proxy server, the request is directed to a server behind the firewall, as opposed to redirecting the request to a remote server over the Internet.

To successfully implement a reverse proxy solution, modify the http.conf file by adding ProxyPass and ProxyPassReverse directives.

The ProxyPass directive allows remote servers to be mapped to the space of the local server. The local server essentially mirrors the specified remote server.

- The ProxyPass directive takes the following form:

  ProxyPass *path URL*

  The *path* is the name of the local virtual path, and the *URL* is a partial URL for the remote server. For example:

  ProxyPass /realma/ http://server.myorg.org/realma/

- The ProxyPassReverse directive allows Apache to adjust the Location header on the HTTP redirect responses. This directive takes the following form:

  ProxyPassReverse *path URL*

  The *path* is the name of the local virtual path, and the *URL* is a partial URL for the remote server. For example:

  ProxyPassReverse /realma/ http://server.myorg.org/realma/

Configure the SOA Agent to Work With the Domino Web Server

Unless you installed the SOA Agent over a previously installed and configured Domino Web Agent, you must configure the underlying Web Agent to work with the Domino Web Server.

Run the Configuration Wizard for a Domino Web Agent on Windows

Before you configure the Agent, you may want to register the system where the Agent is installed as a trusted host; however, you can do this at a later time.

web_agent_home

Indicates the directory where the Web Agent is installed.

Default (Windows installations): [set Access Path variable]\webagent

Default (UNIX installations): [set the Installation Path variable]/webagent

To configure a Domino Web Agent

1. If necessary, start the Web Agent Configuration Wizard.

   The default method is to select Start, Programs, SiteMinder Web Access Manager, Web Agent Configuration Wizard. If you've placed the Wizard shortcut in a non-default location, the procedure will be different.

   Important! If you are running this wizard on Windows Server 2008, run the executable file with Administrator permissions, even if you are logged into the system as an Administrator. For more information, see the release notes for your SiteMinder Web Access Manager component.

2. If you have already done host registration, skip to the next step. Otherwise, select the option to skip host registration, then click Next.

   For information on registering the trusted host, see the installation chapter for your platform.

3. In the Select the Web server(s) dialog box, select the radio button for the Domino Web Sever and click Next.

4. In the Domino Web Server Path dialog box, specify the location of the notes.ini file, such as C:\Lotus\Domino\notesdata, then click Next.

   Note: The installation automatically writes the path to the WebAgent.conf in the notes.ini file.

5. Select the Web server instances that you want to configure with Web Agents.

   If you have already configured a server with a Web Agent and you are running the Configuration Wizard to configure additional Web servers instances, the Wizard displays the Select One or More Instances to Overwrite dialog box. This dialog box lists the Web servers that you have previously configured.

   a. Select one of the following:

      - Overwrite--replaces the existing configuration of server instance with the new one.
      - Preserve--keeps the existing web server's configuration without changing it.

Important! If you uncheck a previously configured server, the Web Agent will be removed from this server.

b. Click Next.

6. In the Agent Configuration Object field, enter the name of the Agent Configuration Object for this Web server instance, then click Next.

   This name must match an Agent Configuration Object already defined at the Policy Server. For example, to use the default enter DominoDefaultSettings.

7. In the Web Server Configuration Summary dialog box, confirm that the configuration settings are correct, then click Install.

   The Web Agent files are installed.

8. Click Done when the installation is complete.

9. Enable the Web Agent:

   a. Open the WebAgent.conf file, located in where you installed the Domino Web server root directory.

   b. Set the EnableWebAgent parameter to YES.

   c. Save the file.

Add the Domino Web Agent DLL (Windows)

To make the Domino Web Agent operate properly, you must add the DOMINOWebAgent.dll file to the filter DLLs. The Web Agent DLL must be the first DLL in the list.

To add the Domino Web Agent DLL

1. Open Lotus Notes.
2. Select File, Database, Open.
3. In the Server field, select the Domino Server where you installed the Web Agent.
4. In the Database scroll box, select the server's address book.

   In the Filename field you should see names.nsf displayed.
5. Click Open.

   The server's address book opens.
6. In the left pane, expand the Server folder and double-click on the All Server Documents icon.
7. Select your server and click Edit Server.

   The Domino server's administration console opens.
8. Select the Internet Protocols tab.
9. In the DSAPI section of the window, find the DSAPI filter file names field and enter the full path to the Domino Web Agent DLL, for example:

   [set Access Path variable]webagent\bin\
   DOMINOWebAgent.dll

   Note: This entry should be the first in the list of filters.
10. Click Save and Close.
11. Restart the web server.

    You may be required to reboot your machine after the Agent is configured.

Configuration Methods for Domino Web Agents on UNIX Systems

The following configuration methods are available for Web Agents on UNIX systems:

- GUI mode
- Console mode
- Unattended mode

Configure Domino Web Agents in GUI or Console Mode

These instructions are for GUI and Console Mode configuration. The steps for the two modes are the same, with these exceptions for Console Mode:

- You may be instructed to select an option by entering a corresponding number. For example, to select the Apache Web Server, you enter a 1, which corresponds to this server.
- Press ENTER after each step to proceed through the process instead of "clicking Next," as stated in the following procedure.
- All passwords that you enter are displayed in clear text. To workaround this issue, run the installation in GUI or unattended mode.

The prompts for each mode will help guide you through the process.

1. If necessary, start the Configuration Wizard.
   a. Open a console window.
   b. Navigate to *web_agent_home*/install_config_info
   c. Enter one of the following commands:

      GUI mode: **.**/ca-pep-config.bin

      Console mode: ./ca-pep-config.bin -i console

   Note: If you chose to configure the Web Agent immediately after the installation, SiteMinder Web Access Manager automatically starts the Configuration Wizard.
2. If you have already done host registration, skip to the next step. Otherwise, select the option to skip host registration, then click Next.

   To register the trusted host, go to the installation chapter for your platform.
3. In the Select the Web server(s) dialog box, select the radio button for the Domino Web Sever and click Next.
4. In the Domino Web Server Path dialog box, specify the location of the notes.ini file, such as /local/notesdata, then click Next.

   Note: The installation automatically writes the path to the WebAgent.conf in the notes.ini file.
5. Select the web server instances that you want to configure with Web Agents.

   If you have already configured a server with a Web Agent and you are running the Configuration Wizard to configure additional web servers instances, the Wizard displays the Select One or More Instances to Overwrite dialog box. This dialog box lists the web servers that you have previously configured.
   a. Select one of the following:

      Overwrite—to overwrite the server instance configuration.

Preserve—to preserve the web servers configuration.

Important! If you uncheck a previously configured server, the Web Agent will be removed from this server.

b. Click Next.

6. In the Agent Configuration Object field, enter the name of the Agent Configuration Object for this web server instance.

   This name must match an Agent Configuration Object already defined at the Policy Server. For example, to use the default enter DominoDefaultSettings.

7. In the Web Server Configuration Summary dialog box. Confirm that the configuration settings are correct, then click Install.

   The Web Agent files are installed.

8. Click Done when the installation is complete.

9. Enable the Web Agent:

   a. Open the WebAgent.conf file, located in the Domino web server root directory.

   b. Set the EnableWebAgent parameter to Yes.

   c. Save the file.

10. If your Domino web server runs on AIX, you must enable run time linking after configuring your agent (you only need to do this once) with the following steps:

    a. Run the following commands:

    ```
    cd domino_server_path/lotus/notes/latest/ibmpow
    /usr/bin/rtl_enable ./http -brtl
    mv ./http ./http.orig
    mv ./http.new ./http
    ```

    b. Start the Domino web server.

Add the Domino Web Agent DLL (UNIX)

For the Domino Web Agent to operate properly, you must add the dominowebagent.so library to the filter DLLs. This library must be first in the list.

web_agent_home

Indicates the directory where the Web Agent is installed.

Default (Windows installations): [set Access Path variable]\webagent

Default (UNIX installations): [set the Installation Path variable]/webagent

To add the Domino Web Agent DLL

1. Open Lotus Notes.
2. Select File, Database, Open.
3. In the Server field, select the Domino Server where you installed the Web Agent.
4. In the Database scroll box, select the server's address book.

   In the Filename field you should see names.nsf displayed.
5. Click Open.

   The server's address book opens.
6. In the left pane, expand the Server folder and double-click on the All Server Documents icon.
7. Select your server and click Edit Server.

   The Domino server's administration console opens.
8. Select the Internet Protocols tab.
9. In the DSAPI section of the window, find the DSAPI filter file names field and enter the full path to the Domino Web Agent file, for example:

   web_agent_home>/bin/dominowebagent.so

   Note: This entry should be the first in the list of filters.
10. Click Save and Close.
11. Restart the web server.

Configuration Procedures That Apply for All Web Agent Types

The following configuration procedures apply for all Web Agent types.

Check SmHost.conf File Permissions for Shared Secret Rollover

If you enable shared secret rollover, the user who owns the web server process must have permissions to write to the SmHost.conf file. If this file cannot be modified by this user, then the shared secret rollover cannot be updated.

For example, for Sun Java System and Apache web servers, the person specified by the User directive needs write permission to the SmHost.conf file. If the SmHost.conf file is owned by User1 and no other user has write permissions, the shared secret rollover cannot be updated if User2 owns the server process.

Reconfigure a Web Agent

Reconfigure a Web Agent for the following reasons:

- You have upgraded the Web Agent and now you need to update the configuration
- You need to change the configuration settings previously defined for a Web Agent
- You need to remove the configuration settings from the Web Agent without uninstalling the entire Web Agent (you would need to configure the Web Agent again at a later time)
- You want to configure the Web Agent for a different Web Server installed on the same system as the configured server.

To reconfigure a Web Agent in any mode, re-run the Configuration Wizard. There are no additional steps or prompts for reconfiguring an Agent.

How to Set Up Additional Agent Components

The Web Agent Configuration Wizard guides you through basic Agent configuration. However, there are other Agent components that you can configure without the wizard.

All SiteMinder Web Access Manager Web Agents can protect resources, act as forms credential collectors (FCC) and/or an SSL credential collectors (SCC), and serve as a cookie provider for single sign-on. The Web Agent can serve in one or more of these roles simultaneously.

At installation, some of these functions, such as acting as the forms credential collector, are set up automatically; however, other capabilities, such as the cookie provider require additional configuration.

You can set up any of the additional components as follows:

- Configuring an Agent as a forms credential collector

  The libraries and files for forms credential collection are set up automatically during installation.

- Configuring an Agent as an SSL credential collector

  You specify whether the Agent performs SSL credential collection during the initial Agent configuration with the Configuration Wizard.

- Configuring the Agent as a cookie provider for multiple cookie domain single sign-on

  A cookie provider lets the Agent implement single sign-on in a multiple cookie domain environment. All Web Agents can act as a cookie provider, but all cookie providers within a domain must use the same domain name. The cookie provider URL setting in the Agent's configuration dictates which Web Agent is the cookie provider. After you determine which Agent is the cookie provider, you configure all other Agents in the single sign-on environment to point to the cookie provider by entering that Agent's URL.

Configure SOA Agent for Web Servers to Enable Fine-Grain Resource Identification

By default, the SOA Agent for Web Servers identifies incoming requests for web service resources as follows:

[*URL*][*Web Service Name*]

However, the SOA Agent for Web Servers can be configured to provide fine-grain resource identification, additionally identifying the requested web service operation name, so that requests are identified as:

[*URL*][*Web Service Name*][*Web Service Operation*]

This allows you to define fine-grain policies that include the web service operation in authorization decisions, but may adversely affect transaction performance.

Note: For more information on configuring fine-grain authorization policies, see the *CA SOA Security Manager Policy Configuration Guide*.

To configure a SOA Agent to identify web service operations for fine-grain authorization

1. Ensure that the XMLSDKResourceIdentification Agent configuration parameter is present and set to Yes for the target SOA Agent for Web Servers.
2. Edit the XmlToolkit.properties file located in SOA_Agent_Install\java to ensure that the WSDMResourceIdentification entry is present and set to "Yes".
3. Save and close the XmlToolkit.properties file.
4. Restart the target SOA Agent for Web Servers.

Note: You must enable fine-grain resource identification to use the Administrative UI to generate policies for web service resources protected by the SOA Agent for Web Servers from their associated WSDL files.

Configure the Username and Password Digest Token Age Restriction

By default, the WS-Security authentication scheme imposes a 60-minute restriction on the age of Username and Password Digest Tokens to protect against replay attacks.

To configure a different value for the token age restriction for a SOA Agent for Web Servers, add the WS_UT_CREATION_EXPIRATION_MINUTES parameter to the XmlToolkit.properties file for that agent.

To configure a SOA Agent to use a nondefault age restriction for Username and Password Digest token authentication

1. Navigate to *SOA_Agent_Install\java.*
2. Open XmlToolkit.properties in a text editor.
3. Add the following line:

   WS_UT_CREATION_EXPIRATION_MINUTES=*token_age_limit*

   token_age_limit

   Specifies the token age limit restriction in minutes.

4. Save and close the XmlToolkit.properties file.
5. Restart the SOA Agent.

XML Processing Message Logging

In addition to Web Agent logging functionality, the SOA Agent for Web Servers provides an additional level of log information relating specifically to its processing of XML messages. SOA Agent for Web Servers logging is implemented using Apache's *log4j* standard (see http://logging.apache.org).

Note: SOA Agent for Web Servers logging does not start until an XML message that needs to be processed is received.

By default, SOA Agent for Web Servers logging is enabled and written to the soasm_agent.log file in:

- Windows—*Agent_install_location*\bin\
- UNIX—*Agent_install_location*/bin/

You can change logging parameters for your SOA Agent for Web Servers by editing the log.config file, which can be found in:

- Windows—*Agent_install_location*\config\
- UNIX— *Agent_install_location*/config/

Configure the SOA Agent for Web Servers to Process Large XML Messages

By default, the SOA Agent for Web Servers can process XML messages up to 2000 KB in size. However, if you need to process larger files, you can increase this size limit by editing the conapi.conf file on each system hosting a SOA Agent for Web Servers.

The conapi.conf is located in:

- *web_agent_install_dir*\config (Windows)
- *web_agent_install_dir*/config (UNIX)

To increase the maximum message size, uncomment the nete.conapi.service.xmlsdk.maxpacketsize entry and change its value to the maximum message size (in KB) you want the SOA Agent for Web Servers to be able to process. For example:

```
nete.conapi.service.xmlsdk.maxpacketsize=3000
```

Note: Web servers also have incoming file size limits. When planning your Web service implementations, you should ensure that you are aware of these and bear them in mind.

Manage the SOA Agent for Web Servers

The SOA Agent for Web Servers is operationally equivalent to an CA SiteMinder Web Agent. That is, it can be considered the same as a Web Agent in terms of:

- **Agent/Policy Server time calculation**—The SOA Agent for Web Servers calculates time relative to Greenwich Mean Time (GMT).
- **Agent key management**—The SOA Agent for Web Servers uses Agent keys and must be included in key rollovers.

Note: For more information, see the *CA SiteMinder Agent Guide*.

Start and Stop the XML SDK Server Process

The SOA Security Manager XML SDK server is a process that must be running so the SOA Agent for Web Servers can process requests. The XML SDK Server should be started automatically at system startup. This section describes how to start and stop it manually.

Note: Do not confuse the XML SDK server with the SOA Security Manager SDK, which is an API that communicates with the XML SDK server.

To start the SOA Security Manager XML SDK server

- On Windows:
  1. Open the Services dialog.
  2. Right-click TxMinder XML SDK Service entry and then click Start in the menu that opens.
- On UNIX, navigate to *SOA_HOME*/webagent/bin and enter tmxmlsdkserver -start

To stop the SOA Security Manager XML SDK server

- On Windows:
  1. Open the Services dialog.
  2. Right-click TxMinder XML SDK Service entry and then click Stop in the menu that opens.
- On UNIX, navigate to *SOA_HOME*/webagent/bin and enter tmxmlsdkserver -stop

SOA Agent for Web Servers Logging

Being an XML-enabled version of the CA SiteMinder Web Agent, the SOA Agent for Web Servers provides the same logging functionality provided by all CA SiteMinder Agents.

Note: For more information, see the *CA SiteMinder Agent Guide*.

Configure Trace Logging for the SOA Agent for Web Servers

To enable trace logging for the SOA Agent for Web Servers, include a "components:" entry with value "XMLAgent" in the Trace Configuration File (WebAgentTrace.conf). For example:

```
# For TxM XML agent
components: XMLAgent
data: Date, Time, Pid, Tid, TransactionID, Function, Message
```

Note: For more information about configuring the Trace Configuration File, see the *CA SiteMinder Agent Guide*.

XML Processing Message Logging

In addition to Web Agent logging functionality, the SOA Agent for Web Servers provides an additional level of log information relating specifically to its processing of XML messages. SOA Agent for Web Servers logging is implemented using Apache's *log4j* standard (see http://logging.apache.org).

Note: SOA Agent for Web Servers logging does not start until an XML message that needs to be processed is received.

By default, SOA Agent for Web Servers logging is enabled and written to the tmxmltoolkit.log file in:

- Windows—*Agent_install_location*\bin\
- UNIX—*Agent_install_location*/bin/

You can change logging parameters for your SOA Agent for Web Servers by editing the log.config file, which can be found in:

- Windows—*Agent_install_location*\config\
- UNIX— *Agent_install_location*/config/

XML Processing Message Logging

In addition to Web Agent logging functionality, the SOA Agent for Web Servers provides an additional level of log information relating specifically to its processing of XML messages. SOA Agent for Web Servers logging is implemented using Apache's *log4j* standard (see http://logging.apache.org).

Note: SOA Agent for Web Servers logging does not start until an XML message that needs to be processed is received.

By default, SOA Agent for Web Servers logging is enabled and written to the soasm_agent.log file in:

- Windows—*Agent_install_location*\bin\
- UNIX—*Agent_install_location*/bin/

You can change logging parameters for your SOA Agent for Web Servers by editing the log.config file, which can be found in:

- Windows—*Agent_install_location*\config\
- UNIX— *Agent_install_location*/config/

Custom SOA Agent for Web Servers

While the SOA Agent for Web Servers works with the standard features of SOA Security Manager, you can extend Agent functionality by creating a custom Agent. You create a custom SOA Agent for Web Servers using the Agent APIs provided by the CA SiteMinder SDK and the SOA Security Manager SDK and then configure it by using the Policy Server User Interface.

Custom Agents work with the CA Policy Server to control access to a wide range of resources beyond Web resources. For example, custom Agents could control access to a software architecture method, an application, or a task performed by an application.

Together with the Policy Server, the SOA Agent for Web Servers protects Web resources that can be identified by a URL. Because the Policy Server is a general-purpose rules engine, it can protect:

- Any resource that can be expressed as a string
- Any operation on a resource.

Consequently, a custom Agent, working with the Policy Server as the core engine, can extend the types of resources that CA SiteMinder can protect.

Using the CA SiteMinder and SOA Security Manager Agent APIs, you can create a custom SOA Agent for Web Servers to implement security for any type of resource. API functionality for creating a custom SOA Agent for Web Servers includes:

- Protection inquiry—Determines whether or not a resource is protected by the Policy Server.
- XML document authentication—Determines whether or not a given XML document has been authenticated. If not, the XML document is authenticated based on the scheme associated with the requested Web service.
- XML document and resource authorization—Determines whether or not the authenticated XML document is authorized to be passed on to the requested Web service.

For detailed information about creating a custom SOA Agent for Web Servers, see the following guides:

- *SOA Security Manager Programming Guide*
- *SiteMinder Web Access Manager Programming Guide for C*
- *SiteMinder Web Access Manager Programming Guide for Java*

Chapter 2: SOA Agent for IBM WebSphere

This section contains the following topics:

SOA Agent for IBM WebSphere Introduction

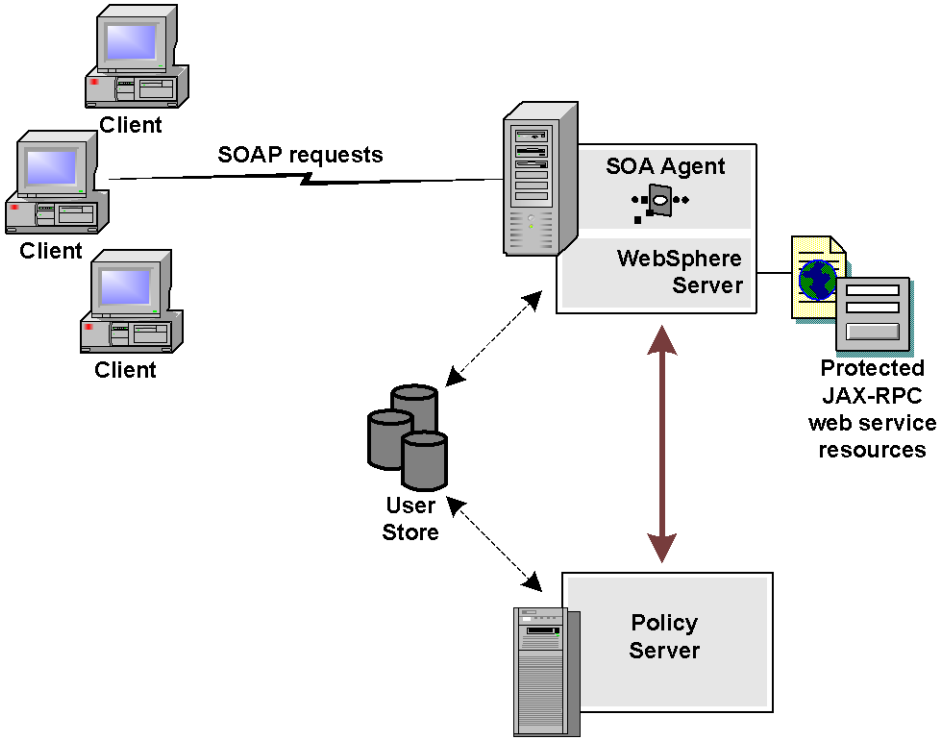
The SOA Agent for IBM WebSphere provides a SOA Security Manager-based access control solution for IBM WebSphere Application Server. The SOA Agent integrates the WebSphere Application Server into the SOA Security Manager environment, enabling you to implement policy-based access control to protect WebSphere-hosted JAX-RPC web services.

SOA Agent for IBM WebSphere Overview

The SOA Agent for IBM WebSphere resides in a WebSphere Application Server, enabling you to protect WebSphere-hosted JAX-RPC web service resources.

The SOA Agent for IBM WebSphere intercepts all SOAP messages sent over HTTP or HTTPS transport to JAX-RPC web services deployed on the Websphere Application Server. The SOA Agent then communicates with the Policy Server to authenticate and authorize the message sender and, upon successful authentication and authorization, passes the SOAP message on to the addressed web service.

A high-level overview of the SOA Agent for IBM WebSphere Server architecture is shown in the following figure.



The SOA Agent for IBM WebSphere provides the following features:

- SOA Security Manager Integration with the J2EE platform
- Fine-grained access control of JAX-RPC web service resources
- Support for bi-directional SOA Security Manager/SiteMinder and WebSphere single sign-on (SSO)
- Support for WebSphere clustering

The SOA Agent additionally supports:

- J2EE RunAs identity
- Multi-byte character usernames
- User mapping to support environments in which WebSphere and SOA Security Manager are not configured to use the same user store
- Centralized and dynamic agent configurations
- Caching of resource protection decisions and authentication and authorization decisions
- Logging
- Authorization auditing

Required Background Information

This guide assumes that you have the following technical knowledge:

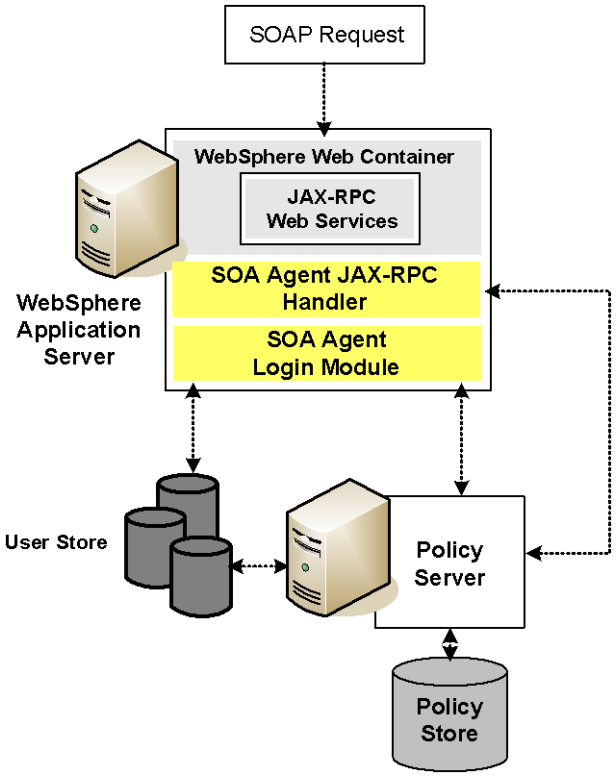
- An understanding of Java, J2EE standards, J2EE application servers, and multi-tier architecture
- An understanding of JAX-RPC web service implementations and JAX-RPC handlers
- Experience with the IBM WebSphere Application Server Version 6.1, its architecture and security infrastructure.
- Familiarity with Java Authentication and Authorization Server (JAAS) and WebSphere security-related topics
- Familiarity with SOA Security Manager concepts, terms, and Policy Server configuration tasks

Additionally, to effectively plan your security infrastructure, you must be familiar with the web services that you plan to protect with SOA Security Manager.

SOA Agent for IBM WebSphere Components

The SOA Agent for IBM WebSphere consists of two modules that plug into WebSphere's security infrastructure.

- SOA Agent JAX-RPC Handler (see page 60)
- SOA Agent Login Module (see page 61)



SOA Agent JAX-RPC Handler

The SOA Agent JAX-RPC Handler is a custom JAX-RPC Handler that, when added to the deployment descriptor of a JAX-RPC web service, intercepts SOAP message requests for JAX-RPC web services and diverts them to the SOA Agent Login Module for authentication and authorization decisions.

SOA Agent Login Module

The SOA Agent Login Module is a JAAS Login Module that performs authentication and authorization for JAX-RPC web services protected by the SOA Agent for IBM WebSphere.

The SOA Agent Login Module authenticates credentials obtained from the following request types against associated user directories configured in SOA Security Manager:

- SOAP requests intercepted by the SOA Agent JAX-RPC Handler .
- Requests for web service resources from users with pre-established SOA Security Manager and SiteMinder sessions (validating the session and obtaining user names from associated SiteMinder session ticket cookies)
- System login (such as J2EE RunAs identity) requests.

If SOA Security Manager authentication is successful, the SOA Agent Login Module populates a JAAS Subject with a SOA Security Manager Principal that contains the username and associated SOA Security Manager session data.

The SOA Agent Login Module then determines whether an authenticated user is allowed to access a protected WebSphere resource, based on associated SOA Security Manager authorization policies.

Recommended Reading List

To learn about the WebSphere Application Server and Java, see the following resources:

- IBM Redbooks Online

  http://www.redbooks.ibm.com/Redbooks.nsf/redbooks/
- IBM WebSphere Application Server Information Center

  http://www-306.ibm.com/software/webservers/appserv/was/
- Sun Microsystems, Inc., online documentation

  http://java.sun.com.

Installation Location References

In this guide:

- *SOA_HOME* refers to the installed location of the SOA Agent for IBM WebSphere.
- *WS_HOME* refers to the installed location of the WebSphere Application Server.

Configure the SOA Agent

To configure the SOA Agent, you must specify the following:

- Host Configuration Object (one for each host server)
- Agent Configuration Object (one for each SOA Agent)
- Agent identity (one for each SOA Agent)

Note: For detailed information about how to configure Agent-related objects, see *the SOA Security Manager Policy Configuration Guide* and the *SOA Security Manager Implementation Guide*.

To configure the SOA Agent

1. On the Policy Server:
   a. Duplicate or create a Host Configuration Object, which holds initialization parameters for a Trusted Host.

      The Trusted Host is a server that hosts one or more Agents and handles their connection to the Policy Server.

   b. As necessary, add or edit Trusted Host parameters in the Host Configuration Object that you just created.

   c. Duplicate or create an Agent Configuration Object, which holds Agent configuration parameters and can be used to centrally configure a group of Agents.

   d. Add or edit required Agent parameters in the Agent Configuration Object.

      The configuration object must include the DefaultAgentName or AgentName parameter to specify the Agent identity.

   e. Create an Agent identity for the SOA Agent. You must select *Web Agent* as the Agent type for a SOA Agent.
2. On the system where the SOA Agent is installed:
   a. Run the Agent Configuration Wizard, which registers the Trusted Host.

   b. Enable the SOA Agent by setting the EnableWebAgent parameter in the Agent configuration file to Yes.

SOA Agent for WebSphere Configuration File

By default, the SOA Agent for WebSphere installation creates a single agent configuration file, JavaAgent.conf.

The agent configuration file is located in the *SOA_HOME*/config directory, where *SOA_HOME* is the location where you installed the SOA Agent. For example:

- For Windows

  C:\SoaSecurityManager\config

- For UNIX

  /opt/SoaSecurityManager/config

Each Agent configuration file is created with the following required default configuration parameters/values:

Parameter	Description
DefaultAgentName	The agent identity the Policy Server uses to associate policies with the SOA Agent. The default value is "SoaAgent". Do not change this value.
EnableAgent	Specifies whether the SOA Agent is enabled. Possible values are Yes and No. Default value is Yes.
AgentConfigObject	The Agent Configuration Object specified during installation.
SmHostFile	Path to the local Host Configuration File. Path can be specified in absolute terms or relative to *SOA_HOME.* **Note:** On Windows, you must specify paths using double backslashes ("\\") rather than single backslash ("\") to separate directories. On UNIX, use standard single slash ("/") separators. Example values: ■ (Windows) C:\\Program Files\\CA\\SOASecurityManager\\soaagent\\wasagent\\config\\SmHost.conf ■ (Windows) config\\SmHost.conf ■ (UNIX) export/soaagent/wasagent/config/SmHost.conf ■ (UNIX) /config/SmHost.conf
ServerName	A string that will be used in the SOA Agent log to identify the WebSphere Server. Default value is "SOAWAS61".
appserverjaasloginhandler	Specifies the Application Server-specific SOA Agent handler class for WebSphere. Default value is

Parameter	Description
	"com.ca.soa.agent.appserver.jaas.was.WasLoginHandler". Do not change this value.

You should not need to edit the preconfigured values unless the location of the Host Configuration File changes or you want to refer to a different Agent Configuration Object. If you choose to use local configuration, you can add other Agent configuration parameters to these preconfigured values.

Note: Parameters held in the Agent configuration file are static; if you change these settings while the WebSphere server is running, the SOA Agent will not pick up the change until WebSphere is restarted.

The JavaAgent.conf file also contains a list of SOA Agent plugin classes; you do not need to alter this information.

Generally, you only need to edit the JavaAgent.conf.file if you change the name of your Agent Configuration Object.

Sample JavaAgent.conf (Windows)

```
# SOA Agent Configuration File
#
# This file contains bootstrap information required by
# the SOA Agent
#
defaultagentname=SoaAgent
enableagent=yes
agentconfigobject=wsagent1_ac
servername=SOAWAS61
smhostfile=config\\SmHost.conf
appserverjaasloginhandler=com.ca.soa.agent.appserver.jaas.was.WasLoginHandler

# Configure plugins for the agent SoaAgent
transport_plugin_list=com.ca.soa.agent.httpplugin.pluginconfig.HttpPluginConfig,
com.ca.soa.agent.jaxrpcplugin.pluginconfig.JaxRpcPluginConfig
msg_body_plugin_list=com.ca.soa.agent.txmplugin.pluginconfig.TxmPluginConfig
credential_plugin_list=com.ca.soa.agent.httpplugin.pluginconfig.HttpPluginConfig,
com.ca.soa.agent.txmplugin.pluginconfig.TxmPluginConfig
variable_resolver_plugin_list=com.ca.soa.agent.txmplugin.pluginconfig.TxmPluginConfig

# <EOF>
```

More information:

Agent Configuration Object (see page 65)

Agent Configuration Object

An Agent Configuration Object is a SOA Security Manager policy object that holds Agent parameters for an Agent when using central agent configuration.

Note: Parameters held in an Agent Configuration Object are dynamic; if you change these settings while the WebSphere server is running, the SOA Agent will pick up the change.

SOA Agent for Application Servers Configuration Parameters

The following table contains a complete list of all Agent configuration parameters supported by SOA Agents for Application Servers.

Unless otherwise noted,you can define parameters in either the Agent Configuration Object or the Agent configuration file depending upon how you decide to configure the SOA Agent.

Parameter Name	Value	Description
AcceptTPCookie	YES or NO	(Optional) If set to yes, configures the SOA Agent to assert identities from third-party SiteMinder session cookies (that is, session cookies generated by custom Agents created using the SiteMinder and SOA Security Manager SDKs. **Note:** AcceptTPCookie must be set to Yes to assert identities from session cookies generated by CA SOA Security Gateway. Default is Yes.
AllowLocalConfig (Applies only in the Agent Configuration Object)	YES or NO	If set to yes, parameters set locally in the Agent configuration file take precedence over parameters in the Agent Configuration Object. Default is NO.
AuthCacheSize	Number	(Optional) Size of the authentication cache for the SOA Agent (in number of entries). For example: authcachesize="1000" Default is 0. To flush this cache, use the Policy Server User Interface.

Parameter Name	Value	Description
AzCacheSize		(Optional) Size of the authorization cache (in number of entries) for the SOA Agent. For example: authcachesize="1000" Default is 0. To flush this cache, use the Policy Server User Interface.
CacheTimeout	Number	(Optional) Number of seconds before cache times out. For example: cachetimeout="1000" Default is 600 (10 minutes).
ConfigObject (Applies only in Agent configuration file)	String	The name of the Agent Configuration Object associated with the SOA Agent. No default value.
CookieDomain	String	(Optional) Name of the cookie domain. For example: cookiedomain="ca.com" No default value. For more information, see the cookiedomainscope parameter.
CookieDomainScope	Number	(Optional) Further defines the cookie domain for assertion of SiteMinder session cookies by the SOA Agent. The scope determines the number of sections, separated by periods, that make up the domain name. A domain always begins with a period (.) character. For example: cookiedomainscope="2" Default is 0, which takes the domain name specified in the cookiedomain parameter.
DefaultAgentName (Applies only in the Agent Configuration Object)	String	The agent identity the Policy Server will use to associate policies with the SOA Agent. Default is "SoaAgent"; this value should not changed.
EnableWebAgent (Applies only in Agent configuration file)	YES or NO	Enables or disables the SOA Agent. When set to 'yes', the SOA Agent will protect resources using the Policies configured in the Policy Server for the configured agent

Parameter Name	Value	Description
		identity. Default is Yes.
LogOffUri	String	(Optional) The URI of a custom HTTP file that will perform a full log off (removing the session cookie from a user's browser). A fully qualified URI is not required. For example, LogOffUri could be set to: /Web pages/logoff.html No default value.
PsPollInterval	Number	(Optional) The frequency with which the SOA Agent polls the Policy Server to retrieve information about policy changes. Default is 30 seconds.
ResourceCacheSize	Number	(Optional) Size (in number of entries) of the cache for resource protection decisions. For example: resourcecachesize="1000" Default is 2000. To flush this cache, use the Policy Server User Interface.
SAMLSessionTicketLogoffi	YES or NO	(Optional) Determines whether the SOA Agent should attempt to log off session tickets in SAML assertions. Default is Yes.
ServerName (Applies only in Agent configuration file.)	String	A string to be used in the SOA Agent log to identify the target application server. Default value is "SOAWAS61" for SOA Agent for WebSphere and "SOAWLS92" for SOA Agent for WebLogic.
SessionGracePeriod	Number	(Optional) Grace period (in seconds) between the regeneration of session tokens. Default is 30
SmHostFile (Applies only in Agent configuration file)	String	Path to the local Host Configuration File (typically *SOA_HOME*\conf\SmHost.conf). No default value.

Parameter Name	Value	Description
XMLAgentSoapFaultDetails	YES or NO	(Optional) Determines whether or not the SOA Agent should insert the authentication/authorization rejection reason (if provided by the Policy Server) into the SOAP fault response sent to the web service consumer. Default is No.
XMLSDKAcceptSMSessionCookie	YES or NO	(Optional) Determines whether or not the SOA Agent accepts an CA SiteMinder session cookie to authenticate a client. Default is No. If set to Yes, the SOA Agent uses information in a session cookie sent as an HTTP header in the request as a means of authenticating the client. If set to No, session cookies are ignored and the SOA Agent requests credentials required by the configured authentication scheme.
XMLSDKMimeTypes	String	(Optional) A comma-delimited list of MIME types that the SOA Agent will accept for processing by SOA Security Manager. All POSTed requests having one of the listed MIME types are processed. Examples: ■ text/xml ■ application/octet-stream ■ text/xml,multipart/related If you do not add this parameter to the Agent Configuration Object, the SOA Agent defaults to accepting text/xml and application/soap+xml MIME types.

Configure the Username and Password Digest Token Age Restriction

By default, the WS-Security authentication scheme imposes a 60-minute restriction on the age of Username and Password Digest Tokens to protect against replay attacks.

To configure a different value for the token age restriction for a SOA Agent for Application Servers, add the WS_UT_CREATION_EXPIRATION_MINUTES parameter to the XmlToolkit.properties file for that agent.

To configure a SOA Agent to use a nondefault age restriction for Username and Password Digest token authentication

1. Navigate to *SOA_Agent_Install**config.*
2. Open XmlToolkit.properties in a text editor.
3. Add the following line:

   WS_UT_CREATION_EXPIRATION_MINUTES=*token_age_limit*

   token_age_limit

   Specifies the token age limit restriction in minutes.
4. Save and close the XmlToolkit.properties file.
5. Restart the SOA Agent.

Configure WebSphere to Work with the SOA Agent

You must configure WebSphere to use the SOA Agent to make authentication and authorization decisions for JAX-RPC web services.

Set the JAVA_AGENT_ROOT JVM System Property

Because the SOA Agent may not be installed in the same file system location on every system in clustered and SSO WebSphere environments, you must define a JVM system property, JAVA_AGENT_ROOT to define the installed location of the SOA Agent.

To set the JAVA_AGENT_ROOT JVM system property

1. Open the WebSphere Integrated Solutions Console.
2. Click the following, in the order shown:

   In the navigation tree: Servers, Application Server

   In the work area: *server_name*, Java and Process Management, Process Definition, Java virtual Machine, Additional Properties, Custom Properties.

3. Create a new variable in Custom Properties named JAVA_AGENT_ROOT and specify its value as the location where the SOA Agent is installed. For example, in Windows enter:
   JAVA_AGENT_ROOT=C:\SoaSecurityManager
4. Save the changes in the master repository.

Set the log.log-config-properties Environment Variable

You must define a JVM system property, log.log-config-properties, to define the location of the SOA Agent logging configuration file.

To set the log.log-config-properties JVM system property

1. Open the WebSphere Integrated Solutions Console.
2. Click the following, in the order shown:

   In the navigation tree: Servers, Application Server

   In the work area: *server_name*, Java and Process Management, Process Definition, Java Virtual Machine, Additional Properties, Custom Properties.
3. Create a new variable in Custom Properties named log.log-config-properties and specify its value as the location of the SOA Agent logging configuration file (relative to the installed location of the SOA Agent, *SOA_HOME)*.

   For example, in Windows enter:
   log.log-config-properties=config\log-config.properties
4. Save the changes in the master repository and restart the server.

More information:

SOA Agent Log Configuration File Summary (see page 76)

Configure General WebSphere Settings

Before you configure the SOA Agent, you must:

- Configure the active user registry for security
- Enable WebSphere Global Security
- Enable Security Attribute Propagation for WebSphere SSO, if required

Enable WebSphere Global Security

To enable security options for the WebSphere managed domain

1. If necessary, start the WebSphere Server and the WebSphere Integrated Solutions Console.
2. In the navigation tree click Security, Secure administration, applications, and infrastructure.
3. Set the Enable Administrative Security option.
4. Set the Use Java 2 security to restrict application access to local resources option.
5. Click Apply to apply your changes. To save changes, click System Administration and Save Changes to Master Repository.

   Note: Until you save changes to the master repository, the Integrated Solutions Console uses a local workspace to track your changes.

Configure LDAP as a WebSphere User Registry

In a typical deployment, WebSphere and the Policy Server are configured to use the same LDAP user registry.

Note: If you are not configuring WebSphere and the Policy Server to use the same LDAP user registry (typically because WebSphere is already configured with a custom user registry), ensure that the custom registry is properly configured (see the WebSphere documentation for information) and configure user mapping.

To configure a Policy Server LDAP user directory as a WebSphere user registry

1. If necessary, start the WebSphere Server and the WebSphere Integrated Solutions Console.
2. In the navigation tree, select Security, Secure administration, applications, and infrastructure.
3. In the User account repository section, select Standalone LDAP Registry from the Available Realm Definitions drop-down menu.
4. Click Apply to save your changes.
5. Click Configure.
6. Under Server user identity, enter the select the Server identity that is stored in repository option and type the identity and password of a user account that will be used to run the application server for security purposes in the corresponding fields.

7. Under General Properties , fill in the following fields and then click Apply.
   - Server user ID
   - Server user Password
   - Type
   - Host
   - Port
   - Base Distinguished Name (DN)
   - Bind Distinguished Name (DN)
   - Bind Password
   - Search timeout
8. Depending on the WebSphere configuration, check Reuse Connection and Ignore case for authorization.
9. Click Apply to apply your changes. To save changes to the master repository, click System Administration and Save Changes to Master Repository.

   Note: Until you save changes to the master repository, the Integrated Solutions Console uses a local workspace to track your changes.

Configure the SOA Agent Login Module in WebSphere

You configure the SOA Agent Login Module in the WebSphere Application Server using the WebSphere Integrated Solutions Console. General information about configuring Login Modules is available in the WebSphere documentation.

To configure the WebSphere Application Server to use the SOA Agent Login Module

1. If necessary, start the WebSphere Server and the WebSphere Integrated Solutions Console.
2. Click the following, in the order shown:

   In the navigation tree: Security, Secure Administration, Applications and Infrastructure.

   In the work area: Java Authentication and Authorization Service, System Logins.
3. Click New to create a new System Login profile. This profile will contain SOA Agent Login Module and two other standard WebSphere login modules create the WebSphere identity and credentials so that the identity is propagated to the rest of WebSphere and can be used for WebSphere single sign-on.
4. Under General Properties on the New page, enter "SOA Agent Login Module" in the Alias field and click Apply.

5. Under Additional Properties, click JAAS login modules.
6. Add the SOA Agent Login Module:
   a. On the JAAS Login Modules page, click New.
   b. Under General Properties on the New page, enter the SOA Agent Login Module class name:
      com.ca.soa.agent.appserver.jaas.XMLAgentLoginModule
   c. Ensure that REQUIRED is selected from the Authentication strategy drop-down list.
   d. Click Apply to save your changes.
7. Add the WebSphere LTPA Login Module:
   a. Back on the JAAS Login Modules page, click New.
   b. Under General Properties on the New page, enter the WebSphere LTPA Login Module class name:
      com.ibm.ws.security.server.lm.ltpaLoginModule
   c. Ensure that REQUIRED is selected from the Authentication strategy drop-down list.
   d. Click Apply to save your changes.
8. Add the WebSphere Default Inbound Login Module:
   a. Back on the JAAS Login Modules page, click New.
   b. Under General Properties on the New page, enter the WebSphere Default Inbound Login Module class name:
   c. com.ibm.ws.security.server.lm.wsMapDefaultInboundLoginModule
   d. Ensure that REQUIRED is selected from the Authentication strategy drop-down list.
   e. Click Apply to save your changes.
9. Back on the JAAS Login Modules page, click Set Order.
10. Under General Properties on the JAAS Login Module Order page, if necessary, move the Login Modules so that they appear in the following order:

    com.ca.soa.agent.appserver.jaas.XMLAgentLoginModule

    com.ibm.ws.security.server.lm.ltpaLoginModule

    com.ibm.ws.security.server.lm.wsMapDefaultInboundLoginModule

11. Click Apply to save your changes.To save changes permanently, click System Administration and Save Changes to the Master Repository.

    Note: Until you save changes to the master repository, the Integrated Solutions Console uses a local workspace to track your changes.

SOA Agent for Application Servers Logging

The SOA Agent for Application Servers logger is implemented using Apache's log4j. For more information, see http://logging.apache.org/log4j/docs/.

Log Files

Two log files provide important information about the SOA Agent:

- SOA Agent log file—Logs SOA Agent error and processing messages.
- SOA Agent XML message processing log file—Logs messages information relating specifically to the SOA Agent's processing of XML messages

SOA Agent Log

This SOA Agent writes information about its standard operations and performance to the SOA Agent log.

By default, SOA Agent logging is enabled and written to the XmlAgent.log file in:

- Windows—*SOA_HOME*\log
- UNIX—*SOA_HOME*/log

You can change SOA Agent logging parameters by editing the log-config.properties file located in:

  - Windows—*SOA_HOME*\config\
  - UNIX— *SOA_HOME*/config/

Note: These are the default values; the logging configuration file name and location can be changed by editing the log.log-config-properties JVM system property.

More information:

Set the log.log-config-properties Environment Variable (see page 70)

XML Processing Message Logging

In addition to its standard logging functionality, SOA Agents for Application Servers also logs information relating specifically to its processing of XML messages. Like the SOA Agent log, the XML message processing log is also implemented using Apache's *log4j* standard.

Note: SOA Agent XML message processing logging does not start until an XML message that needs to be processed is received.

By default, SOA Agent XML message processing logging is enabled and written to the soasm_agent.log file in:

- Windows—*SOA_HOME*\bin\
- UNIX—*SOA_HOME*/bin/

You can change SOA Agent XML message processing logging parameters by editing the log.config file, which can be found in:

- Windows—*SOA_HOME*\config\
- UNIX— *SOA_HOME*/config/

Change the SOA Agent Log File Name

To change pathname of the SOA Agent log file, edit the log.logfile-pattern parameter. Possible values are valid pathnames. If you specify a relative value, the path is set relative to the JAVA_AGENT_ROOT JVM system property.

Default value: "log\XmlAgent.log"

For example:

```
log.logfile-pattern=log\XmlAgent.log
```

Append Messages to an Existing SOA Agent Log File

To add logging information to an existing SOA Agent log file instead of rewriting the entire file each time logging is invoked, enable the log.logfile-append-on-reset parameter.

For example:

```
log.logfile-append-on-reset=YES
```

Set the SOA Agent File Log Level

To change the SOA Agent log level, edit the log.logging-level parameter. Possible values are:

- DEBUG - Logs all; most verbose
- CONFIG - Configuration information
- INFO - Information
- WARNING - Warnings
- SEVERE - Errors only; least verbose

Default value: WARNING

For example:

log.logging-level=INFO

Roll Over the SOA Agent Log File

To change file size limit at which the SOA Agent log should rollover, change the log.logfile-limit parameter. Rolling over a log file starts a new log file, preventing a single log file from becoming unmanageable. Possible values are numbers, representing kilobytes.

The default value is 1000.

For example:

log.logfile-limit=*512*

SOA Agent Log Configuration File Summary

The SOA Agent logging configuration file defines default SOA Agent logging settings.

Available configuration parameters are:

Name	Description
log.logfile-append-on-reset	Add logging information to an existing log file instead of creating a new file each time logging is invoked. Default value: no

Name	Description
log.logfile-pattern	Specifies the pathname (relative to *SOA_HOME*) of the SOA Agent log file. Default value: log/XmlAgent.log
log.logging-level	Defines the logging level. The levels are: ■ DEBUG - all logging, most verbose ■ CONFIG - configuration information ■ INFO - information ■ WARNING -warnings ■ SEVERE - errors Default value: WARNING
log.logfile-limit	Specifies the size limit, in kilobytes Rollover a log file after it reaches the specified size. Default value: 1,000KB

Note: Once the SOA Agent connects to the Policy Server, corresponding logging settings found in the Agent Configuration Object override the values in log-config.properties.

Restart WebSphere

After completing WebSphere-side configuration of the SOA Agent, you must restart WebSphere.

To restart IBM WebSphere

1. Log out of the Integrated Solutions Console.
2. From a command line or shell in the *WS_HOME*/bin directory, stop and then restart the WebSphere Server.

   To stop the server, you will require the server user ID and Server user password you entered when configuring LDAP as a WebSphere user registry. The command is:

   stopServer server1 -username *serveruserID* -password *serveruserpassword*

   To start the server, you do not need a password:

   startServer server1

3. To make sure everything is working as expected, view the SOA Agent and WebSphere (SystemOut.log, SystemErr.log) log files.

   WebSphere's SystemOut.log and SystemErr.log file resides in:

   WS_HOME/profiles/*profile_name*/logs/*server_name*

   The logs indicate should indicate that everything is working correctly. If the logs indicate problems, you should troubleshoot your configuration.

More information:

Configure LDAP as a WebSphere User Registry (see page 71)

Edit Deployment Descriptors of JAX-RPC Applications

To protect a JAX-RPC web service you must edit its deployment descriptor to add the SOA Agent JAX-RPC Handler.

To edit a JAX-RPC web service deployment descriptor

1. Unpack the enterprise archive (EAR) containing one or more web services.
2. Examine the EAR to determine which of the modules within it contains a JAX-RPC web service.(A module that contains a JAX-RPC web service if it has a webservices.xml file in the META-INF folder for EJB endpoints, or the WEB-INF folder for servlet endpoints.)
3. For each module in the EAR identified as a JAX-RPC web service:
   a. Unpack the archive containing the module. (The archive will be a JAR file for EJB endpoints and a WAR file for servlet endpoints.)
   b. Find the webservices.xml file.
   c. For each port-component element found in the webservices.xml file, add a handler element:

```
<handler>
  <handler-name>SOA Agent Handler<handler-name>
  <handler-class>
    com.ca.soa.agent appserver.jaxrpc.SOAAgentJaxrpcHandler
  </handler-class>
</handler>
```

   Note: The SOA Agent JAX-RPC handler must always be invoked first; If other handler elements are already present or subsequently added to the webservices.xml file, the SOA Agent JAX-RPC Handler element must placed before them.
4. Repackage the module into the appropriate archive type (JAR or WAR).

5. When all modules have been configured, repackage the EAR.
6. Install or update the enterprise application.

Configure Policies for the SOA Agent

You create authentication and authorization policies to protect web service resources hosted on WebSphere from their associated WSDL files using the SOA Security Manager Configuration User Interface. For more information, see the *SOA Security Manager Policy Configuration Guide*.

Policy Considerations for Coexistence With SiteMinder Agent for IBM WebSphere

If a SiteMinder Agent for IBM WebSphere coexists with a SOA Agent for IBM WebSphere on the same IBM WebSphere Application Server platform, you must consider the following when implementing authorization policies:

- Web Resources protected by the SiteMinder TAI must be configured in a separate realm from web services (the SiteMinder TAI is not called when invoking web service endpoints even if they are implemented as servlets).
- EJB resources that are protected by the SiteMinder Agent Login Module should be in a separate realm from EJBs that act as endpoints for web services protected by the SOA Agent.
- All resources protected by the SiteMinder Agent JACC module should be in separate realms than web service resources (the SOA Agent handles its own authorization functions).

Chapter 3: SOA Agent for BEA WebLogic

This section contains the following topics:

SOA Agent for BEA WebLogic Introduction

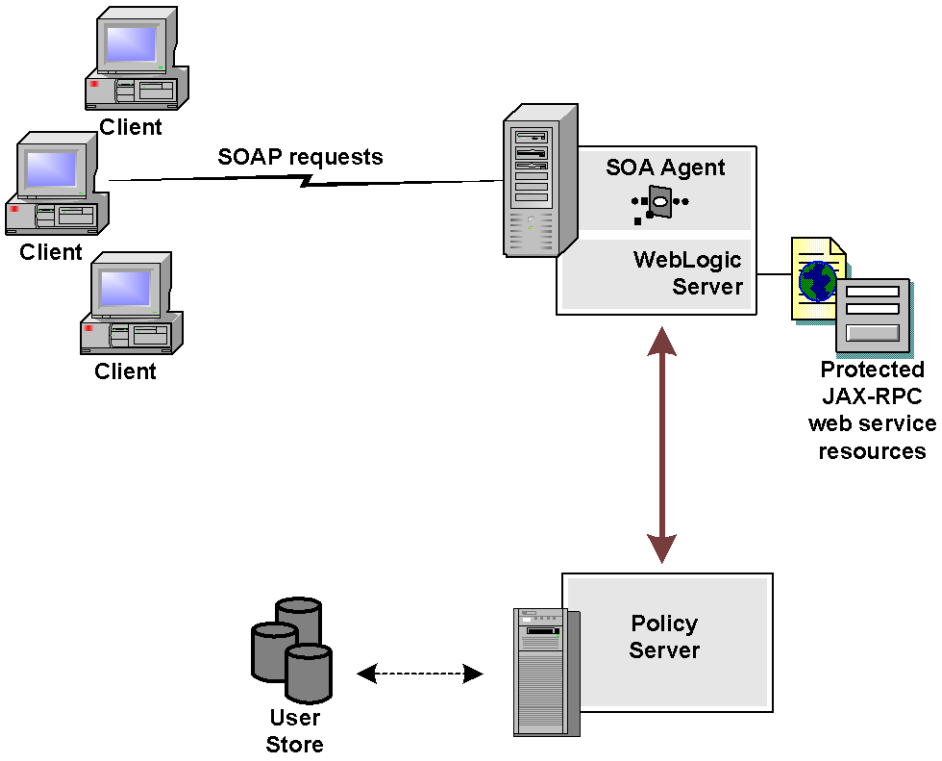
The SOA Agent for BEA WebLogic provides a SOA Security Manager-based access control solution for BEA WebLogic Server Server. The SOA Agent integrates the WebLogic Server into the SOA Security Manager environment, enabling you to implement policy-based access control to protect WebLogic-hosted JAX-RPC web services.

SOA Agent for BEA WebLogic Overview

The SOA Agent for BEA WebLogic resides in a WebLogic application server, enabling you to protect WebLogic-hosted JAX-RPC web service resources.

The SOA Agent for BEA WebLogic intercepts all SOAP messages sent over HTTP(S) or JMS transports to JAX-RPC web services deployed on the WebLogic Server. The SOA Agent then communicates with the Policy Server to authenticate and authorize the message sender and, upon successful authentication and authorization, passes the SOAP message on to the addressed web service.

A high-level overview of the SOA Agent for BEA WebLogic Server architecture is shown in the following figure.



The SOA Agent for BEA WebLogic provides the following features:

- Fine-grained access control of JAX-RPC web service resources
- Support for bi-directional SOA Security Manager/SiteMinder and WebLogic single sign-on (SSO)

The SOA Agent additionally supports:

- Multi-byte character user names
- Centralized and dynamic agent configurations
- Caching of resource protection decisions and authentication and authorization decisions
- Logging
- Authorization auditing

Required Background Information

This section is not intended for users who are new to Java, J2EE standards, or application server technology. It assumes that you have the following technical knowledge:

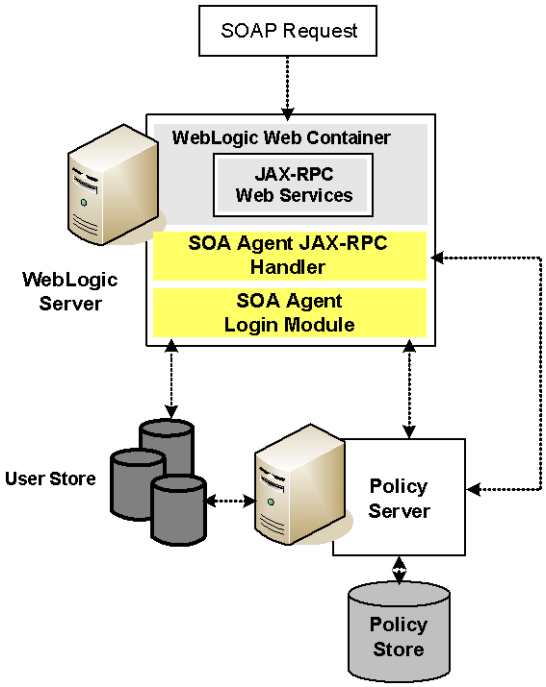
- An understanding of Java, J2EE standards, J2EE application servers, and multi-tier architecture
- An understanding of JAX-RPC web service implementations and JAX-RPC handlers
- Familiarity with the WebLogic Security Framework for WebLogic Server
- Familiarity with Java Authentication and Authorization Server (JAAS) and WebLogic security-related topics
- Experience with managing the WebLogic Server, including tasks such as accessing the administrative console
- Familiarity with SOA Security Manager concepts, terms, and Policy Server configuration tasks

Additionally, to effectively plan your security infrastructure, you must be familiar with the web services that you plan to protect with SOA Security Manager.

SOA Agent for BEA WebLogic Components

The SOA Agent for BEA WebLogic consists of two modules that plug into WebLogic's security infrastructure.

- SOA Agent JAX-RPC Handler
- SOA Agent Login Module



SOA Agent JAX-RPC Handler

The SOA Agent JAX-RPC Handler is a custom JAX-RPC Handler that, when added to the deployment descriptor of a JAX-RPC web service, intercepts SOAP message requests for JAX-RPC web services and diverts them to the SOA Agent Login Module for authentication and authorization decisions.

SOA Agent Login Module

The SOA Agent Login Module is a JAAS Login Module that performs authentication and authorization for JAX-RPC web services protected by the SOA Agent for BEA WebLogic.

The SOA Agent Login Module authenticates credentials obtained from the following request types against associated user directories configured in SOA Security Manager:

- SOAP requests intercepted by the SOA Agent JAX-RPC Handler .
- Requests for web service resources from users with pre-established SOA Security Manager and SiteMinder sessions (validating the session and obtaining user names from associated SiteMinder session ticket cookies)

If SOA Security Manager authentication is successful, the SOA Agent Login Module populates a JAAS Subject with a SOA Security Manager Principal that contains the username and associated SOA Security Manager session data. The SOA Agent Login Module then determines whether an authenticated user is allowed to access a protected WebLogic resource, based on associated SOA Security Manager authorization policies.

Installation Location References

In this guide:

- *SOA_HOME* refers to the installed location of the SOA Agent for BEA WebLogic.
- *WLS_HOME* refers to the installed location of the WebLogic Server.

Configure the SOA Agent

To configure the SOA Agent, you must specify the following:

- Host Configuration Object (one for each host server)
- Agent Configuration Object (one for each SOA Agent)
- Agent identity (one for each SOA Agent)

Note: For detailed information about how to configure Agent-related objects, see *the SOA Security Manager Policy Configuration Guide* and the *SOA Security Manager Implementation Guide*.

To configure the SOA Agent

1. On the Policy Server:
   a. Duplicate or create a Host Configuration Object, which holds initialization parameters for a Trusted Host.

      The Trusted Host is a server that hosts one or more Agents and handles their connection to the Policy Server.

   b. As necessary, add or edit Trusted Host parameters in the Host Configuration Object that you just created.

c. Duplicate or create an Agent Configuration Object, which holds Agent configuration parameters and can be used to centrally configure a group of Agents.

d. Add or edit required Agent parameters in the Agent Configuration Object.

   The configuration object must include the DefaultAgentName or AgentName parameter to specify the Agent identity.

e. Create an Agent identity for the SOA Agent. You must select *Web Agent* as the Agent type for a SOA Agent.

2. On the system where the SOA Agent is installed:

   a. Run the Agent Configuration Wizard, which registers the Trusted Host.

   b. Enable the SOA Agent by setting the EnableWebAgent parameter in the Agent configuration file to Yes.

SOA Agent for WebLogic Agent Configuration File

By default, the SOA Agent for WebLogic installation creates a single agent configuration file, JavaAgent.conf.

The agent configuration file is located in the *SOA_HOME*/config directory, where *SOA_HOME* is the location where you installed the SOA Agent. For example:

- For Windows

  C:\SoaSecurityManager\config

- For UNIX

  /opt/SoaSecurityManager/config

Each Agent configuration file is created with the following required default configuration parameters/values:

Parameter	Description
DefaultAgentName	The agent identity the Policy Server uses to associate policies with the SOA Agent. The default value is "SoaAgent". Do not change this value.
EnableAgent	Specifies whether the SOA Agent is enabled. Possible values are Yes and No. Default value is Yes.
AgentConfigObject	The Agent Configuration Object specified during installation.

Parameter	Description
SmHostFile	Path to the local Host Configuration File. Path can be specified in absolute terms or relative to *SOA_HOME.* **Note:** On Windows, you must specify paths using double backslashes ("\\") rather than single backslash ("\") to separate directories. On UNIX, use standard single slash ("/") separators. Example values: ■ (Windows) C:\\Program Files\\CA\\SOASecurityManager\\soaagent\\wlsagent\\config\\SmHost.conf ■ (Windows) config\\SmHost.conf ■ (UNIX) export/soaagent/wlsagent/config/SmHost.conf ■ (UNIX) /config/SmHost.conf
ServerName	A string that will be used in the SOA Agent log to identify the WebLogic Server. Default value is "SOAWLS92".
appserverjaasloginhandler	Specifies the Application Server-specific SOA Agent handler class for WebLogic Default value is "com.ca.soa.agent.appserver.jaas.wls.WlsLoginHandler". Do not change this value.

You should not need to edit the preconfigured values unless the location of the Host Configuration File changes or you want to refer to a different Agent Configuration Object. If you choose to use local configuration, you can add other Agent configuration parameters to these preconfigured values.

Note: Parameters held in the Agent configuration file are static; if you change these settings while the WebLogic server is running, the SOA Agent will not pick up the change until WebLogic is restarted.

The JavaAgent.conf file also contains a list of SOA Agent plugin classes; you do not need to alter this information.

Generally, you only need to edit the JavaAgent.conf.file if you change the name of your Agent Configuration Object.

Sample JavaAgent.conf (Windows)

```
# Java Agent Configuration File
#
# This file contains bootstrap information required by
# the SiteMinder Java Agent
#
#
# Configuration for agent testagent
#
defaultagentname=soaagent
enablewebagent=yes
agentconfigobject=soaagentconfig
servername=wsdell110.systemtest.local
smhostfile=C:\\SOASecurityManager\\wlsagent\\config\\SmHost.conf
appserverjaasloginhandler=com.ca.soa.agent.appserver.jaas.wls.WLSLoginHandler
appserverjmshandler=com.ca.soa.agent.appserver.jaxrpc.jms.wls.WLSJMSMessageHandler
# Configure plugins for the agent testagent
transport_plugin_list=com.ca.soa.agent.httpplugin.pluginconfig.HttpPluginConfig,
com.ca.soa.agent.jaxrpcplugin.pluginconfig.JaxRpcPluginConfig,
com.ca.soa.agent.jmsplugin.pluginconfig.JMSPluginConfig
msg_body_plugin_list=com.ca.soa.agent.txmplugin.pluginconfig.TxmPluginConfig
credential_plugin_list=com.ca.soa.agent.httpplugin.pluginconfig.HttpPluginConfig,
com.ca.soa.agent.txmplugin.pluginconfig.TxmPluginConfig
variable_resolver_plugin_list=com.ca.soa.agent.txmplugin.pluginconfig.TxmPluginConfig
# <EOF>
```

Agent Configuration Object

An Agent Configuration Object is a SOA Security Manager policy object that holds Agent parameters for an Agent when using central agent configuration.

Note: Parameters held in an Agent Configuration Object are dynamic; if you change these settings while the WebLogic server is running, the SOA Agent will pick up the change.

SOA Agent for Application Servers Configuration Parameters

The following table contains a complete list of all Agent configuration parameters supported by SOA Agents for Application Servers.

Unless otherwise noted,you can define parameters in either the Agent Configuration Object or the Agent configuration file depending upon how you decide to configure the SOA Agent.

Parameter Name	Value	Description
AcceptTPCookie	YES or NO	(Optional) If set to yes, configures the SOA Agent to assert identities from third-party SiteMinder session cookies (that is, session cookies generated by custom Agents created using the SiteMinder and SOA Security Manager SDKs. **Note:** AcceptTPCookie must be set to Yes to assert identities from session cookies generated by CA SOA Security Gateway. Default is Yes.
AllowLocalConfig (Applies only in the Agent Configuration Object)	YES or NO	If set to yes, parameters set locally in the Agent configuration file take precedence over parameters in the Agent Configuration Object. Default is NO.
AuthCacheSize	Number	(Optional) Size of the authentication cache for the SOA Agent (in number of entries). For example: authcachesize="1000" Default is 0. To flush this cache, use the Policy Server User Interface.
AzCacheSize		(Optional) Size of the authorization cache (in number of entries) for the SOA Agent. For example: authcachesize="1000" Default is 0. To flush this cache, use the Policy Server User Interface.
CacheTimeout	Number	(Optional) Number of seconds before cache times out. For example: cachetimeout="1000" Default is 600 (10 minutes).

Parameter Name	Value	Description
ConfigObject (Applies only in Agent configuration file)	String	The name of the Agent Configuration Object associated with the SOA Agent. No default value.
CookieDomain	String	(Optional) Name of the cookie domain. For example: cookiedomain="ca.com" No default value. For more information, see the cookiedomainscope parameter.
CookieDomainScope	Number	(Optional) Further defines the cookie domain for assertion of SiteMinder session cookies by the SOA Agent. The scope determines the number of sections, separated by periods, that make up the domain name. A domain always begins with a period (.) character. For example: cookiedomainscope="2" Default is 0, which takes the domain name specified in the cookiedomain parameter.
DefaultAgentName (Applies only in the Agent Configuration Object)	String	The agent identity the Policy Server will use to associate policies with the SOA Agent. Default is "SoaAgent"; this value should not changed.
EnableWebAgent (Applies only in Agent configuration file)	YES or NO	Enables or disables the SOA Agent. When set to 'yes', the SOA Agent will protect resources using the Policies configured in the Policy Server for the configured agent identity. Default is Yes.
LogOffUri	String	(Optional) The URI of a custom HTTP file that will perform a full log off (removing the session cookie from a user's browser). A fully qualified URI is not required. For example, LogOffUri could be set to: /Web pages/logoff.html No default value.

Parameter Name	Value	Description
PsPollInterval	Number	(Optional) The frequency with which the SOA Agent polls the Policy Server to retrieve information about policy changes. Default is 30 seconds.
ResourceCacheSize	Number	(Optional) Size (in number of entries) of the cache for resource protection decisions. For example: resourcecachesize="1000" Default is 2000. To flush this cache, use the Policy Server User Interface.
SAMLSessionTicketLogoffi	YES or NO	(Optional) Determines whether the SOA Agent should attempt to log off session tickets in SAML assertions. Default is Yes.
ServerName (Applies only in Agent configuration file.)	String	A string to be used in the SOA Agent log to identify the target application server. Default value is "SOAWAS61" for SOA Agent for WebSphere and "SOAWLS92" for SOA Agent for WebLogic.
SessionGracePeriod	Number	(Optional) Grace period (in seconds) between the regeneration of session tokens. Default is 30
SmHostFile (Applies only in Agent configuration file)	String	Path to the local Host Configuration File (typically *SOA_HOME*\conf\SmHost.conf). No default value.
XMLAgentSoapFaultDetails	YES or NO	(Optional) Determines whether or not the SOA Agent should insert the authentication/authorization rejection reason (if provided by the Policy Server) into the SOAP fault response sent to the web service consumer. Default is No.

Parameter Name	Value	Description
XMLSDKAcceptSMSessionCookie	YES or NO	(Optional) Determines whether or not the SOA Agent accepts an CA SiteMinder session cookie to authenticate a client. Default is No. If set to Yes, the SOA Agent uses information in a session cookie sent as an HTTP header in the request as a means of authenticating the client. If set to No, session cookies are ignored and the SOA Agent requests credentials required by the configured authentication scheme.
XMLSDKMimeTypes	String	(Optional) A comma-delimited list of MIME types that the SOA Agent will accept for processing by SOA Security Manager. All POSTed requests having one of the listed MIME types are processed. Examples: ■ text/xml ■ application/octet-stream ■ text/xml,multipart/related If you do not add this parameter to the Agent Configuration Object, the SOA Agent defaults to accepting text/xml and application/soap+xml MIME types.

Configure the Username and Password Digest Token Age Restriction

By default, the WS-Security authentication scheme imposes a 60-minute restriction on the age of Username and Password Digest Tokens to protect against replay attacks.

To configure a different value for the token age restriction for a SOA Agent for Application Servers, add the WS_UT_CREATION_EXPIRATION_MINUTES parameter to the XmlToolkit.properties file for that agent.

To configure a SOA Agent to use a nondefault age restriction for Username and Password Digest token authentication

1. Navigate to *SOA_Agent_Install\config.*
2. Open XmlToolkit.properties in a text editor.

3. Add the following line:

   WS_UT_CREATION_EXPIRATION_MINUTES=*token_age_limit*

   token_age_limit

   Specifies the token age limit restriction in minutes.

4. Save and close the XmlToolkit.properties file.
5. Restart the SOA Agent.

Set the WebLogic Environment for the SOA Agent

Before the SOA Agent can operate with the WebLogic Application Server, you must configure SOA Agent-related environment settings.

WebLogic Environment Setting Locations

You configure SOA Agent-related environment settings in one of the following locations depending on your environment:

- The WebLogic start script for both managed and standalone servers (startWebLogic.cmd on Windows; startWebLogic.sh on UNIX)

  Note: The startWebLogic.cmd (Windows) or startWebLogic.sh (Unix) script that contains the environment configuration is placed in the bin folder of a created domain. For example:

  C:\bea\user_projects\domains\MyDomain\bin\startWebLogic.cmd (Windows)

- If using the Node Manager to control Managed Servers, in the Server Start configuration page in the WebLogic Adminstration Console.

  For details regarding the Server Start configuration page, see BEA's Online Documentation http://e-docs.bea.com/wls/docs92/ConsoleHelp/taskhelp/startstop/ConfigureStartupArgumentsForManagedServers.html.

More information:

Set the WebLogic Environment on Windows (see page 94)
Set the WebLogic Environment on UNIX (see page 95)

Set the WebLogic Environment on Windows

Before the SOA Agent can operate with the WebLogic Application Server on Windows, you must configure SOA Agent-related environment settings in the location appropriate for your environment.

To configure SOA Agent-related environment settings

1. Define a SMSOA_CLASSPATH variable as follows:

   ```
   set SMSOA_CLASSPATH=SOA_HOME\config;
   %SOA_HOME%\lib\smagentapi.jar;
   %SOA_HOME%\lib\sm_jsafe.jar;
   %SOA_HOME%\lib\sm_jsafeJCE.jar;
   %SOA_HOME%\lib\soaagent-proxy.jar;
   %SOA_HOME%\lib\xalan.jar
   ```

   SOA_HOME

   The location where the SOA Agent for WebLogic is installed.

2. Add %SMSOA_CLASSPATH% to the beginning of the CLASSPATH variable definition. The modified CLASSPATH variable should resemble the following:

   ```
   set CLASSPATH=%SMSOA_CLASSPATH%;%CLASSPATH%
   ```

3. Define the SM_JAVA_OPTIONS variable as follows:

   ```
   set SM_JAVA_OPTIONS=-DJAVA_AGENT_ROOT=%SOA_HOME%
   -Dlog.log-config-properties=%SOA_HOME%\config\log-config.properties
   -Djava.security.auth.login.config==%SOA_HOME%\config\soa_jaas.config
   -Djavax.xml.soap.SOAPFactory=weblogic.xml.saaj.SOAPFactoryImpl
   -Djavax.xml.soap.MessageFactory=weblogic.xml.saaj.MessageFactoryImpl
   ```

4. Add %SM_JAVA_OPTIONS% to the execution entry. The modified execution entry should resemble the following:

   ```
   %JAVA_HOME%\bin\java %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS%
   %SM_JAVA_OPTIONS% -Dweblogic.Name=%SERVER_NAME%
   -Djava.security.policy=%WL_HOME%\server\lib\weblogic.policy %PROXY_SETTINGS%
   %SERVER_CLASS%
   ```

5. Save your changes.

6. Restart the WebLogic Application Server for changes to take effect.

More information:

WebLogic Environment Setting Locations (see page 93)

Set the WebLogic Environment on UNIX

Before the SOA Agent can operate with the WebLogic Application Server on UNIX, you must configure SOA Agent-related environment settings in the location appropriate for your environment.

To configure SOA Agent-related environment settings

1. Define the SMSOA_CLASSPATH as follows:

   ```
   SMSOA_CLASSPATH=${SOA_HOME}/config:
   ${SOA_HOME}/lib/smagentapi.jar:
   ${SOA_HOME}/lib/sm_jsafe.jar:
   ${SOA_HOME}/lib/sm_jsafeJCE.jar:
   ${SOA_HOME}/lib/soaagent-proxy.jar:
   ${SOA_HOME}/lib/xalan.jar

   export SMSOA_CLASSPATH
   ```

   SOA_HOME

   The location where the SOA Agent for WebLogic is installed.

2. Add ${SMSOA_CLASSPATH} to the beginning of the CLASSPATH definition. The modified CLASSPATH variable should resemble the following:

   ```
   CLASSPATH=${SMSOA_CLASSPATH}${CLASSPATHSEP}${CLASSPATH}
   export CLASSPATH
   ```

3. Define the SM_JAVA_OPTIONS variable as follows:

   ```
   SM_JAVA_OPTIONS="-DJAVA_AGENT_ROOT=${SOA_HOME}
   -Dlog.log-config-properties=${SOA_HOME}/config/log-config.properties
   -Djava.security.auth.login.config==${SOA_HOME}/config/soa_jaas.config
   -Djavax.xml.soap.SOAPFactory=weblogic.xml.saaj.SOAPFactoryImpl
   -Djavax.xml.soap.MessageFactory=weblogic.xml.saaj.MessageFactoryImpl"
   ```

4. Add ${SM_JAVA_OPTIONS} to the execution entry. The modified execution entry should resemble the following:

   ```
   "${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} ${JAVA_OPTIONS} ${SM_JAVA_OPTIONS}
   -Dweblogic.Name=${SERVER_NAME} -Djava.security.policy=${WL_HOME}/server/lib/weblogic.policy
   ${PROXY_SETTINGS} ${SERVER_CLASS}"
   ```

5. Save your changes.
6. Restart the WebLogic Application Server for changes to take effect.

More information:

WebLogic Environment Setting Locations (see page 93)

Configure the JVM to Use the JSafeJCE Security Provider

The SOA Agent for WebLogic XML encryption function requires that the JVM is configured to use the JSafeJCE security provider.

To configure the SOA Agent to use the JSafeJCE security provider, add a JSafeJCE security provider entry (com.rsa.jsafe.provider.JsafeJCE) to *JVM_HOME*\lib\security\java.security file.

JVM_HOME

Is the installed location of the JVM used by the WebLogic Server.

In the following example, the JSafeJCE security provider entry has been added as the second security provider:

```
security.provider.1=sun.security.provider.Sun
security.provider.2=com.rsa.jsafe.provider.JsafeJCE
security.provider.3=sun.security.rsa.SunRsaSign
security.provider.4=com.sun.net.ssl.internal.ssl.Provider
security.provider.5=com.sun.crypto.provider.SunJCE
security.provider.6=sun.security.jgss.SunProvider
security.provider.7=com.sun.security.sasl.Provider
```

SOA Agent for Application Servers Logging

The SOA Agent for Application Servers logger is implemented using Apache's log4j. For more information, see http://logging.apache.org/log4j/docs/.

Log Files

Two log files provide important information about the SOA Agent:

- SOA Agent log file—Logs SOA Agent error and processing messages.
- SOA Agent XML message processing log file—Logs messages information relating specifically to the SOA Agent's processing of XML messages

SOA Agent Log

The SOA Agent for BEA WebLogic writes information about its standard operations and performance to the SOA Agent log.

By default, SOA Agent logging is enabled and written to the XmlAgent.log file in:

- Windows—*WLS_HOME*\user_projects\domain\defaultdomain\soa-log
- UNIX—*WLS_HOME*/user_projects/domain/defaultdomain/soa-log

You can change SOA Agent logging parameters by editing the log-config.properties file located in:

- Windows—*SOA_HOME*\config\
- UNIX— *SOA_HOME*/config/

Note: These are the default values; the logging configuration file name and location can be changed by editing the log.log-config-properties JVM system property.

More information:

Set the WebLogic Environment on Windows (see page 94)
Set the WebLogic Environment on UNIX (see page 95)

XML Processing Message Logging

In addition to its standard logging functionality, SOA Agents for Application Servers also logs information relating specifically to its processing of XML messages. Like the SOA Agent log, the XML message processing log is also implemented using Apache's *log4j* standard.

Note: SOA Agent XML message processing logging does not start until an XML message that needs to be processed is received.

By default, SOA Agent XML message processing logging is enabled and written to the soasm_agent.log file in:

- Windows—*SOA_HOME*\bin\
- UNIX—*SOA_HOME*/bin/

You can change SOA Agent XML message processing logging parameters by editing the log.config file, which can be found in:

- Windows—*SOA_HOME*\config\
- UNIX— *SOA_HOME*/config/

Change the SOA Agent Log File Name

To change pathname of the SOA Agent log file, edit the log.logfile-pattern parameter. Possible values are valid pathnames. If you specify a relative value, the path is set relative to the JAVA_AGENT_ROOT JVM system property.

Default value: "log\XmlAgent.log"

For example:

```
log.logfile-pattern=log\XmlAgent.log
```

Append Messages to an Existing SOA Agent Log File

To add logging information to an existing SOA Agent log file instead of rewriting the entire file each time logging is invoked, enable the log.logfile-append-on-reset parameter.

For example:

log.logfile-append-on-reset=YES

Set the SOA Agent File Log Level

To change the SOA Agent log level, edit the log.logging-level parameter. Possible values are:

- DEBUG - Logs all; most verbose
- CONFIG - Configuration information
- INFO - Information
- WARNING - Warnings
- SEVERE - Errors only; least verbose

Default value: WARNING

For example:

log.logging-level=INFO

Roll Over the SOA Agent Log File

To change file size limit at which the SOA Agent log should rollover, change the log.logfile-limit parameter. Rolling over a log file starts a new log file, preventing a single log file from becoming unmanageable. Possible values are numbers, representing kilobytes.

The default value is 1000.

For example:

log.logfile-limit=*512*

SOA Agent Log Configuration File Summary

The SOA Agent logging configuration file defines default SOA Agent logging settings.

Available configuration parameters are:

Name	Description
log.logfile-append-on-reset	Add logging information to an existing log file instead of creating a new file each time logging is invoked. Default value: no
log.logfile-pattern	Specifies the pathname (relative to *SOA_HOME*) of the SOA Agent log file. Default value: log/XmlAgent.log
log.logging-level	Defines the logging level. The levels are: ■ DEBUG - all logging, most verbose ■ CONFIG - configuration information ■ INFO - information ■ WARNING -warnings ■ SEVERE - errors Default value: WARNING
log.logfile-limit	Specifies the size limit, in kilobytes Rollover a log file after it reaches the specified size. Default value: 1,000KB

Note: Once the SOA Agent connects to the Policy Server, corresponding logging settings found in the Agent Configuration Object override the values in log-config.properties.

Prevent WebLogic 10 from Loading Incompatible Version of XML Security

By default, Weblogic Server 10 loads an older version of XML security (1.3.0) that is incompatible with the version used by the SOA Agent (1.4.1).

To prevent WebLogic 10 from loading the 1.3.0 XML security JAR, rename *WLS_HOME*\modules\com.bea.core.apache.xml.security_1.3.0.jar to some other name. For example, com.bea.core.apache.xml.security_1.3.0_backup.jar.

Restart WebLogic

After completing WebLogic-side configuration of the SOA Agent, you must restart the WebLogic server.

Configure Web Services to Invoke the SOA Agent JAX-RPC Handler

To protect a JAX-RPC web service using the SOA Agent, you must configure it to invoke the SOA Agent JAX-RPC Handler. To do this, you must add the SOA Agent JAX-RPC Handler class (com.ca.soa.agent.appserver.jaxrpc.XMLAgentJaxrpcHandlerProxy) to the web service deployment descriptor in the webservices.xml file.

You can do this manually by editing the webservices.xml file for each JAX-RPC web service module. However, if your web services are implemented as JWS files and you have set up an Ant-based development environment, it is more efficient to update your web services to use handler chains.

Manually Edit Deployment Descriptors

To configure JAX-RPC web services not implemented as JWS files to invoke the SOA Agent, you must manually edit their deployment descriptors to add the SOA Agent JAX-RPC Handler.

To manually edit JAX-RPC web service deployment descriptors

1. Unpack the enterprise archive (EAR) containing one or more web services.
2. Examine the EAR to determine which of the modules within it contains a JAX-RPC web service.(A module that contains a JAX-RPC web service if it has a webservices.xml file in the META-INF folder for EJB endpoints, or the WEB-INF folder for servlet endpoints.)
3. For each module in the EAR identified as a JAX-RPC web service:
    a. Unpack the archive containing the module. (The archive will be a JAR file for EJB endpoints and a WAR file for servlet endpoints.)
    b. Find the webservices.xml file.

c. For each port-component element found in the webservices.xml file, add a handler element:

```
<handler>
  <handler-name>SOA Agent Handler<handler-name>
  <handler-class>
    com.ca.soa.agent.appserver.jaxrpc.XMLAgentJaxrpcHandlerProxy
  </handler-class>
</handler>
```

Note: The SOA Agent JAX-RPC handler must always be invoked first; If other handler elements are already present or subsequently added to the webservices.xml file, the SOA Agent JAX-RPC Handler element must placed before them.

4. Repackage the module into the appropriate archive type (JAR or WAR).
5. When all modules have been configured, repackage the EAR.
6. Install or update the enterprise application.

Use Handler Chains

The most efficient way to configure services implemented as JWS files to invoke the SOA Agent is to define the SOA Agent JAX-RPC Handler class in a handler chain configuration file which can then be referenced from the JWS files of all web services in your enterprise that you need to protect. Another benefit.

Note: The following procedures assume that you have set up an Ant-based development environment and have a working build.xml file that includes a target for running the jwsc Ant task.

Use a Handler Chain to Invoke the SOA Agent for HTTP Requests

To configure services implemented as JWS files to invoke the SOA Agent for requests received over HTTP transport, define the SOA Agent JAX-RPC Handler class in a handler chain configuration file.

To use a handler chain to invoke the SOA Agent for a JWS web service for HTTP requests

1. Create a handler chain configuration file that defines a JAX-RPC handler chain. The chain can include as many handler classes as you require but must define the SOA Agent JAX-RPC Handler class first.

   Example HandlerConfig.xml:

```
<jwshc:handler-config xmlns:jwshc="http://www.bea.com/xml/ns/jws"
 xmlns:soap1="http://HandlerInfo.org/Server1"
 xmlns:soap2="http://HandlerInfo.org/Server2"
 xmlns="http://java.sun.com/xml/ns/j2ee" >
  <jwshc:handler-chain>
   <jwshc:handler-chain-name>HandlerChainName</jwshc:handler-chain-name>
   <jwshc:handler>
        <handler-name>handlerOne</handler-name>
        <handler-class>com.ca.soa.agent.appserver.jaxrpc.XMLAgentJaxrpcHandlerProxy
        </handler-class>
       </jwshc:handler>
  </jwshc:handler-chain>
</jwshc:handler-config>
```

   HandlerChainName

   Specifies the name of handler chain.

2. Add the JWS annotation @HandlerChain(file="HandlerConfig.xml", name="HandlerChainName") to the web service JWS file.
3. Rebuild the JWS web service.

WebLogic server will invoke SOA Agent JAX-RPC handler.

Note: For more information on SOAP message handlers and handler chains, see the WebLogic documentation.

Use a Handler Chain to Invoke the SOA Agent for JMS Requests

To configure services implemented as JWS files to invoke the SOA Agent for requests received over JMS transport, define the SOA Agent JAX-RPC Handler class in a handler chain configuration file.

To use a handler chain to invoke the SOA Agent for a JWS web service for JMS requests

1. Create a handler chain configuration file that defines a JAX-RPC handler chain. The chain can include as many handler classes as you require but must define the SOA Agent JAX-RPC Handler class first.

   Example HandlerConfig.xml:

```
<jwshc:handler-config xmlns:jwshc="http://www.bea.com/xml/ns/jws"
 xmlns:soap1="http://HandlerInfo.org/Server1"
 xmlns:soap2="http://HandlerInfo.org/Server2"
 xmlns="http://java.sun.com/xml/ns/j2ee" >
  <jwshc:handler-chain>
   <jwshc:handler-chain-name>HandlerChainName</jwshc:handler-chain-name>
   <jwshc:handler>
        <handler-name>handlerOne</handler-name>
        <handler-class>
          com.ca.soa.agent.appserver.jaxrpc.jms.XMLAgentJMSJaxrpcHandlerProxy
        </handler-class>
       </jwshc:handler>
  </jwshc:handler-chain>
</jwshc:handler-config>
```

   HandlerChainName

   Specifies the name of handler chain.

2. Add the JWS annotation @HandlerChain(file="HandlerConfig.xml", name="HandlerChainName") to the web service JWS file.
3. Rebuild the JWS web service.

WebLogic server will invoke SOA Agent JAX-RPC handler for JMS requests.

Note: For more information on SOAP message handlers and handler chains, see the WebLogic documentation.

Configure Policies for the SOA Agent

You create authentication and authorization policies to protect web service resources hosted on WebLogic from their associated WSDL files using the SOA Security Manager Configuration User Interface. For more information, see the *SOA Security Manager Policy Configuration Guide*.

Troubleshoot Issues Related to Protected Web Services Making Use of Apache Commons Logging and Log4j

Because the SOA Agent for WebLogic implements logging using the Apache Commons Logging and Apache log4j packages, class loading problems can occur if protected web service applications also use the Commons Logging and log4j packages.

Commons Logging Class Loading Problem

Symptom:

The SOA Agent for WebLogic uses the Apache Commons Logging package as part of its logging implementation. Class loading conflicts can occur when the SOA Agent is configured to protect web services that also use Commons Logging if the web service and SOA Agent are not using the same version of Commons Logging.

If such a class loading conflict is occurring, WebLogic may log an error output similar to the following during initialization:

```
<Sep 16, 2009 9:15:54 AM EDT> <Error> <HTTP> <BEA-101216> <Servlet: "ServicesServlethttp"
failed to preload on startup in Web application: "XXXXXX.war".
java.lang.ExceptionInInitializerError
….
org.apache.commons.logging.LogConfigurationException: org.apache.commons.logging.LogConfig
urationException: org.apache.commons.logging.LogConfigurationException: Invalid class loader hierarchy.  You
have more than one version of 'org.apache.commons.logging.Log' visible, which is not allowed. (Caused by
org.apache.commons.logging.LogConfigurationException: Invalid class loader hierarchy.  You have more than one
version of 'org.apache.commons.logging.Log' visible, which is not allowed.) (Caused by
org.apache.commons.logging.LogConfigurationException:
org.apache.commons.logging.LogConfigurationException: Invalid class loader hierarchy.  You have more than one
version of 'org.apache.commons.logging.Log' visible, which is not allowed. (Caused by
org.apache.commons.logging.LogConfigurationException: Invalid class loader hierarchy.  You have more than one
version of 'org.apache.commons.logging.Log' visible, which is not allowed.))
      at org.apache.commons.logging.impl.LogFactoryImpl.newInstance(LogFactoryImpl.java:543)
….
```

Solution:

To remove class loading conflicts between the SOA Agent and protected web service, verify that the web service uses the same version of the Apache Commons Logging jar file (commons-logging.jar) as the SOA Agent (version 1.1.1). If the web service is not using the same version, replace the commons-logging.jar used by the web service with the 1.1.1 version.

log4j Class Loading Problem

Symptom:

The SOA Agent for WebLogic uses the Apache log4j package to produce logging output. This can lead to class loading and logging output conflict problems when the SOA Agent is configured to protect web services that also use the log4j package.

If such a conflict is occurring WebLogic logs error output similar to the following:

```
log4j:ERROR A "org.apache.log4j.DailyRollingFileAppender" object is not assignable to a "o
rg.apache.log4j.Appender" variable.
log4j:ERROR The class "org.apache.log4j.Appender" was loaded by
log4j:ERROR [java.net.URLClassLoader@d8116d] whereas object of type
log4j:ERROR "org.apache.log4j.DailyRollingFileAppender" was loaded by [weblogic.utils.clas
sloaders.ChangeAwareClassLoader@10abd60 finder:
weblogic.utils.classloaders.CodeGenClassFinder@5e33d4 annotation:
StockTradeDemoDCC@StockTradeDemoDCC.war].
log4j:ERROR Could not instantiate appender named "CUSTOMER_APPENDER_NAME".
```

This error is caused by the web service class loader loading the specific Appender implementation class: DailyRollingFileAppender and the SOA Agent class loader loading the Appender interface class.

Solution:

Configure the system class loader to load the log4j jar instead of the SOA Agent and web service class loaders by adding log4j.jar to the JVM classpath.

To configure the system class loader to load the log4j package

1. Add *WLS_HOME*/weblogic*XY*/server/lib/log4j.jar to the classpath variable passed into the Java Virtual Machine at startup.

   For a WebLogic Server running as a command line application, make this change by editing the startWeblogic.cmd (Windows) or startWeblogic.sh (UNIX) script located in *WLS_HOME*/user_projects/domains/*your_domain*/bin.

2. Rename the SOA Agent log4j.jar to prevent it from being loaded:

   a. Navigate to *SOA_HOME*/wlsagent/lib/thirdparty.

   b. Rename log4j.jar to log4j.jar_hidden.

3. For each protected web service that also uses the log4j package, rename log4j.jar to prevent it from being loaded:

   a. Navigate to the directory that contains the log4j.jar file used by the web service.

   b. Rename log4j.jar to log4j.jar_hidden.

4. Modify the XML message processing logging configuration:
   a. Navigate to *SOA_HOME*/wlsagent/config
   b. Open the log.config file in text editor.
   c. Locate the line that defines the log4j.category.com.netegrity.tm parameter. For example:

      ```
      log4j.category.com.netegrity.tm=DEBUG
      ```

   d. Append an explicit reference to the Message Processing Appender also declared in the log.config file. For example, if the name of the Daily Rolling File Appender is "A2", the log4j.category.com.netegrity.tm parameter definition line becomes:

      ```
      log4j.category.com.netegrity.tm=DEBUG, A2
      ```