Technical Note: Using Access Manager Distributed Authentication

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

Part No: 819–4566–11

060215@13996

Sun Java™ System Access Manager 7 2005Q4 Technical Note: Using Access Manager

Distributed Authentication

This document contains the following sections:

Revision History

Release Date	Description of Changes
Oct. 19, 2005	Initial publication of this technical note.
Nov. 14, 2005	Added distributed authentication package information to the Building the Distributed Authentication Web Application section.
Feb. 5, 2006	Updated with fixes and maintenance.

Overview

In order for authentication to occur, Access Manager must be able to send HTTP or HTTPS packets with the authentication interface directly to the web browser. This deployment architecture requires opening holes in any firewalls between the end user and Access Manager.

In order to do this securely, Access Manager 7 supports a distributed authentication user interface web application. With distributed authentication, an additional server provides the authentication UI web application. The authentication UI servers exist solely for the purpose of serving up the authentication interface to web browsers. They let users eliminate the need for opening holes in firewalls between the end user and Access Manager. The following figure outlines the components and runtime flow of a basic distributed authentication deployment scenario.

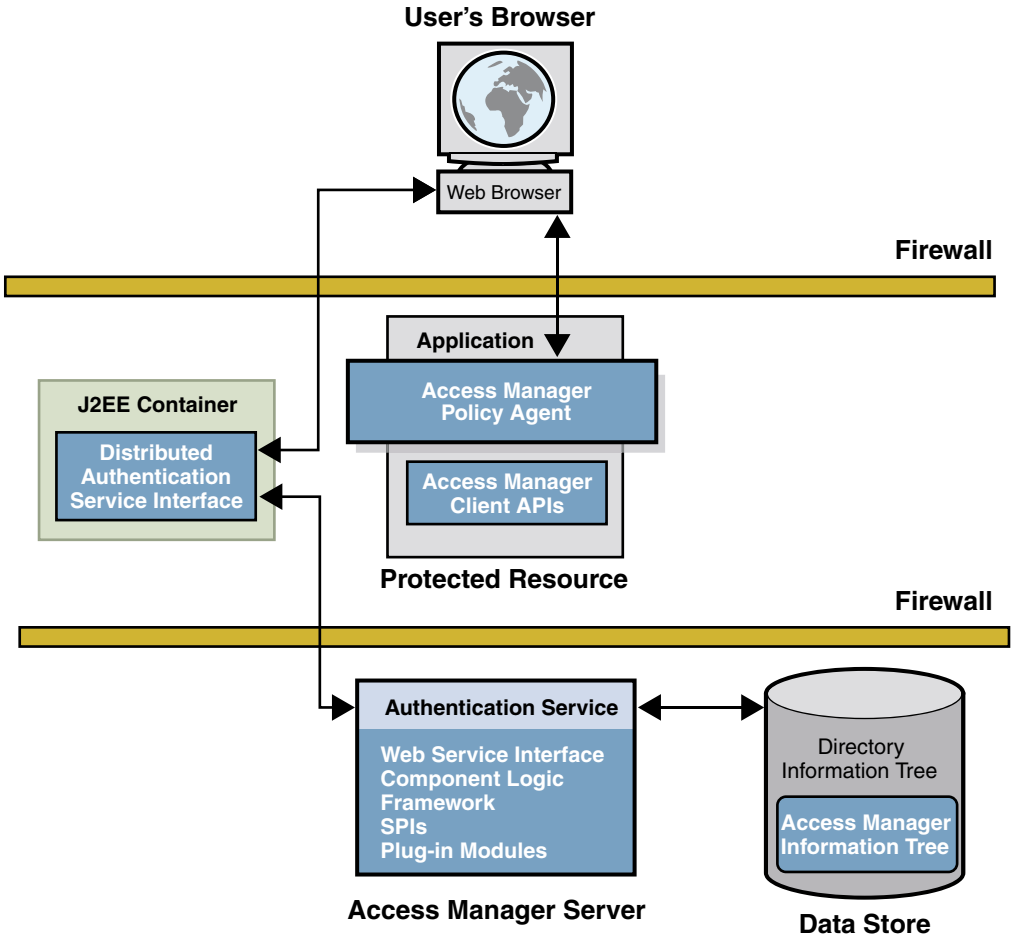

FIGURE 1 Distributed Authentication Overview

The Distributed Authentication UI service has the following dependencies:

- The `SUNWamclnt` (Access Manager Client SDK package) is installed.
- The `SUNWjato` (JATO package) is installed.
- Access Manager Server is available (remotely).
- The Client SDK will be able to communicate with Access Manager Server using http(s).

The Certificate, HTTP Basic, and MSISDN authentication modules are not supported through the distributed authentication interface.

The following service flow outlines how distributed authentication works in a typical scenario:

1. The web browser makes a request to the resource protected by a policy agent.
2. If there is no cookie containing an SSO token, the policy agent issues a redirect to its authentication URL. With the distributed authentication interface, the authentication URL is the URL of the distributed authentication service.
3. The browser follows the redirect and makes a request
4. The distributed authentication service recognizes the inbound request. Using the parameters in the request, it communicates with the authentication server (Access Manager) on the back end to determine the appropriate authentication instance, and gets the necessary callbacks to use in the presentation framework/layer. It also determines which presentation to use.
5. The distributed authentication service, using the information from the server, returns a presentation extraction page back to the web browser with the appropriate callbacks info from the server.
6. The browser replies with the credentials in a POST operation.
7. The distributed authentication service gets the credentials and passes them to Access Manager.
8. Access Manager authenticates using the appropriate authentication instance. If successful, it passes back the SSOToken, or it passes back the appropriate error information.
9. If successful, the distributed authentication service, replies with a 302 redirect back to the originally requested resource, which includes the SSOToken in a `set-cookie` header.

Installing and Customizing the Distributed Authentication Interface

You can install the remote authentication user interface component on any servlet-compliant web container within the non-secure layer of an Access Manager deployment. The remote component works with Authentication client APIs and authentication utility classes to authenticate web users. The remote component is customizable and uses a JATO presentation framework.

If you are deploying multiple Distributed Authentication servers behind a load balancer, stickiness is not required for the load balancer to talk to only one Distributed Authentication server for authentication process completion.

Supported JDK Versions

Supported JDK versions are J2SE 1.4.1, or higher. In order to run using JDK 1.3.1, the following additional jar files would be needed in the CLASSPATH:

- Java Authentication and Authorization Service (JAAS) located at `http://java.sun.com/products/jaas/`
- Java Secure Socket Extension (JSSE) for SSL located at `http:///java.sun.com/products/jsse/`
- JDK Logging (`jdk_logging.jar`). Can be obtained from `SUNWamsdk` package for Solaris and `sun-identity-sdk` RPM for Linux

Building the Distributed Authentication Web Application

The JES 4 Installation Utility installs the distributed authentication package in the *AccessManager-base*/SUNWcomm/SUNWam, however in order for distributed authentication to function properly, the packages must be located in *AccessManager-base*/SUNWam. Copy the following files to the *AccessManager-base*/SUNWam directory and proceed with the steps described in this document:

README.distAuthUI	Contains installation and package instructions.
amauthistui.war	The distributed authentication Web application.
Makefile.distAuthUI	The Makefile.distAuthUIis used to generate and build required web application. The makefile defines targets to build configuration properties and web application and jars

▼ To Build the Distributed Authentication Web Application

1 Edit the Makefile.distAuthUI to provide the following required parameters:

- JAVA_HOME
- SERVER_PROTOCOL
- SERVER_HOSTNAME
- SERVER_PORT
- SERVER_DEPLOY_URI
- APPLICATION_USERNAME
- APPLICATION_PASSWORD
- DISTAUTH_DEPLOY_URI
- DISTAUTH_PROTOCOL
- DISTAUTH_HOSTNAME
- DISTAUTH_PORT

2 Enter the following command to run the makefile:

```
make -f Makefile.distAuthUI
```

This command generates a deployable war file (amauthdistui_deploy.war) that can be deployed in any Servlet 2.3 compliant container. The targets defined in the Makefile.distAuthUI are:

properties	Generates AMConfig.properties in the temporary directory. This file is used as a template for setting Access Manager Distributed Authentication Web Application's properties
webapp	Generates amauthdistui_deploy.war that can be deployed on any Servlet 2.3–compliant web container

Initializing the Distributed Authentication Web Application

In order for the Access Manager Client SDK to communicate with Access Manager Server, you must initialize several properties. These properties can be set in one of the following methods:

1. Through the properties file – Set the properties in a file and provide a path to it at runtime using the –`Damconfig=`*filename* command. The properties files should be in the CLASSPATH. The default properties file name is `AMConfig.properties` and is always read at start-up. A sample `AMConfig.properties` can be generated using the `make -f Makefile.distAuthUI properties` command. The `AMConfig.properties` will be present in the `/temp` directory.
2. Through the Java API —

   ```
   com.iplanet.am.util.SystemProperties.initializeProperties
   ```

   (where the `java.util.Properties` file contains the properties).
3. Individual properties can be set at runtime using the –`D` flag. For example, `-D` *propertyName=propertyValue*.

The properties expected by Distributed Authentication web application are:

Naming URL property	`com.iplanet.am.naming.url` and `com.iplanet.am.naming.failover.url`. This is a mandatory property and it specifies the Access Manager Server's Naming URL. For example:
	`com.iplanet.am.naming.url=`
	`http://is.example.com/amserver/namingservice`

Debug Level and Directory	The `com.iplanet.services.debug.level` and `com.iplanet.services.debug.directoryproperties` specify the debug level and directory. The possible values for debug levels are off, error, warning, and message.
Notification URL property:	The web application can register for changes to server attributes. The `com.iplanet.am.notification.url` property must be set to receive such notifications.

Security Properties

Some of the Access Manager components such as Service Management, User Management, and so forth, require an identity for the client (application) to read configuration data and to identify the client. The identity for the client can be set up by providing either a username and password that can be authenticated, or by providing an implementation for the interface `com.sun.identity.security.AppSSOTokenProvider` that returns a single-sign-on (SSO) token.

1. The properties to set the username, password and shared secret are:

   ```
   com.sun.identity.agents.app.username
   ```

`com.iplanet.am.service.password`

`com.iplanet.am.service.secret`

2. The property to set the SSO Token provider:

   `com.sun.identity.security.AdminToken`

3. Some of the configuration attributes (such as password) are encrypted and stored in the data store. If such attributes have to be decrypted by the client, the following property must be set, and must be the same as that of the Access Manager Server:

   `am.encryption.pwd`

Accessing the Distributed Authentication User Interface

Once you have the distributed authentication deployed and configured, you can access it by entering the following access URL syntax into your browser:

`http://`*dist_auth_server_host*`.`*dist_auth_server_domain*`:`
dist_auth_server_port`/`*DISTAUTH_DEPLOY_URI*`/UI/Login`

This URL should always be an absolute URL, ideally would be used as `goto=`*absolute successful redirect URL* query parameter with the above Access URL since this web application would be normally deployed in a DMZ layer in production.

For testing purpose, if you happen to use the server's returned default successful redirect URL (the server's Administration Console URL), make sure that you change the URL from its relative value (*SERVER_DEPLOY_URI*`/console`) to the absolute value in the authentication properties in the Administration Console. For example:

SERVER_PROTOCOL`://`*SERVER_HOSTNAME*`:`
SERVER_PORT`/`*SERVER_DEPLOY_URI*`/console`

Accessing Sun Resources Online

The docs.sun.com^{SM} web site enables you to access Sun technical documentation online. You can browse the docs.sun.com archive or search for a specific book title or subject. Books are available as online files in PDF and HTML formats. Both formats are readable by assistive technologies for users with disabilities.

To access the following Sun resources, go to `http://www.sun.com`:

- Downloads of Sun products
- Services and solutions
- Support (including patches and updates)
- Training
- Research

- Communities (for example, Sun Developer Network)

Third-Party Web Site References

Third-party URLs are referenced in this document and provide additional, related information.

Note – 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 or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. To share your comments, go to `http://docs.sun.com` and click Send Comments. In the online form, provide the full document title and part number. The part number is a 7-digit or 9-digit number that can be found on the book's title page or in the document's URL. For example, the part number of this book is 819-4566-11.