Sun Java System Access Manager Policy Agent 2.2 Guide for Sun Java System Application Server 9.0/Web Services

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 819–6717–11
October, 2006

070420@17039

Contents

Preface

The *Sun™ Java System Access Manager Policy Agent 2.2 Guide for Sun Java System Application Server 9.0/Web Services* provides information about the agent which can be used to provide security for web services deployed in Sun Java System Application Server 9.0. This guide includes an introduction to the draft specifications used to develop the agent and information on how to install it and use it.

Who Should Use This Book

This guide is intended for use by IT professionals, network administrators and software developers who implement a security framework for web services using Sun Java System servers and software. It is recommended that administrators understand the following technologies:

- Lightweight Directory Access Protocol (LDAP)
- Java™
- JavaServer Pages™ (JSP)
- HyperText Transfer Protocol (HTTP)
- HyperText Markup Language (HTML)
- eXtensible Markup Language (XML)
- Web Services Description Language (WSDL)
- Security Assertion Markup Language (SAML)
- SOAP

Related Books

The Sun Java System Access Manager Policy Agent 2.2 for Sun Java System Application Server 9.0/Web Services is used in tandem with both Access Manager and Application Server. The product documentation for these products is available at:

- Sun Java System Access Manager 7.1
- Sun Java System Application Server Platform Edition 9

Additionally, useful information may be found in the documentation for the following Sun Java System products:

- Sun Java System Directory Server

- Sun Java System Web Server
- Sun Java System Web Proxy Server

Accessing Sun Resources Online

For product downloads, professional services, patches, support, and additional developer information, go to:

- Download Center
- Sun Software Services
- Sun Java Systems Services Suite
- Sun Enterprise Services, Solaris Patches, and Support
- Developer Information

Contacting Sun Technical Support

If you have technical questions about this product that are not answered in the product documentation, contact Sun Support Services.

Related Third-Party Web Site References

Third-party URLs are referenced in this documentation set and provide additional, related information. Sun is not responsible for the availability of third-party Web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused by or in connection with the use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Sun Welcomes Your Feedback

Sun Microsystems is interested in improving its documentation and welcomes your comments and suggestions. To share your thoughts, go to `http://docs.sun.com` and click the Send Comments link at the bottom of the page. In the online form provided, include the document title and part number. The part number is a seven-digit or nine-digit number that can be found on the title page of the book or at the top of the document. For example, the title of this book is *Sun Java System Access Manager Policy Agent 2.2 Guide for Sun Java System Application Server 9.0/Web Services* , and the part number is 819–6717.

Documentation, Support, and Training

The Sun web site provides information about the following additional resources:

- Documentation (`http://www.sun.com/documentation/`)
- Support (`http://www.sun.com/support/`)
- Training (`http://www.sun.com/training/`)

Typographic Conventions

The following table describes the typographic conventions that are used in this book.

TABLE P–1 Typographic Conventions

Typeface	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`
`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`** `Password:`
aabbcc123	Placeholder: replace with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file. **Note:** Some emphasized items appear bold online.

Shell Prompts in Command Examples

The following table shows the default UNIX® system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P–2 Shell Prompts

Shell	Prompt
C shell	`machine_name%`
C shell for superuser	`machine_name#`
Bourne shell and Korn shell	`$`
Bourne shell and Korn shell for superuser	`#`

◆ ◆ ◆ CHAPTER 1

Authentication Agents for Web Services

Sun Java™ System web services security providers are authentication agents integrated into a Java™ Platform, Enterprise Edition (Java EE) compatible web container. This chapter provides introductory material and background on these authentication agents. It includes the following sections:

Web Services

A *web service* is an application that exposes some type of functionality using a platform-independent interface. Enterprises use web services as a mechanism for allowing their applications to cross network boundaries and communicate with those of their partners, customers and suppliers. Web services are accessed by sending a request to one of the service's defined endpoints. The following open technologies are used to allow this access.

Web Services Definition Language

Web Services Definition Language (WSDL) is a variant of the eXtensible Markup Language (XML) that is used to describe the public interfaces for a web service. Specifically, this includes the protocol bindings, service endpoints (identified by URLs), and message formats required to interact with it. Once defined, a web service's WSDL description file is published to a worldwide directory of services allowing it to be accessed over the Internet. The directory might use Universal Description, Discovery, and Integration (UDDI) although alternate forms are available.

eXtensible Markup Language

Data exchanged with a web service (requests sent to it and responses received from it) is formatted using XML.

SOAP

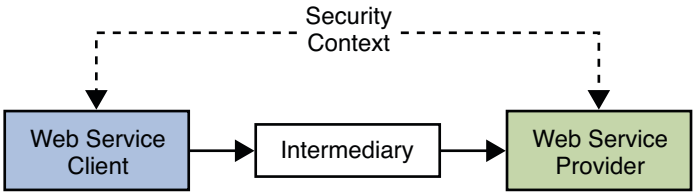
The XML requests and responses generally conform to the SOAP messaging standard.

Hypertext Transfer Protocol (HTTP) SOAP messages are transported between applications using HTTP although File Transfer Protocol (FTP), Simple Mail Transfer Protocol (SMTP) and Extensible Messaging and Presence Protocol (XMPP) can also be used.

The built-in openness of these technologies unfortunately though creates security risks. The following security requirements have been identified and must be supported to insure that the communications between a web service provider (WSP) and a web service client (WSC) are not compromised.

- Data integrity and confidentiality during transport
- Authentication of the sending entity
- Message uniqueness

Securing web services communications was initially addressed on the transport level, relying on securing the HTTP transmissions themselves using Secure Sockets Layer (SSL). This is not adequate though when access to an application is requested through an intermediary. The solution to this is to encrypt the entire request using *message level security* before it is sent. In message level security, security information is contained within the SOAP message or attachment, making it independent of the transport level security. The request may then securely pass through multiple intermediaries before reaching its intended receiver for decryption. The following illustration depicts message level security.



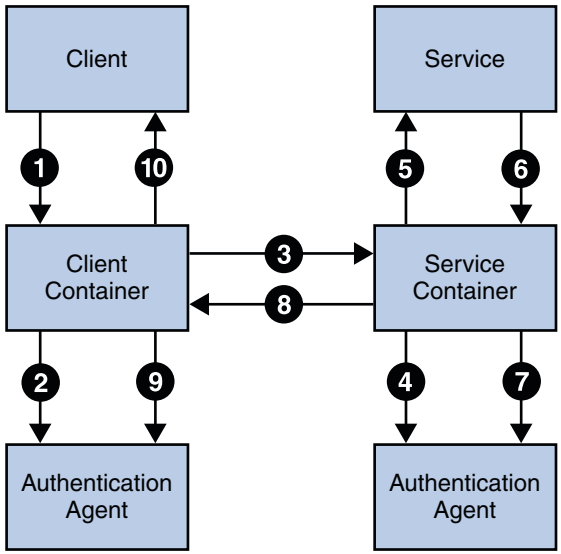
Java Authentication Service Provider Interface for Containers

There are a number of organizations that work on web services security specifications, guidelines, and tools including the World Wide Web Consortium (W3C), the Organization for Advancement of Structured Information Standards (OASIS), the Liberty Alliance Project and the Java Community Process (JCP). The JCP primarily guides the development and approval of Java technical specifications, one of which is the Java Specification Request (JSR) 196. JSR 196 is a draft of the *Java Authentication Service Provider Interface for Containers*. It defines a standard service provider interface (SPI) with which a message level authentication agent can be developed for Java EE containers on either the client side or the server side. These agents may establish the authenticated identities used by the containers allowing:

- A server side agent to verify security tokens or signatures on incoming requests and extract principal data or assertions before adding them to the client security context.
- A client side agent to add security tokens to outgoing requests, sign messages, and interact with the trusted authority to locate targeted web service providers.

Note – The JSR 196 draft specifications are available at `http://www.jcp.org/en/jsr/detail?id=196`.

A typical interaction between a WSC and a WSP begins with a request from the WSC. The container to which the WSP is deployed receives the request and dispatches it to the correct web service to perform the requested operation. When the web service completes the operation, it creates a response that is returned back to the client. The following illustration illustrates this process in more detail with the steps provided below it. This process illustrates a scenario when both client and service web containers employ the Java Authentication SPI.



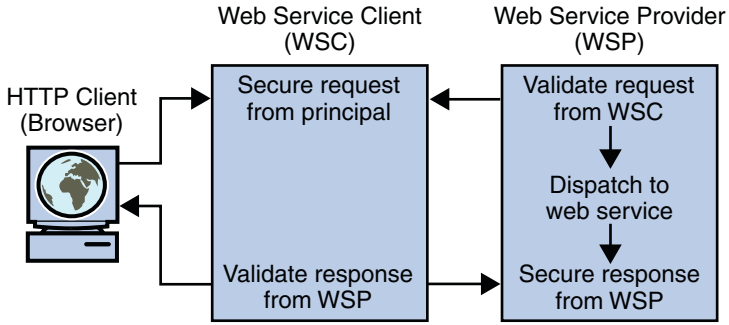
1. The client browser's attempt to invoke a web service is intercepted by the client's web container.
2. The deployed authentication agent on the client's web container is invoked to secure the request (based on the security policy of the web service being invoked).
3. The client's web container sends the secured request message to the web service.
4. The web service's web container receives the secured request message and it's deployed authentication agent is invoked to validate the request and obtain the identity of the caller.
5. Assuming successful authentication, the web service's web container invokes the requested web service.
6. This action (the invocation of the web service) is returned to the web service's web container as a response.
7. The deployed authentication agent on the web service's web container is invoked to secure the response message.

8. The web service's web container sends the secured response message to the client.
9. The deployed authentication agent on the client's web container is invoked to validate the secured response message.
10. The invocation of the web service is returned to the client browser.

The JSR–196 SPI is structured so that the security processes can be delegated to an authentication agent at any of four interaction points in this scenario. The four points represent the methods of the corresponding `ClientAuthModule` and `ServerAuthModule` interfaces defined by the SPI and include:

- Securing a request on the client side
- Validating a request on the provider side
- Securing a response on the provider side
- Validating a response on the client side

Thus, when a WSC and WSP are both deployed in a Java EE web container protected by a JSR–196 authentication agent, the initial request from the WSC is intercepted by the authentication agent on the client side. The client side agent queries a *trusted authority* (for example, the Discovery Service in Access Manager) to retrieve the necessary authorization credentials and secure them to the request. The request is then passed to the WSP. The authentication agent on the provider side receives the request to validate the authorization credentials. If validation is successful, the request is exposed to the web service and a response is created using the sender's credentials and the application specific request. The response is then intercepted by the authentication agent on the provider side to secure it and return it to the WSC. Upon receiving the response, the authentication agent on the client side validates it and dispatches it to the client application. This is illustrated in the following graphic.



Sun Java System Access Manager Policy Agent 2.2 for Sun Java System Application Server 9.0 / Web Services

In general, securing web services involves establishing trust between a WSC and a WSP. For identity-based web services specifically (for example, a calendar service), the WSP would have to trust the WSC to have authenticated the user, or the WSC would have to include the user's credentials as part of the web service request. The distinguishing factor is that identity-based web services authenticate both the WSC and the user's identity. (The user must be authenticated so that the WSC can send the user's *token* to the WSP in a SOAP security header.)

The Sun Java System Access Manager Policy Agent 2.2 for Sun Java System Application Server 9.0 / Web Services plugs into Application Server Platform Edition 9.0, provides message level security, and supports both Liberty Alliance Project token profiles as well as Web Services-Interoperability Basic Security Profiles (WS-I BSP). (A *profile* defines the HTTP exchanges required to transfer XML requests and responses between web service clients and providers.)

Note – More information on Application Server Platform Edition 9.0 can be found in the collection of documentation provided for the product located at `http://docs.sun.com/coll/1343.3`.

This release of the agent uses an instance of Access Manager for all authentication decisions. Web services requests and responses are passed to Access Manager authentication modules using standard Java representations based on the transmission protocol. Currently, the Policy Agent 2.2 for Sun Java System Application Server 9.0 / Web Services provides the following agents:

- "HTTP Authentication Agent" on page 13
- "SOAP Authentication Agent" on page 15

HTTP Authentication Agent

The HTTP authentication agent protects the endpoints of a web service that uses HTTP for communication. After the HTTP authentication agent is deployed in an instance of Application Server on the WSP side, all HTTP requests for access to web services protected by the agent are redirected to the login and authentication URLs defined in the Access Manager `AMConfig.properties` file on the WSC side. `AMConfig.properties` is located in *javaee.home*`/domains/`*domain_name*`/config` when the Java Platform, Enterprise Edition (Java EE) 5 SDK is installed and in *javaee.home*`/addons/amserver` when the Java EE 5 Tools Bundle is installed. The configurable properties are:

- **`com.sun.identity.loginurl=`***amserver_protocol*`://`*amserver_host*`:`*amserver_port*`/amserver/UI/`

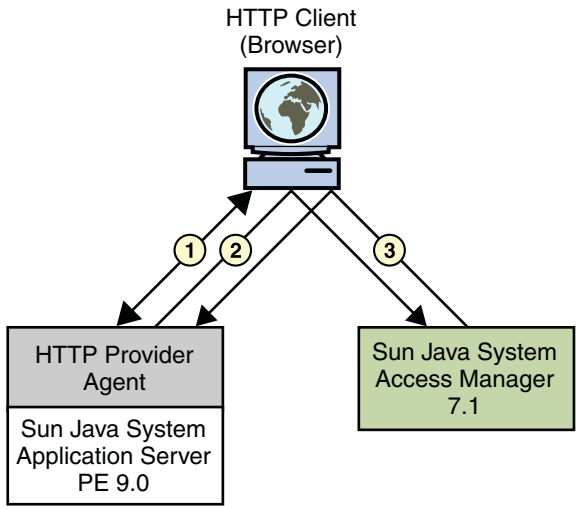
- **com.sun.identity.liberty.authnsvc.url=***amserver_protocol***://***amserver_host***:***amserver_port***/amser**

Note – Application Server 9 has the ability to configure only one HTTP agent per instance. Therefore, all authentication requests for all web applications hosted in the container will be forwarded to the one configured agent.

When the WSC makes a request to access a web application protected by an HTTP authentication agent (**1** in the illustration below), the agent intercepts the request and redirects it (via the browser) to Access Manager for authentication (**2**). Upon successful authentication, a response is returned to the application, carrying a token as part of the Java EE Subject (**3**). This token is used to bootstrap the appropriate Liberty ID-WSF security profile. If the response is successfully authenticated, the request is granted (**3**).

Note – For this release, the HTTP authentication agent is used primarily for bootstrapping. Future releases will contain information on how to protect web applications.

The following figure illustrates the interactions described.



Note – The functionality of the HTTP Provider agent is similar in to that of the Sun Java System Access Manager Java EE agents when used in SSO ONLY mode. This is a non restrictive mode that uses only the Access Manager Authentication Service to authenticate users attempting access. For more information on Java EE agents, see the *Sun Java System Access Manager Policy Agent 2.2 User's Guide*.

SOAP Authentication Agent

The SOAP authentication agent secures SOAP messages between a WSC and a WSP. The agent can be configured for use as an authentication provider on either the WSC server or the WSP server. This initial release encapsulates the Liberty Identity Web Services Framework (Liberty ID-WSF) SOAP Binding Specification as implemented by Access Manager and supports the following:

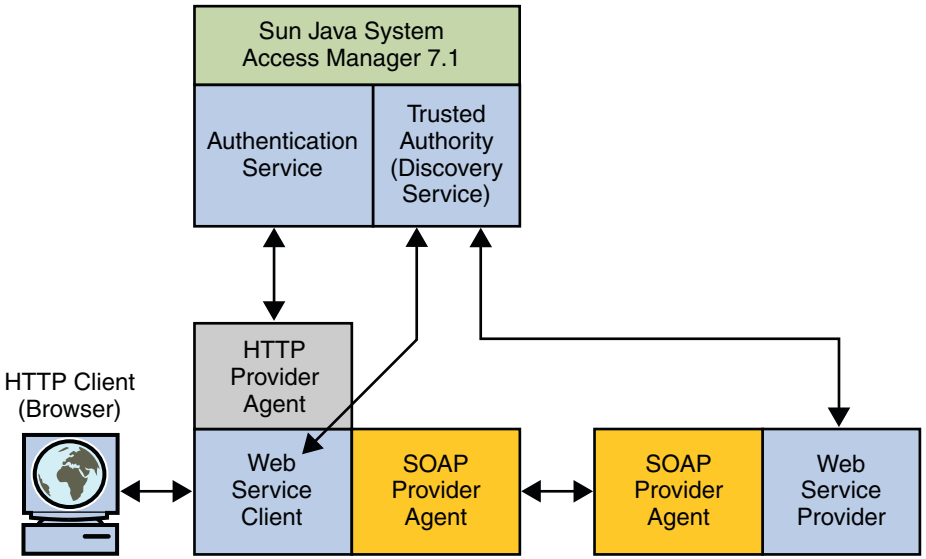
- "Supported Liberty Alliance Project Security Tokens" on page 15
- "Supported Web Services-Interoperability Basic Security Profile Security Tokens" on page 16

Note – The configuration process for the SOAP authentication agent is described in "Installing the Policy Agent 2.2 for Sun Java System Application Server 9.0 / Web Services" on page 22.

Supported Liberty Alliance Project Security Tokens

In a scenario where security is enabled using Liberty Alliance Project tokens, the HTTP client requests (via the WSC) access to a service. The HTTP authentication agent redirects the request to the Access Manager Authentication Service for authentication and to determine the security mechanism registered by the WSP and obtain the security tokens expected. After a successful authentication, the WSC provides a SOAP body while the SOAP authentication agent on the WSC side inserts the security header and a token. The message is then signed before the request is sent to the WSP.

When received by the SOAP authentication agent on the WSP side, the signature and security token in the SOAP request are verified before forwarding the request on to the WSP itself. The WSP then processes it and returns a response, signed by the SOAP authentication agent on the WSP side, back to the WSC. The SOAP authentication agent on the WSC side then verifies the signature before forwarding the response on to the WSC. The following diagram illustrates the interactions as described.



The following Liberty Alliance Project security tokens are supported in this release:

X.509 A secure web service uses a PKI (public key infrastructure) in which the web service consumer supplies a public key as the means for identifying the requester and accomplishing authentication with the web service provider. Authentication with the web service provider using processing rules defined by the Liberty Alliance Project.

BearerToken A secure web service uses the Security Assertion Markup Language (SAML) *SAML Bearer token* confirmation method. The web service consumer supplies a SAML assertion with public key information as the means for authenticating the requester to the web service provider. A second signature binds the assertion to the SOAP message This is accomplished using processing rules defined by the Liberty Alliance Project

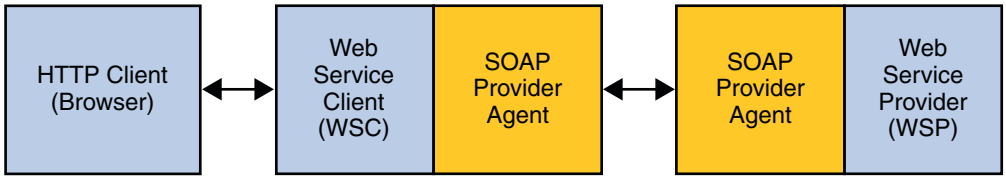
SAMLToken A secure web service uses the SAML *holder-of-key* confirmation method. The web service consumer adds a SAML assertion and a digital signature to a SOAP header. A sender certificate or public key is also provided with the signature. This is accomplished using processing rules defined by the Liberty Alliance Project.

Supported Web Services-Interoperability Basic Security Profile Security Tokens

In a scenario where security is enabled using Web Services-Interoperability Basic Security Profile (WS-I BSP) tokens, the HTTP client requests (via the WSC) access to a service. The SOAP authentication agent redirects the request to the Access Manager Authentication Service for authentication and to determine the security mechanism registered by the WSP and obtain the expected security tokens. After a successful authentication, the WSC provides a SOAP body

while the SOAP authentication agent on the WSC side inserts the security header and a token. The message is then signed before the request is sent to the WSP.

When received by the SOAP authentication agent on the WSP side, the signature and security token in the SOAP request are verified before forwarding the request on to the WSP itself. The WSP then processes it and returns a response, signed by the SOAP authentication agent on the WSP side, back to the WSC. The SOAP authentication agent on the WSC side then verifies the signature before forwarding the response on to the WSC. The following diagram illustrates the interactions as described.



The following WS-I BSP security tokens are supported in this release.

User Name A secure web service requires a user name, password and, optionally, a signed the request. The web service consumer supplies a username token as the means for identifying the requester and a password, shared secret, or password equivalent to authenticate the identity to the web service provider.

X.509 A secure web service uses a PKI (public key infrastructure) in which the web service consumer supplies a public key as the means for identifying the requester and accomplishing authentication with to the web service provider.

SAML-Holder-Of-Key A secure web service uses the SAML *holder-of-key* confirmation method. The web service consumer supplies a SAML assertion with public key information as the means for authenticating the requester to the web service provider. A second signature binds the assertion to the SOAP payload.

SAML-SenderVouches A secure web service uses the SAML *sender-vouches* confirmation method. The web service consumer adds a SAML assertion and a digital signature to a SOAP header. A sender certificate or public key is also provided with the signature.

◆ ◆ ◆ CHAPTER 2

Installing the Access Manager Policy Agent 2.2 for Application Server 9.0 / Web Services

The Sun Java™ System Access Manager Policy Agent 2.2 for Sun Java System Application Server 9.0 / Web Services is installed in a Java 2 Enterprise Edition (Java EE) container (for example, Sun Java System Application Server), and used in conjunction with Sun Java System Access Manager. This chapter contains installation instructions and includes the following sections:

Installation Overview

The Access Manager Policy Agent 2.2 for Sun Java System Application Server 9.0 / Web Services is installed when installing any of the following bundles.

- Java EE 5 SDK Update 3 Preview

  Download the bits by clicking either Download, Download with JDK, or Download with Tools. This is the most recent bundle which includes the full version of Access Manager 7.1.

- Java Application Platform SDK Update 2

  Download the bits by clicking either Download, Download with JDK, or Download with Tools. This bundle includes the beta version of Access Manager 7.1.

- Java EE 5 SDK Update 2

  Download the bits by clicking Download with Tools only. This bundle includes the beta version of Access Manager 7.1.

- NetBeans™ Enterprise Pack 5.5

  Download the bits by clicking Download. This bundle includes the beta version of Access Manager 7.1.

Additionally, the Sun Java System Access Manager 7.1 web archive (WAR) will be generated and deployed. Although this deployment process has been automated by the installers of the respective products, information on the Access Manager 7.1 WAR itself can be found in Chapter 12, "Deploying Access Manager as a Single WAR File," in *Sun Java System Access Manager 7.1 Postinstallation Guide*.

Note – If you have already installed Access Manager 7.1 and the Policy Agent 2.2 for Sun Java System Application Server 9.0 / Web Services, you can move on to Chapter 3.

The installation procedures documented in this chapter are also performed by the installers of the respective products. They are documented here for use with third-party Java EE containers and for informational purposes.

Installing Access Manager

The initial step in installing Access Manager 7.1 is to deploy the Access Manager WAR as a web application using the Application Server administration console. Instructions on how to do this can be found in "Downloading an Access Manager 7.1 WAR File" in *Sun Java System Access Manager 7.1 Postinstallation Guide*. Following is the procedure to complete the installation of Access Manager 7.1.

▼ To Complete the Installation of Access Manager 7.1

The following configurations will complete the installation of Access Manager 7.1.

Before You Begin These instructions assume that Sun Java System Application Server Platform Edition 9.0 has already been installed and the Access Manager WAR has already been deployed. For more information, see *Sun Java System Application Server Platform Edition 9 Installation Guide* and "Downloading an Access Manager 7.1 WAR File" in *Sun Java System Access Manager 7.1 Postinstallation Guide* respectively.

1 Add the following as Java security permissions to the `server.policy` file of the Application Server.

Each Application Server domain has its own standard J2SE policy file named `server.policy`. It is located in the *domain-dir*`/config` directory. More information can be found in "The server.policy File" in *Sun Java System Application Server Platform Edition 9 Developer's Guide*.

```
// ADDITIONS FOR Access Manager
 grant codeBase "file:\${com.sun.aas.instanceRoot}/applications/j2ee-modules/amserver/-" {
      permission java.net.SocketPermission "*", "connect,accept,resolve";
      permission java.util.PropertyPermission "*", "read, write";
```

```
    permission java.lang.RuntimePermission "modifyThreadGroup";
    permission java.lang.RuntimePermission "setFactory";
    permission java.lang.RuntimePermission "accessClassInPackage.*";
    permission java.util.logging.LoggingPermission "control";
    permission java.lang.RuntimePermission "shutdownHooks";
    permission javax.security.auth.AuthPermission "insertProvider.Mozilla-JSS";
    permission java.security.SecurityPermission "putProviderProperty.Mozilla-JSS";
    permission javax.security.auth.AuthPermission "getLoginConfiguration";
    permission javax.security.auth.AuthPermission "setLoginConfiguration";
    permission javax.security.auth.AuthPermission "modifyPrincipals";
    permission javax.security.auth.AuthPermission "createLoginContext.*";
    permission java.security.SecurityPermission "insertProvider.Mozilla-JSS";
    permission javax.security.auth.AuthPermission "putProviderProperty.Mozilla-JSS";
    permission java.io.FilePermission "<<ALL FILES>>", "execute,delete";
    permission java.util.PropertyPermission "java.util.logging.config.class", "write";
    permission java.security.SecurityPermission "removeProvider.SUN";
    permission java.security.SecurityPermission "insertProvider.SUN";
    permission java.security.SecurityPermission "removeProvider.Mozilla-JSS";
    permission javax.security.auth.AuthPermission "doAs";
    permission java.util.PropertyPermission "java.security.krb5.realm", "write";
    permission java.util.PropertyPermission "java.security.krb5.kdc", "write";
    permission java.util.PropertyPermission "java.security.auth.login.config", "write";
    permission java.util.PropertyPermission "user.language", "write";
    permission javax.security.auth.kerberos.ServicePermission "*", "accept";
    permission javax.net.ssl.SSLPermission "setHostnameVerifier";
    permission java.security.SecurityPermission "putProviderProperty.IAIK";
    permission java.security.SecurityPermission "removeProvider.IAIK";
    permission java.security.SecurityPermission "insertProvider.IAIK";

};
// END OF ADDITIONS FOR Access Manager
```

2. **Auto POST the following configuration data to** `configurator.jsp`**.**

   `configurator.jsp` is the dynamic configuration page for the Access Manager single WAR application. It is used after deploying the WAR. When you launch Access Manager 7.1, if you have not yet configured the application, you will be directed to `configurator.jsp`. If Access Manager 7.1 is already configured, you will be directed to the Access Manager Console login page. `configurator.jsp` is located in the *Access Manager_protocol*`://`*Access Manager_host*`:`*Access Manager_port*`/amserver/` directory. The required request parameters in `configurator.jsp` and accompanying values are:

   - **SERVER_URL**: The fully qualified name and port of the host on which Access Manager is installed. Use the format:

     Access Manager_protocol`://`*Access Manager_host*`:`*Access Manager_port*
   - **SERVER_URI**: By default, the value is `/amserver`.
   - **BASE_DIR**: The path to the directory in which Access Manager will create its flat file database. By default, `/tmp/amserver`.

- **ADMIN_PWD**: The password of the top-level administrator; by default, `admin123`.
- **ADMIN_CONFIRM_PWD**: Confirmation of the password defined in **ADMIN_PWD**.

More information on the `configurator.jsp` can be found in Chapter 12, "Deploying Access Manager as a Single WAR File," in *Sun Java System Access Manager 7.1 Postinstallation Guide*.

Note – *Auto POST* means to use an HTTP POST of the required request parameters for this JavaServer Page (JSP) programmatically (from the installer code itself) without showing these parameters or prompting the user.

---

3 Check that the Access Manager server is running using the following URL:

Access Manager_protocol`://`*Access Manager_host*`:`*Access Manager_port*`/amserver/isAlive.jsp`

4 Log in to Access Manager as the top-level administrator using the following URL:

Access Manager_protocol`://`*Access Manager_host*`:`*Access Manager_port*`/amserver`

By default, the top-level administrator is `amadmin`, and the `amadmin` password is `admin123`.

Installing the Policy Agent 2.2 for Sun Java System Application Server 9.0 / Web Services

Following is the procedure to complete the installation of the Policy Agent 2.2 for Sun Java System Application Server 9.0 / Web Services.

▼ To Complete the Installation of the Policy Agent 2.2 for Sun Java System Application Server 9.0 / Web Services

Before You Begin The initial step in installing the Policy Agent 2.2 for Sun Java System Application Server 9.0 / Web Services is to deploy the Access Manager WAR as a web application using the Application Server administration console. See "Installing Access Manager" on page 20 if this has not been done.

Note – *javaee.home* is a variable that should be replaced with the installation directory of the Java EE 5 SDK.

---

1 Note the directory name and the path to the directory into which the following files are placed:

- `amWebServicesProvider.jar`
- `amclientsdk.jar`

- `AMConfig.properties`
- `amclientkeystore.jks`
- `.storepass`
- `.keypass`

If you used one of the installers, the files were put in a particular directory: */javaee.home*`/addons/accessmanager` for installations of Java Application Platform SDK (when Download or Download with JDK is selected), and */javaee.home*`/addons/amserver` for installations of Java Application Platform SDK or Java EE 5 SDK Update 1 (when Download with Tools is selected), and NetBeans Enterprise Pack 5.5. Be sure to make a note of this directory and path. Otherwise, put the files in a directory and make a note of the directory and path in which they were placed.

2 Modify the global Java Virtual Machine (JVM) settings in Application Server by adding the following to the classpath suffix:

- `amwebServiceProvider.jar` (including the complete path)
- `amclientsdk.jar` (including the complete path)
- The complete path to the directory which contains the client's `AMConfig.properties`:
  - */javaee.home*`/domains/`*domain_name*`/config` for installations of Java Application Platform SDK (when Download or Download with JDK is selected).
  - */javaee.home*`/addons/amserver` for installations of Java Application Platform SDK or Java EE 5 SDK Update 1 (when Download with Tools is selected) and NetBeans Enterprise Pack 5.5.

3 Add the following web services security providers configurations to the `domain.xml` file as per Application Server guidelines.

`domain.xml` is located in the */ApplicationServer-install*`/domains/domain1/config` directory and contains most of the Application Server configuration information.

Note – More information can be found in Chapter 1, "The domain.xml File," in *Sun Java System Application Server Platform Edition 9 Administration Reference.*

The following provider code fragment needs to be added under the `<message-security-config auth-layer="HttpServlet">` tag:

```
<provider-config class-name="com.sun.identity.agents.jsr196.as9soap.AMHttpAuthModule"
provider-id="AMHttpProvider" provider-type="server">
<request-policy auth-source="content"/>
<response-policy auth-source="content"/>
</provider-config>
```

The following provider code fragments need to be added under the `<message-security-config auth-layer="SOAP">` tag:

```
<provider-config class-name="com.sun.identity.agents.jsr196.as9soap.AMServerAuthModule"
provider-id="AMServerProvider-SAML-HolderOfKey" provider-type="server">
<request-policy auth-source="content"/>
<response-policy auth-source="content"/>
<property name="providername" value="SAML-HolderOfKey"/>
</provider-config>
```

```
<provider-config class-name="com.sun.identity.agents.jsr196.as9soap.AMServerAuthModule"
provider-id="AMServerProvider-SAML-SenderVouches" provider-type="server">
<request-policy auth-source="content"/>
<response-policy auth-source="content"/>
<property name="providername" value="SAML-SenderVouches"/>
</provider-config>
```

```
<provider-config class-name="com.sun.identity.agents.jsr196.as9soap.AMServerAuthModule"
provider-id="AMServerProvider-X509Token" provider-type="server">
<request-policy auth-source="content"/>
<response-policy auth-source="content"/>
<property name="providername" value="X509Token"/>
</provider-config>
```

```
<provider-config class-name="com.sun.identity.agents.jsr196.as9soap.AMServerAuthModule"
provider-id="AMServerProvider-LibertySAMLToken" provider-type="server">
<request-policy auth-source="content"/>
<response-policy auth-source="content"/>
<property name="providername" value="LibertySAMLToken"/>
</provider-config>
```

```
<provider-config class-name="com.sun.identity.agents.jsr196.as9soap.AMClientAuthModule"
provider-id="AMClientProvider" provider-type="client">
<request-policy auth-source="content"/>
<response-policy auth-source="content"/>
<property name="providername" value="wsc"/>
</provider-config>
```

```
<provider-config class-name="com.sun.identity.agents.jsr196.as9soap.AMServerAuthModule"
provider-id="AMServerProvider-UserNameToken" provider-type="server">
<request-policy auth-source="content"/>
<response-policy auth-source="content"/>
<property name="providername" value="UserNameToken"/>
</provider-config>
```

```
<provider-config class-name="com.sun.identity.agents.jsr196.as9soap.AMServerAuthModule"
provider-id="AMServerProvider-LibertyX509Token" provider-type="server">
<request-policy auth-source="content"/>
<response-policy auth-source="content"/>
<property name="providername" value="LibertyX509Token"/>
</provider-config>
```

```
<provider-config class-name="com.sun.identity.agents.jsr196.as9soap.AMServerAuthModule"
provider-id="AMServerProvider-LibertyBearerToken" provider-type="server">
<request-policy auth-source="content"/>
<response-policy auth-source="content"/>
<property name="providername" value="LibertyBearerToken"/>
</provider-config>
```

```
<provider-config class-name="com.sun.identity.agents.jsr196.as9soap.AMServerAuthModule"
provider-id="AMServerProvider" provider-type="server">
<request-policy auth-source="content"/>
<response-policy auth-source="content"/>
<property name="providername" value="wsp"/>
</provider-config>
```

4 Modify `AMConfig.properties` **as follows:**

```
JAVA_HOME=/usr/java

# AM Server Information
# Protocol can be either http or https
SERVER_PROTOCOL=amserver_protocol
SERVER_HOSTNAME=amserver_host
SERVER_PORT=amserver_port

# Application username and password
APPLICATION_USERNAME=amadmin
APPLICATION_PASSWORD=admin123

NAMING_URL=amserver_protocol://amserver_host:amserver_port/amserver/namingservice

# Debug information
DEBUG_LEVEL=error
DEBUG_DIR=/tmp/amclient

# Cookie information
AM_COOKIE_NAME=iPlanetDirectoryPro

# SAML xml signature keystore file, keystore password file,
# key password file and Liberty trusted CA aliases.
# path_to_file should be replaced by the appropriate value as below:
# /javaee.home/addons/accessmanager for installations of Java Application Platform SDK
# (when Download or Download with JDK is selected), and /javaee.home/addons/amserver
```

```
# for installations of Java Application Platform SDK or Java EE 5 SDK Update 1
# (when Download with Tools is selected), and NetBeans Enterprise Pack 5.5 (when Download is selected).
SAML_KEYSTORE=/path_to_file/amclientkeystore.jks
SAML_STOREPASS=/path_to_file/.storepass
SAML_KEYAPSS=/path_to_file/.keypass
LIBERTY_TRUSTEDCA_ALIASES=amserver:<amserver_host>

# Login URL and Authentication service URL for Liberty use case
LOGIN_URL=amserver_protocol://amserver_host:amserver_port/amserver/UI/Login
LIBERTY_AUTHSVC_URL=amserver_protocol://amserver_host:amserver_port/amserver/Liberty/authnsvc
```

Note – The directory specified as a value for `DEBUG_DIR` in `AMConfig.properties` should be different than the one specified as the value for `BASE_DIR` in "Installing Access Manager" on page 20.

5 **Restart the Application Server.**

Uninstallation

The following procedure is to uninstall Access Manager 7.1 and the Policy Agent 2.2 for Sun Java System Application Server 9.0 / Web Services.

▼ To Uninstall Access Manager 7.1 and the Policy Agent 2.2 for Sun Java System Application Server 9.0 / Web Services

1 **Undeploy the `amserver` web application using the Application Server administration console.**

2 **Note the path to the Access Manager flat file directory from the `AccessManager/AMConfig_`*ApplicationServer-base*`_domains_`*Domain name*`_applications_j2ee-modules_amserver_` file located under the home directory of the user who has installed and configured Access Manager.**

For example, the `AccessManager/AMConfig_opt_SUNWappserver_domains_domain1_applications_j2ee-modules_amserve` file under the user's home directory.

Note – The location of the user's directory depends on the user and operating system. For example, on a UNIX system, if the user is `root`, the user's home directory is `/`. If the user is `xyz`, the user's home directory is `/home/xyz`.

3. **Delete the Access Manager flat file directory.**

4. **Delete the** `AccessManager/AMConfig_`*ApplicationServer-base*`_domains_`*Domain name*`_applications_j2ee-modules_amserver_` **file under the user's home directory.**

5. **Restart the Application Server.**

◆ ◆ ◆ CHAPTER 3

Using the Access Manager Policy Agent 2.2 for Application Server 9.0 / Web Services

Sun Java™ System Access Manager Policy Agent 2.2 for Sun Java System Application Server 9.0 / Web Services is installed with Java BluePrints. This chapter contains information regarding these BluePrints and other administrative procedures. It includes the following sections:

Java BluePrints

Access Manager Policy Agent 2.2 for Sun Java System Application Server 9.0 / Web Services is installed with Sun Microsystems' best practice applications called Java BluePrints. The Java BluePrints program defines the application programming model for the Java Enterprise Edition (Java EE) platform. The following sections describe the included BluePrints which focus on web services security.

Stock Service BluePrint

This BluePrint focuses on building a web service provider (WSP) and a web service client (WSC), authenticating the WSC before access to the service is given, and guaranteeing the integrity of the authentication data. This is accomplished by using Web Services Interoperability Basic Security Profile (WS-I BSP) tokens to secure communications between the participants. The BluePrint encompasses a web service that provides details for a given stock symbol. The instructions for the Stock Service BluePrint is the `index.html` file found in */javaee.home*`/blueprints/ws-security/stock-jaxrpc/` directory.

Calendar Service BluePrint

This BluePrint focuses on securing an identity-based WSP. Identity-based web services must know the identity of the user accessing the service. The Calendar Service BluePrint is a calendar service which uses the identity of the user to enforce permission checks on the event(s) being accessed. In securing an identity-based WSP, the identity accessing the service (via the WSC) is authenticated before being given access. Additionally, the WSC would also be authenticated by the WSP before being given access. The instructions for the Calendar Service BluePrint is the `index.html` file found in */javaee.home*`/blueprints/ws-security/calendar-jaxrpc/` directory.

Keystores

J2EE agents work with Access Manager to protect resources. However, for security purposes, these two pieces of software can only interact with each other after the J2EE agent authenticates with Access Manager by supplying an agent profile name and password. The agent profiles we are using (`wscWSC`, `LibertyBearerToken`, etc.) are configured to use the following keystores, by default:

- The Access Manager Policy Agent 2.2 for Application Server 9.0 / Web Services uses the client keystore shipped with the Java EE SDK, `amclientkeystore.jks`, as its default client keystore. It is located in *javaee.home*`/addons/accessmanager` for installations of Java Application Platform SDK (when Download or Download with JDK is selected), and in *javaee.home*`/addons/amserver` for installations of Java Application Platform SDK or Java EE 5 SDK Update 1 (when Download with Tools is selected), and NetBeans Enterprise Pack 5.5
- The single WAR instance of Access Manager uses the server keystore shipped with the Java EE SDK, `keystore.jks`, for its default keystore. It is located in *javaee.home*`/domains/`*domain_name*`/config/amflatfiledir/amserver` for installations of Java Application Platform SDK (all downloads), Java EE 5 SDK Update 1 (only when Download with Tools is selected), and NetBeans Enterprise Pack 5.5.

You can configure for a custom keystore, though. The following procedure describes the necessary steps.

Note – For more information on agent profiles, see "Agents" in *Sun Java System Access Manager 7 2005Q4 Administration Guide* in the *Sun Java System Access Manager 7 2005Q4 Administration Guide*

During the installation of the J2EE agent, you must provide a valid agent profile name and the respective password to enable authentication attempts to succeed.

▼ To Configure for a Custom Keystore

1. **Export the certificate for the alias `amserver` using the following command:**

   `keytool -list -keystore` *keystore_file* **`-alias amserver -rfc`**

2. **Store the exported X509 certificate, using the RFC format, in a file named `server.txt`.**

3. **Export the certificate from your custom keystore using the following command:**

   `keytool -list -keystore` *custom_keystore_file* **`-alias`** *key alias* **`-rfc`**

   key alias is the alias of the private key used by the WSC to sign SOAP messages.

4. **Store the exported X509 certificate, using the RFC format, in a file named `client.txt`.**

5. **Import the stored `amserver` certificate into the agent's custom keystore file using the following command:**

   `keytool -import -keystore` *custom_keystore_file* **`-alias`** *custom_alias* **`-file server.txt`**

6. **Import the stored custom keystore's certificate into the Access Manager keystore file using the following command:**

   `keytool -import -keystore` *custom_keystore_file* **`-alias`** *custom_alias* **`-file client.txt`**

7. **Generate a Discovery Service token for the WSC that will use the custom keystore with the following command:**

   `keytool -import -keystore` *custom_keystore.jks* **`-alias amserver -file server.txt`**

   This allows the WSP which uses the custom keystore to trust the Access Manager Discovery Service.

8. **Edit the following properties in the client's `AMConfig.properties`:**

   - **`com.sun.identity.liberty.ws.wsc.certalias=`***alias_of_private_key_in_custom_client_keystore*

     This certificate is used by the Liberty X509/SAML profiles for signing the SOAP messages.

   - **`com.sun.identity.liberty.ws.trustedca.certaliases=`***alias_of_private_key_in_custom_serve*

   `AMConfig.properties` is located in *javaee.home*`/domains/`*domain_name*`/config` when the Java Platform, Enterprise Edition (Java EE) 5 SDK is installed and in *javaee.home*`/addons/amserver` when the Java EE 5 Tools Bundle is installed.

Index