Sun Java System Access Manager 7.1 Technical Overview

Beta

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

Part No: 819–4669–10
March 2007

070227@16599

Contents

Preface

Sun Java™ System Access Manager is a component of the Sun Java Enterprise System (Java ES), a set of software components that provide services needed to support enterprise applications distributed across a network or Internet environment. The *Sun Java System Access Manager 7.1 Technical Overview* describes Access Manager features, explains what Access Manager does, and illustrates how Access Manager works.

Before You Read This Book

This book is intended for use by IT administrators and software developers who implement a web access platform using Sun Java System servers and software. Readers of this guide should be familiar with the following:

- Web containers in which Access Manager can be deployed:
  - Sun Java System Application Server
  - Sun Java System Web Server
  - BEA WebLogic
  - IBM WebSphere Application Server
- Technologies:
  - Lightweight Directory Access Protocol (LDAP)
  - Java
  - JavaServer Pages™ (JSP)
  - HyperText Transfer Protocol (HTTP)
  - HyperText Markup Language (HTML)
  - eXtensible Markup Language (XML)
  - SOAP
  - HyperText Transfer Protocol (HTTP)
  - Liberty Alliance Project specifications

Related Books

Related documentation is available as follows:

- "Access Manager Installation Instructions" on page 8
- "Access Manager Core Documentation" on page 8
- "Sun Java System Product Documentation" on page 9
- "Sun Java Enterprise System Product Documentation" on page 9

Access Manager Installation Instructions

For detailed information about installing Access Manager, see the following Sun Java Enterprise System documents:

- *Sun Java Enterprise System 5 Release Notes for UNIX*
- *Sun Java Enterprise System 2006Q3 Deployment Planning Guide*
- *Sun Java Enterprise System 5 Installation Guide for UNIX*
- *Sun Java Enterprise System 2006Q3 Upgrade Guide*

Access Manager Core Documentation

The Access Manager core documentation set contains the following titles:

- The *Sun Java System Access Manager 7.1 Release Notes* will be available online after the product is released. It gathers an assortment of last-minute information, including a description of what is new in this current release, known problems and limitations, installation notes, and how to report issues with the software or the documentation.
- The *Sun Java System Access Manager 7 Technical Overview* (this guide) provides an overview of how Access Manager components work together to protect enterprise assets and web-based applications. It also explains basic Access Manager concepts and terminology.
- The *Sun Java System Access Manager 7.1 Deployment Planning Guide* provides planning and deployment solutions for Sun Java System Access Manager based on the solution life cycle
- The *Sun Java System Access Manager 7.1 Postinstallation Guide* provides information for configuring Access Manager after running the Java ES installer.
- The *Sun Java System Access Manager 7.1 Performance Tuning Guide* provides information on how to tune Access Manager and its related components for optimal performance.
- The *Sun Java System Access Manager 7.1 Administration Guide* describes various administrative tasks such as Realms Management, Policy Management, Authentication and Directory Management. Most of the tasks described in this book are performed through the Access Manager console as well as through the command line utilities.

- The *Sun Java System Access Manager 7.1 Administration Reference* is a look-up guide containing information about the command line interfaces, configuration attributes, Access Manager files, and error codes.
- The *Sun Java System Access Manager 7.1 Federation and SAML Administration Guide* provides information about the Federation module based on the Liberty Alliance Project specifications and the use of the Security Assertion Markup Language (SAML). It includes information on the integrated services based on these specifications, instructions for enabling a web services environment, and summaries of the application programming interface (API) for extending the framework.
- The *Sun Java System Access Manager 7.1 Developer's Guide* offers information on how to customize Access Manager and integrate its functionality into an organization's current technical infrastructure. It also contains details about the programmatic aspects of the product and its API.
- The *Sun Java System Access Manager 7.1 C API Reference* provides summaries of data types, structures, and functions that make up the public Access Manager C APIs.
- The *Sun Java System Access Manager 7.1 Java API Reference* provides information about the implementation of Java packages in Access Manager.
- The *Sun Java System Access Manager Policy Agent 2.2 User's Guide* provides an overview of the policy functionality and the policy agents available for Access Manager.

Updates to the *Release Notes* and links to modifications of the core documentation can be found on the Access Manager page at the Sun Java Enterprise System documentation web site. Updated documents will be marked with a revision date.

Sun Java System Product Documentation

Useful information can be found in the documentation for the following products:

- Sun Java System Directory Server Enterprise Edition 6.0
- Sun Java System Web Server 7.0
- Sun Java System Application Server Enterprise Edition 8.2
- Sun Java System Web Proxy Server 4.0.4

Sun Java Enterprise System Product Documentation

A full list of the Java Enterprise System documentation is documented in the following table.

TABLE P–1 Sun Java Enterprise System Documentation Listing

Document Title	Contents
Sun Java Enterprise System 5 Release Notes for UNIX *Sun Java Enterprise System 5 Release Notes for Microsoft Windows*	Contains the latest information about Java ES, including known problems. In addition, components have their own release notes listed in the Release Notes Collection.
Sun Java Enterprise System 5 Technical Overview	Introduces the technical and conceptual foundations of Java ES. Describes components, the architecture, processes, and features.
Sun Java Enterprise System 2006Q3 Deployment Planning Guide	Provides an introduction to planning and designing enterprise deployment solutions based on Java ES. Presents basic concepts and principles of deployment planning and design, discusses the solution life cycle, and provides high-level examples and strategies to use when planning solutions based on Java ES.
Sun Java Enterprise System 5 Installation Planning Guide	Helps you develop the implementation specifications for the hardware, operating system, and network aspects of your Java ES deployment. Describes issues such as component dependencies to address in your installation and configuration plan.
Sun Java Enterprise System 5 Installation Guide for UNIX *Sun Java Enterprise System 5 Installation Guide for Microsoft Windows*	Guides you through the process of installing Java ES. Also shows how to configure components after installation, and verify that they function properly.
Sun Java Enterprise System 5 Installation Reference for UNIX	Gives additional information about configuration parameters, provides worksheets to use in your configuration planning, and lists reference material such as default directories and port numbers on the Solaris Operating System and Linux operating environment.
Sun Java Enterprise System 2006Q3 Upgrade Guide *Sun Java Enterprise System 5 Upgrade Guide for Microsoft Windows*	Provides instructions for upgrading to Java ES 5 from previously installed versions.
Sun Java Enterprise System 5 Monitoring Guide	Gives instructions for setting up the Monitoring Framework for each product component and using the Monitoring Console to view real-time data and create monitoring rules.
Sun Java Enterprise System Glossary	Defines terms that are used in Java ES documentation.

Searching Sun Product Documentation

Besides searching Sun product documentation from the docs.sun.com^{SM} web site, you can use a search engine by typing the following syntax in the search field:

```
search-term site:docs.sun.com
```

For example, to search for “broker,” type the following:

```
broker site:docs.sun.com
```

To include other Sun web sites in your search (for example, java.sun.com, www.sun.com, and developers.sun.com), use `sun.com` in place of `docs.sun.com` in the search field.

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/`)

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 title of this book is *Sun Java System Access Manager 7.1 Technical Overview*, and the part number is 819–4669–10.

Typographic Conventions

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

TABLE P–2 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	A placeholder to be replaced 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 (note that some emphasized items appear bold online)	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file.

Shell Prompts in Command Examples

The following table shows default system prompts and superuser prompts.

TABLE P–3 Shell Prompts

Shell	Prompt
C shell on UNIX and Linux systems	`machine_name%`
C shell superuser on UNIX and Linux systems	`machine_name#`
Bourne shell and Korn shell on UNIX and Linux systems	`$`
Bourne shell and Korn shell superuser on UNIX and Linux systems	`#`
Microsoft Windows command line	`C:\`

Symbol Conventions

The following table explains symbols that might be used in this book.

TABLE P–4 Symbol Conventions

Symbol	Description	Example	Meaning
`[ ]`	Contains optional arguments and command options.	`ls [-l]`	The `-l` option is not required.
`{ | }`	Contains a set of choices for a required command option.	`-d {y|n}`	The `-d` option requires that you use either the `y` argument or the `n` argument.
`${ }`	Indicates a variable reference.	`${com.sun.javaRoot}`	References the value of the `com.sun.javaRoot` variable.
-	Joins simultaneous multiple keystrokes.	Control-A	Press the Control key while you press the A key.
+	Joins consecutive multiple keystrokes.	Ctrl+A+N	Press the Control key, release it, and then press the subsequent keys.
→	Indicates menu item selection in a graphical user interface.	File → New → Templates	From the File menu, choose New. From the New submenu, choose Templates.

CHAPTER 1

Introduction to Access Manager

Sun Java^{TM} System Access Manager 7.1 integrates authentication and authorization services, policy agents, and identity federation to provide a comprehensive solution for protecting network resources. These functions allow an Access Manager deployment to prevent unauthorized access to web service applications and web content. Topics in this introductory chapter include:

An Access Management Paradigm

Think of all the different types of information a company must store and make available throughout its enterprise. Now consider the various users who must make use of that information in order for the company's business to run smoothly. For example, the following are routine information transactions that occur every day in a typical company:

- An employee looks up a colleague's phone number in the corporate phone directory.
- A manager looks up employee salary histories to help determine an individual's merit raise.
- An administrative assistant adds a new hire to the corporate database, triggering the company's health insurance provider to add the new hire to its enrollment.
- An engineer sends an internal URL for a specification document to another engineer who works for a partner company.
- A customer logs into the company's web site and looks for a product in the company's online catalog.
- A vendor submits an online invoice to the company's accounting department.

In each of these examples, the company must determine who is allowed to view its information or use its applications. Some information such as the company's product descriptions and advertising can be made available to everyone, even the public at large, in the company's online catalog. Other information such as accounting and human resources information must be restricted to employees only. And some internal information is appropriate to share with partners and suppliers, but not with customers.

The Problem

Many enterprises grant access to information on a per-application basis. For example, an employee might have to set up a user name and password to access the company's health benefits administration web site. The same employee must use a different user name and password to access the Accounting Department online forms. Within the same enterprise, a customer sets up a user name and password to access the public branch of the company web site. For each web site or service, an administrator must convert the enterprise user's input into a data format that the service can recognize. Each service added to the enterprise must be provisioned and maintained separately.

The Solution

Sun Java System Access Manager reduces the administrative costs and eliminates the redundant user information associated with per-application solutions. Access Manager enables an administrator to assign specific rules or *policies* that govern the information or services each user can access. Policy agents are deployed on application or web servers to process HTTP requests and to enforce these policies. Together, these access policies and the associated user's information comprise the user's enterprise identity. Thus, Access Manager makes it possible for a user to access many resources in the enterprise with just one identity.

What Access Manager Does

When an enterprise user or an external application tries to access content stored on a company's server, an Access Manager policy agent intercepts the request and directs it to the Access Manager server. Access Manager then asks the user to present credentials such as a username and password. If the credentials match those stored in the appropriate identity repository, Access Manager determines that the user's credentials are authentic.

Following authentication, the Access Manager policy agent evaluates the policies associated with the user's identity to determine authorization to access the requested content. Policies are created using Access Manager and identify which users (or groups of users) are allowed to access a particular resource, specifying the conditions under which this authorization is valid. Based upon the policy evaluation results, the policy agent either grants or denies the user access to the information. Figure 1–1 below illustrates one way Access Manager can be configured to act as the gatekeeper to a company's information resources.

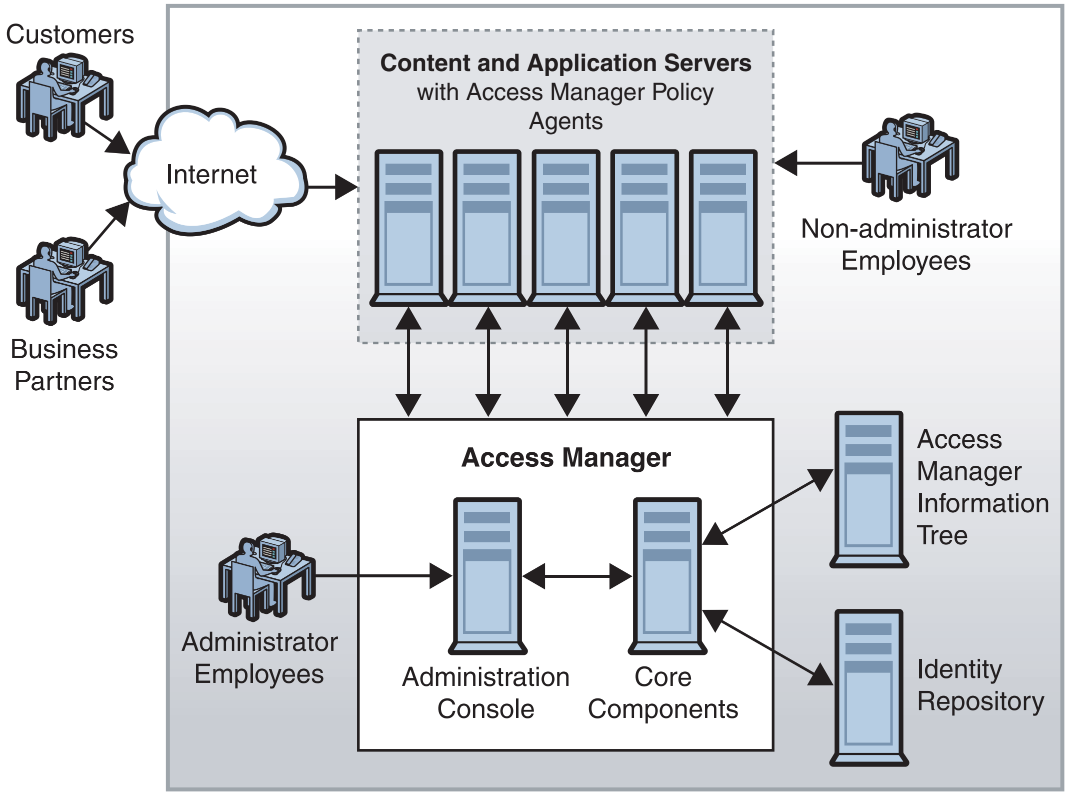


FIGURE 1–1 Access Manager as Gatekeeper to a Company's Enterprise Resources

Access Manager Installation Modes

When you install Access Manager, you are asked to choose either Realm Mode or Legacy Mode. Realm Mode is the new Access Manager architecture; Legacy Mode is based on the Access Manager 6.3 architecture. The following table briefly compares these options. The sections following the table give a more in-depth explanation of each installation modes.

TABLE 1–1 Comparison of Realm and Legacy Modes

	Realm Mode	Legacy Mode
Supports all new Access Manager 7.1 features.	Yes	Yes
Supports identity repositories in Sun Java System Directory Server and other data stores.	Yes	Yes
Supports Access Manager 6 user management features.	No	Yes

TABLE 1–1 Comparison of Realm and Legacy Modes *(Continued)*

	Realm Mode	Legacy Mode
Can coexist with Access Manager 6 2005Q1 in multiple-server installations.	No	Yes
Before installation, identity repository can exist in Sun Java Directory Server .	Yes	Yes
Before installation, identity repository can exist in an LDAP version 3 compliant directory server.	Yes	No

To determine if an already installed instance of Access Manager is running in realm or legacy mode, type the following into the location bar of your web browser:

protocol`://`*FQDN_server*`:`*port*`/amserver/SMSServlet?method=isRealmEnabled`

The server will return *true* if running in realm mode. More information on the installation modes can be found in the following sections:

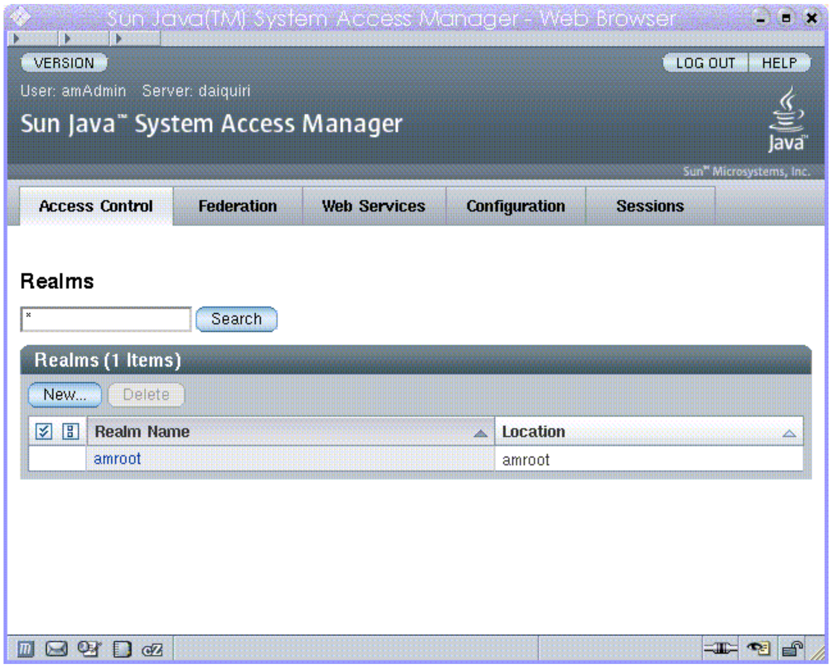
- "Realm Mode" on page 18
- "Legacy Mode" on page 19

Realm Mode

Realm mode is based on the Access Manager information tree and Identity Repository Management Service described in previous sections. Realm Mode is appropriate in most new Access Manager deployments where you want to keep identity repositories independent of access management, or where you cannot maintain user data within the required object classes of Sun Java System Directory Server. If you choose Realm Mode at installation, your identity repositories can exist in any of the following configurations:

- In the same Directory Server instance and the same suffix as the Access Manager information tree.
- In the same Directory Server instance but in a different suffix as the Access Manager information tree.
- In a different directory server instance from the Access Manager information tree.

Figure 1–2 is a screen capture of the Access Manager Administration Console when the product has been installed in Realm Mode.



Realms

Realms (1 Items)

Realm Name	Location
amroot	amroot

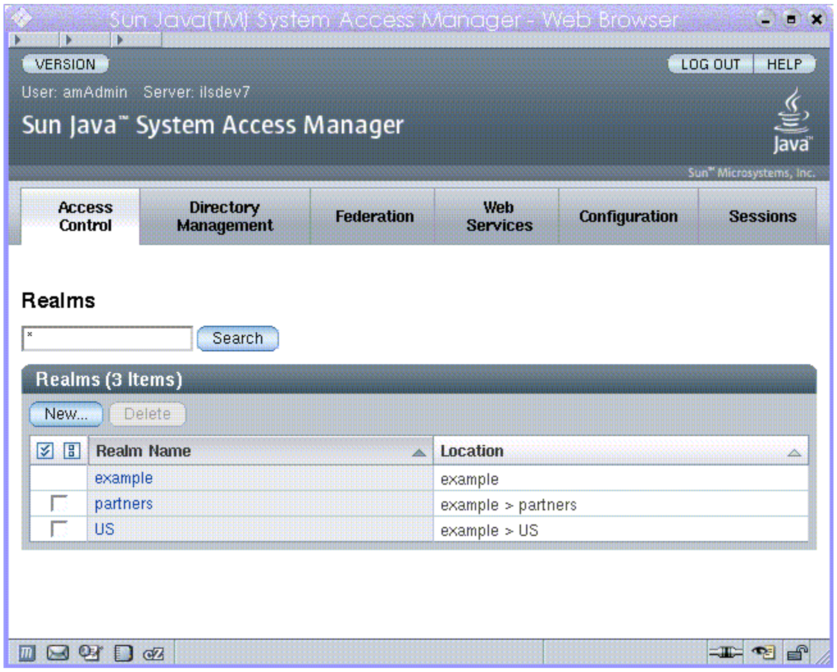
FIGURE 1–2 Realm Mode User Interface

Legacy Mode

Legacy Mode is based on the Access Manager 6.3 architecture. This legacy Access Manager architecture uses the Lightweight Directory Access Protocol (LDAP) directory information tree (DIT) that comes with Sun Java System Directory Server. In Legacy Mode, both user information and access control information are stored in LDAP organizations. When you choose Legacy Mode, an LDAP organization is the equivalent of an access control realm. Realm information is integrated within LDAP organizations.

Legacy Mode is appropriate in deployments where you want to use Access Manager user management. It is typically used in deployments where Access Manager is built upon Sun Java System Portal Server or other Sun Java System communication products that require the use of Sun Java System Directory Server as the central identity repository. If you choose Legacy Mode during installation, the top-level ream resides in the same Directory Server branch as the Access Manager information tree, and user information is intermingled with access information.

Figure 1–3 is a screen capture of the Access Manager Administration Console when the product has been installed in Legacy Mode.



Realms

Realms (3 Items)

Realm Name	Location
example	example
partners	example > partners
US	example > US

FIGURE 1–3 Legacy Mode User Interface

Access Manager Architecture

Access Manager uses a Java technology-based architecture for scalability, performance, and ease of development. It also leverages industry standards including the HyperText Transfer Protocol (HTTP), the eXtensible Markup Language (XML), the Security Assertion Markup Language (SAML), and SOAP. The internal architecture is multi-layered and includes the following:

- A presentation layer
- Web services
- Core components
- Client application programming interfaces (APIs)
- Service provider interfaces (SPIs)
- An integrated framework
- A plug-ins layer

Custom applications access the Access Manager web services through the Access Manager client APIs installed on each protected resource. Custom plug-in modules interact with both the Access Manager SPIs and the plug-ins layer. The plug-in modules retrieve required information from the Access Manager information tree and deliver it to other plug-ins when necessary, and to the Access Manager framework for data processing. Additional information can be found in the following sections.

- "Access Manager Framework" on page 21
- "Access Manager Information Tree" on page 23
- "Realms" on page 25
- "Identity Repository Framework" on page 28
- "Core Components and Internal Services" on page 32
- "Web Services" on page 33
- "SPIs and Plug-ins" on page 34
- "Client APIs" on page 35
- "Access Manager Policy Agents" on page 36

Access Manager Framework

The Access Manager framework is where the Access Manager business logic is implemented. Each core component uses its own framework to retrieve customer data from the plug-in layer and to provide data to the core components. The Access Manager framework integrates all of these frameworks to form one layer in the architecture that is accessible to all core components and Access Manager plug-ins. Figure 1–4 illustrates the plug-ins layer, Access Manager framework, core components, and web services that form the Access Manager architecture.

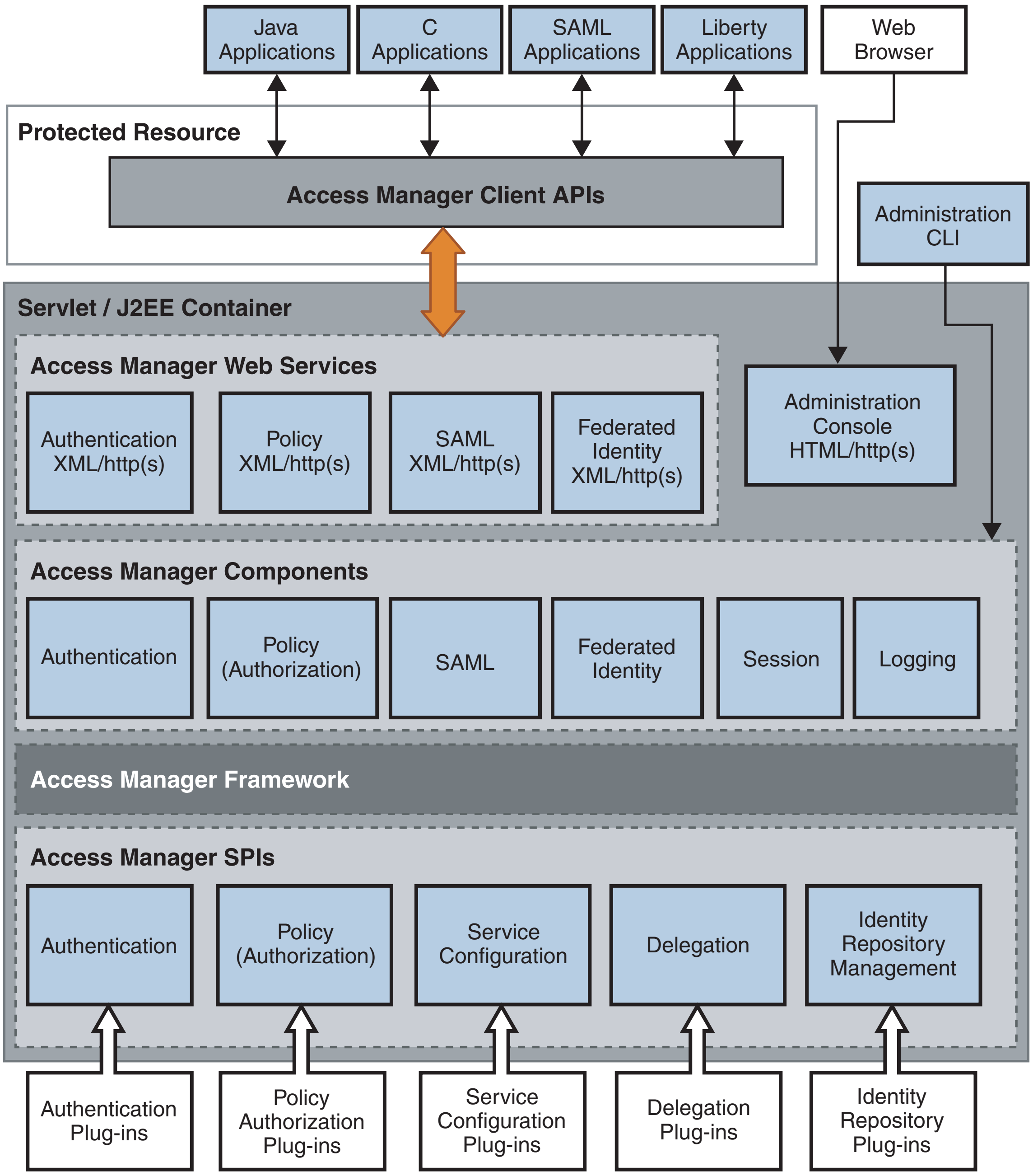


FIGURE 1–4 Access Manager Internal Architecture

Access Manager Information Tree

When installed in Realm Mode, Access Manager creates a special and proprietary branch in an LDAP data store for storing realm configurations, authentication properties, and authorization policies. Access Manager components and plug-ins access the data stored in the Access Manager information tree, and use it for various purposes including the following examples:

- Policy runtime accesses policy data for policy evaluation.
- Identity Repository plug-in finds configuration information for data stores.
- Authentication Service finds authentication configuration information.

By default, the Access Manager information tree is created and maintained by Access Manager as a special branch in Sun Java System Directory Server, apart from any user data (identity repository). Figure 1–5 illustrates this default configuration.

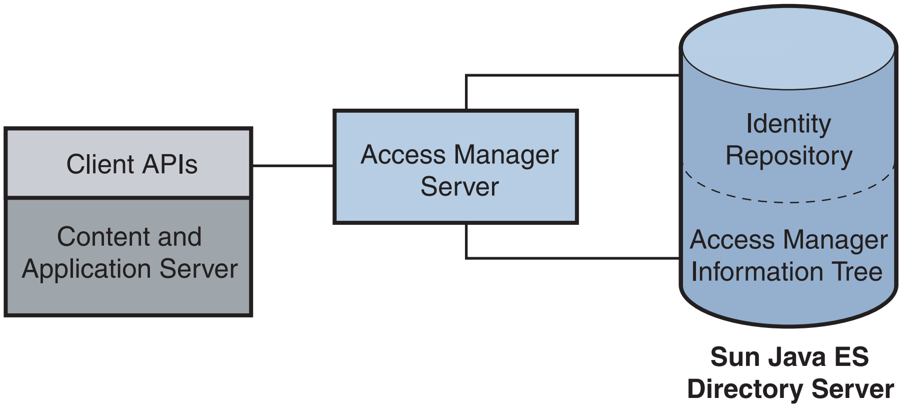

Client APIs
Content and Application Server
Access Manager Server
Identity Repository
Access Manager Information Tree
Sun Java ES Directory Server

FIGURE 1–5 Default Configuration for Access Manager Information Tree

But, the Access Manager information tree can also be created in a directory that is separate from the one hosting the Access Manager Identity Repository. Figure 1–6 illustrates this custom configuration.

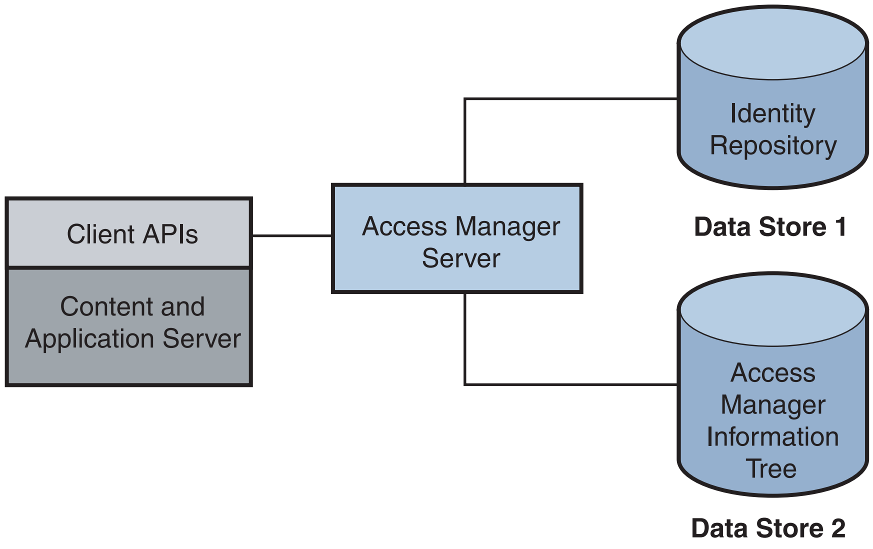


FIGURE 1–6 Access Manager Information Tree Configured in Second Data Store

The following figure compares two directory information trees: the first illustration represents a default hierarchical LDAP structure while the second represents the structure when the Access Manager information tree is integrated.

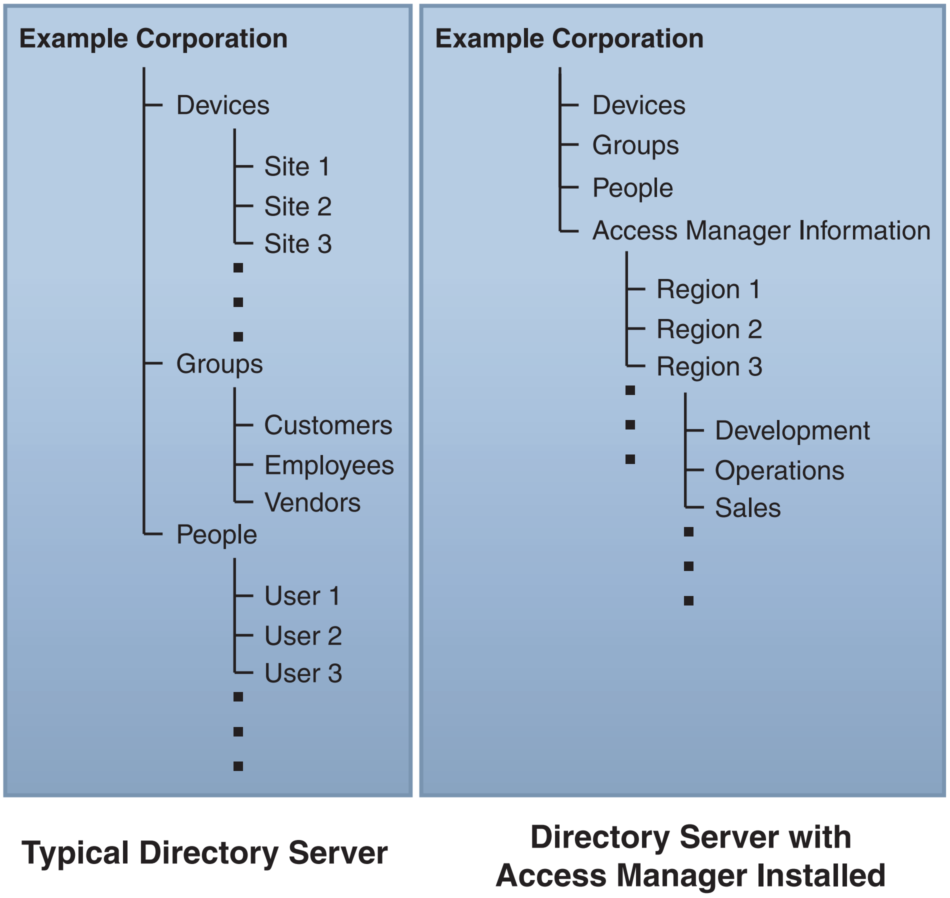


FIGURE 1–7 Directory Server With and Without an Access Manager Information Tree

Realms

An Access Manager *realm* is a grouping of configuration information that you can associate with a user, a group of users, or a collection of protected resources. The configuration information can include, but is not limited to, the following:

- A definition of one or more identity repositories, identifying a set of users, groups, and roles to whom the remaining realm configuration information applies.
- An authentication configuration, identifying, for example, the location of the authentication repository, and the type of authentication required.
- Policy information that will be used to determine which resources protected by Access Manager the subjects can access.

- Responder information that allows applications to personalize the user experience, once the user has successfully authenticated and been given access.

The Access Manager framework aggregates realm properties within the proprietary Access Manager information tree. Figure 4-1 illustrates how realm data stored in an Access Manager information tree can be grouped by region and by company functions.

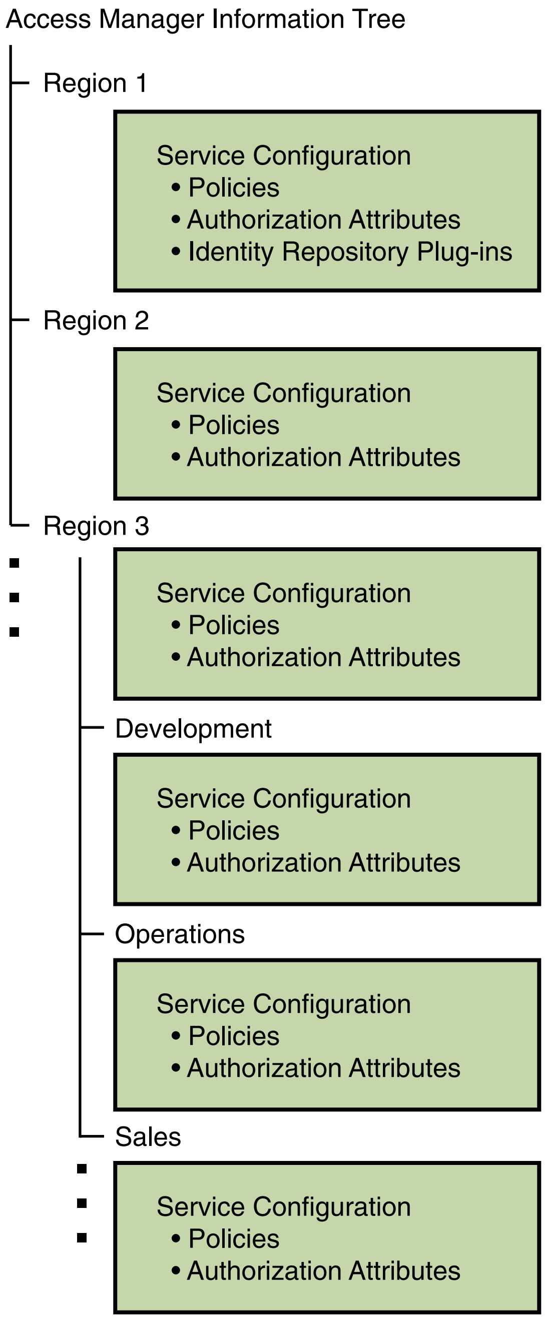
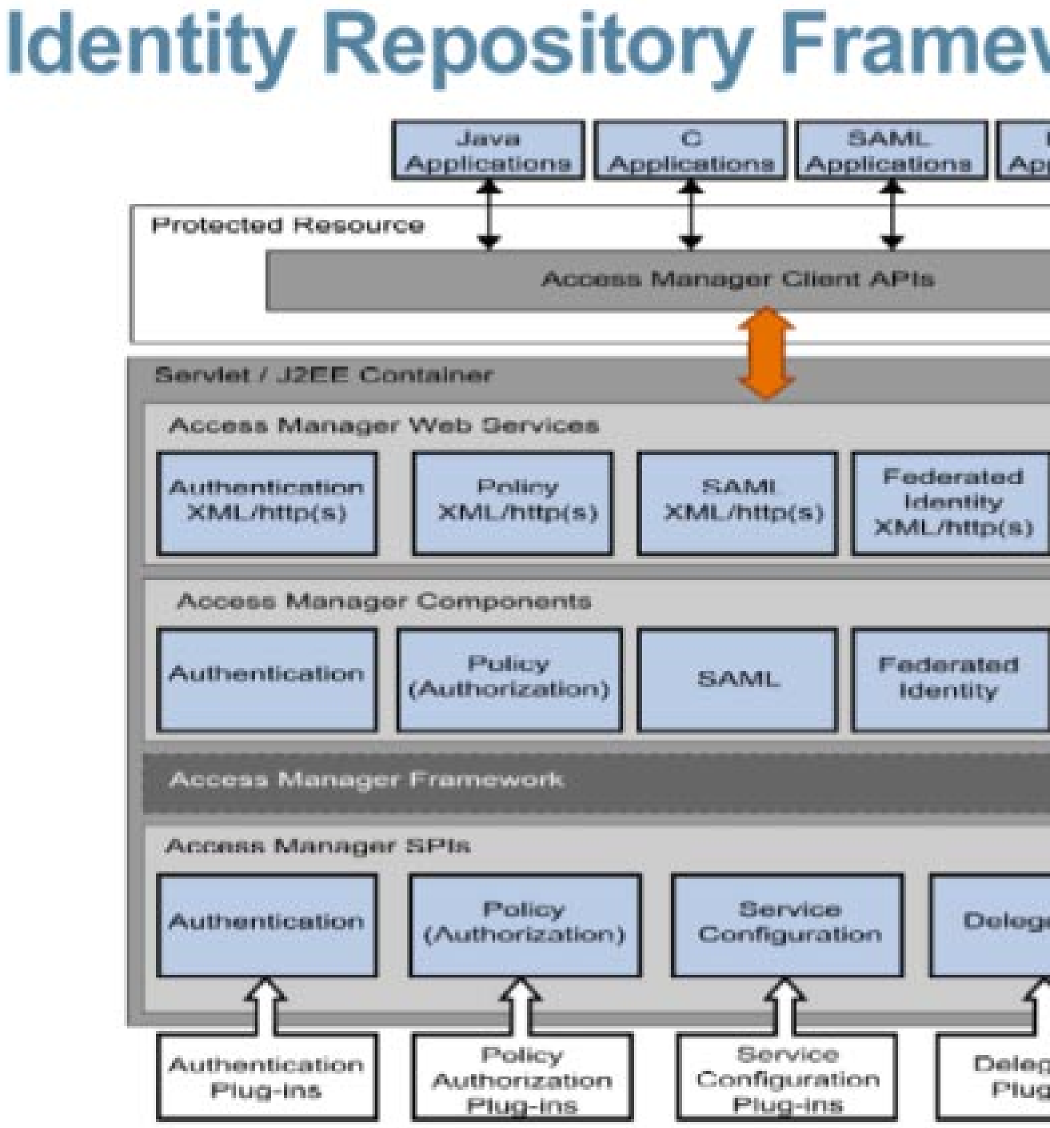


FIGURE 1–8 Realm Data in Access Manager Information Tree

Identity Repository Framework

An *identity repository* is a data store where information about users and groups in a company or organization is stored. The Access Manager Identity Repository Framework and related APIs are a model by which plug-ins can be written that allow communication with different types of identity repositories. Following is an illustration of the Identity Repository Framework and how it is integrated within the other features of Access Manager.



Note – The information in an identity repository is maintained by provisioning products separate from Access Manager. The supported provisioning product is Sun Java System Identity Manager. See Sun Java System Identity Manager for more information.

The Identity Repository Framework is configured as a service within an Access Manager realm. Multiple identity repository plug-ins can be configured for each realm. Each plug-in configuration includes details about what operations are supported on each of the identity types based on the underlying data store. Once an Access Manager realm is configured to use a plug-in, the Identity Repository Framework will instantiate it and execute operations on the identity repository it supports. This model allows the following:

- Data store independence enables you to view and retrieve user information without making changes to your existing user database.
- Universal identities allow you to access the many profiles of one identity across multiple data repositories, if necessary.
- Caching helps to improve repository read performance.

When deploying Access Manager, you must choose one or more of the supported plug-ins based on the data store. You can configure the Identity Repository Service per realm to use its own list of identity repositories to store service configurations for both users and roles. The Access Manager framework integrates data from the identity repository plug-in with data from other Access Manager plug-ins to form a virtual identity for each user. Access Manager can then use this identity in authentication and authorization processes among more than one identity repositories. The virtual user identity is destroyed when the user's session ends.

Each new plug-in developed must have a corresponding service management schema defining its configuration attributes. This schema is enveloped into the service management file for the Identity Repository Service (named `idRepoService.xml`) as a sub schema. Currently, Access Manager provides out-of-the-box plug-in support for the following types of identity repositories:

- "Access Manager Repository Plug-in (Sun Java System Directory Server)" on page 30
- "Generic Lightweight Directory Access Protocol (LDAP) version 3" on page 31
- "Flat Files Repository" on page 31
- "Active Directory" on page 31
- "Sun Directory Server With Access Manager Schema" on page 32

Access Manager Repository Plug-in (Sun Java System Directory Server)

The Access Manager Repository can reside only in Sun Java System Directory Server. During installation, the repository itself is created in the same instance of Sun Java System Directory Server that holds the Access Manager information tree. (This is the default installation mode when using the Sun Java Enterprise System installer.) The two information trees are configured in separate nodes under a single directory suffix. The Access Manager Repository Plug-in is designed to work with Sun Java System Directory Server as it makes use of features specific to the server including *roles* and *class of service*. It uses a DIT structure similar to that of previous versions of Access Manager.

Note – Previously, the functionality of this plug-in was provided by the AMSDK component. In Access Manager 7.1, the AMSDK functionality still exists, but as a plug-in only. (See "AM SDK Plug-in" on page 35.) Thus, the Access Manager Repository is compatible with previous versions of Access Manager.

When you configure an instance of Access Manager in realm mode for the first time, the following occurs:

- An Access Manager Repository is created under the top-level realm.
- The Access Manager Repository is populated with internal Access Manager users.

Note – The Java Enterprise System installer does not set up an Access Manager Repository when you configure an Access Manager instance in legacy mode. Legacy mode requires an identity repository that is mixed with the Access Manager information tree under a single directory suffix.

Active Directory

This data store type uses the LDAP version 3 specification to write identity data to an instance of Microsoft® Active Directory®.

Generic Lightweight Directory Access Protocol (LDAP) version 3

Generic LDAPv3 identity repositories may reside on an instance of any directory that complies with the LDAPv3 specifications. The directory can not make use of features that are not part of the LDAP version 3 specification, and no specific DIT structure can be assumed as LDAPv3 identity repositories are simply DIT branches that contain user and group entries. The identity repositories might or might not reside in the same instance of Sun Java System Directory Server as the Access Manager information tree. Each data store has a name that is unique among a realm's data store names, but not necessarily unique across all realms in the Access Manager information tree. The `com.sun.identity.idm.plugins.ldapv3.LDAPv3Repo` class provides the default LDAPv3 identity repository implementation.

Note – This configuration is not compatible with previous versions of Access Manager.

Flat Files Repository

This repository allows you to store data and identities in a flat DIT structure on the local installation of Access Manager without having to create a separate data store. This is generally used for testing or proof of concept deployments.

Note – If deploying an instance of Access Manager from a single WAR file, the default identity repository is a flat file.

Sun Directory Server With Access Manager Schema

This repository resides in an instance of Sun Java System Directory Server and holds the Access Manager information tree. It differs from the Access Manager Repository Plug-in in that more configuration attributes allow for better customization.

Core Components and Internal Services

The core components provide the logic that performs the main Access Manager functions, working with the services that run within Access Manager. These internal services process data solely for use by Access Manager. The following table lists the core components and internal services with brief descriptions.

TABLE 1–2 Core Components and Internal Services

Core Component or Internal Service	What it Does
Authentication component	Validates user's credentials and verifies that the user is who he claims to be.
Authorization (Policy) component	Evaluates policies to determine whether the user has permission to access the requested resource.
SAML component	Provides a protocol-based alternative to using cookies for performing a SSO session.
Federation component	Enables user to access resources provided by multiple business partners in a SSO session.
User Session Management component	Maintains information about user sessions, and enforces timeout limits. Provides continued proof of identity to enable single sign-on sessions.
Logging Service	Tracks a user's interactions with web applications. Creates log messages to form an audit trail of important events within the system.
Naming Service	Defines URLs for other Access Manager components and internal services, enabling a client to locate them.
Platform Service	Manages configurable attributes used in an Access Manager deployment.

TABLE 1–2 Core Components and Internal Services *(Continued)*

Core Component or Internal Service	What it Does
Client Detection Service	Detects the client type of the browser being used to access the Access Manager application. Client types include HyperText Markup Language (HTML) and Wireless Markup Language (WML), among other protocols.

Web Services

Web services follow a standardized way of integrating Web-based applications using XML, SOAP, and other open standards over an Internet protocol backbone. Web services enable applications from various sources to communicate with each other because they are not tied to any one operating system or programming language. Businesses use web services to communicate with each other and their respective clients without having to know detailed aspects of each other's IT systems. Access Manager provides web services that use XML and SOAP over HTTP. These web services are designed to be centrally provided in an enterprise's network for convenient access by client applications. The following table summarizes the web services provided in Access Manager.

TABLE 1–3 Access Manager Web Services

Web Service Name	Description
Authentication	Verifies that a user really is the person he claims to be.
Policy (Authorization)	Evaluates rules (policies) associated with a user's identity, and determines whether an authenticated user has permission to access a protected resource.
SAML	Enables single sign-on sessions among different business domains. Allows business partners to securely exchange authentication and authorization information over the Internet.
Federation	Enables a user to log in at one service provider's site and move to an affiliated service provider site without having to re-authenticate or re-establish identity.
Session	Maintains information about the user's interaction with various applications the user accesses.

Access Manager uses both XML files and Java interfaces to manage web services and web service configuration data. An Access Manager XML file is based on the structure defined in a Document Type Definition (DTD) file which defines the structure, elements and qualifying attributes needed to form the valid XML document. The DTD files are located in *AccessManager-base*`/SUNWam/dtd`. The main `sms.dtd` file defines the structure for all Access Manager XML service files (located in `/etc/opt/SUNWam/config/xml`).

Caution – Do not modify any of the Access Manager DTD files. The Access Manager APIs and their internal parsing functions are based on the default definitions. Alterations to the DTD files may hinder the operation of Access Manager.

SPIs and Plug-ins

The Access Manager SPIs work with plug-ins to provide customer data to the framework for back-end processing. Some customer data comes from external data base applications such as identity repositories while other customer data comes from the Access Manager plug-ins themselves. You can develop additional custom plug-ins to work with the Access Manager SPIs. For a complete listing of Access Manager SPIs, see the *Sun Java System Access Manager 7.1 Java API Reference*. The following sections contain brief descriptions of the plug-ins installed with Access Manager.

- "Authentication Plug-in" on page 34
- "Delegation Plug-in" on page 34
- "Identity Repository Management Plug-in" on page 35
- "Policy Plug-in" on page 35
- "Service Configuration Plug-in" on page 35
- "AM SDK Plug-in" on page 35

Authentication Plug-in

The Authentication Plug-in accesses user data in a specified identity repository to determine if a user's credentials are valid.

Delegation Plug-in

The Delegation plug-in aggregates policies and roles to determine the scope of a network administrator's authority. The Authentication Service and the Policy Service then use the aggregated data to perform authentication and authorization processes. The Delegation plug-in works together with the Identity Repository Management plug-in (where default administrator roles are defined) to form rules that describe the scope of privileges for each network administrator, and specifies the roles to which these rules apply. The following is a list of roles defined by the Identity Repository Management plug-in, and the default rule the Delegation plug-in applies to each.

TABLE 1–4 Access Manager Administrator Roles and Scope of Privileges

Administrator Role	Delegation Rule
Realm Administrator	Can access all data in all realms of the Access Manager information tree.
Subrealm Administrator	Can access all data within a specific realm of the Access Manager information tree.
Policy Administrator	Can access all policies in all realms of the Access Manager information tree.
Policy Realm Administrator	Can access policies only within the specific realm of the Access Manager information tree.

Note – The Delegation plug-in code is not public in Access Manager.

Identity Repository Management Plug-in

The Identity Repository Management plug-in returns identity information such as user attributes and membership status for purposes of authentication.

Policy Plug-in

The Policy plug-in aggregates policies and rules to determine whether a user is authorized to access a protected resource.

Service Configuration Plug-in

The Service Configuration plug-in stores and manages configuration data required by the core components and other Access Manager plug-ins. In previous versions of Access Manager, the functionality provided by the Service Configuration plug-in was known as the Service Management Service (SMS).

AM SDK Plug-in

The AM SDK plug-in creates and modifies users and stores information in the user branch of the identity repository. It implements the user management APIs used in previous Access Manager releases.

Client APIs

Enterprise resources cannot be protected by Access Manager until the Access Manager client APIs are installed on the Web Server or Application Server that you want to protect. (The client

APIs are automatically installed when you install a policy agent.) The client APIs allow you to customize an application by enabling communication with Access Manager for retrieving user, session, and policy data.

Access Manager Policy Agents

You install an Access Manager Policy Agent on a resource you'd like to protect to enforce the policy decisions determined by the Policy Service. The policy agent intercepts requests from applications, and redirects the requests to Access Manager for authentication. Once the user is authenticated, the policy agent communicates with the Policy Service for authorization. The policy agent allows or denies the user access depending upon the result of policy evaluation. Policy agents are downloaded and installed separately from the Access Manager server. For more information, see *Sun Java System Access Manager Policy Agent 2.2 User's Guide*.

How Access Manager Works

When Access Manager starts up, it initializes the Access Manager information tree with configuration data from various Access Manager service plug-ins including those for Authorization, Policy, Identity Repository Management, and Service Configuration. When a browser sends an HTTP request for access to a protected resource, Access Manager immediately binds to the appropriate Identity Repository to obtain user information (which may include definitions for roles, realms, user IDs, and so forth). At the same time, a policy agent installed on the protected resource intercepts the request and examines it. If no session token is found, the policy agent contacts the Access Manager server which will then invoke the authentication and authorization processes. Figure 1–9 illustrates how policy agents protect the enterprise's web servers by directing HTTP requests to a centralized Access Manager server for processing.

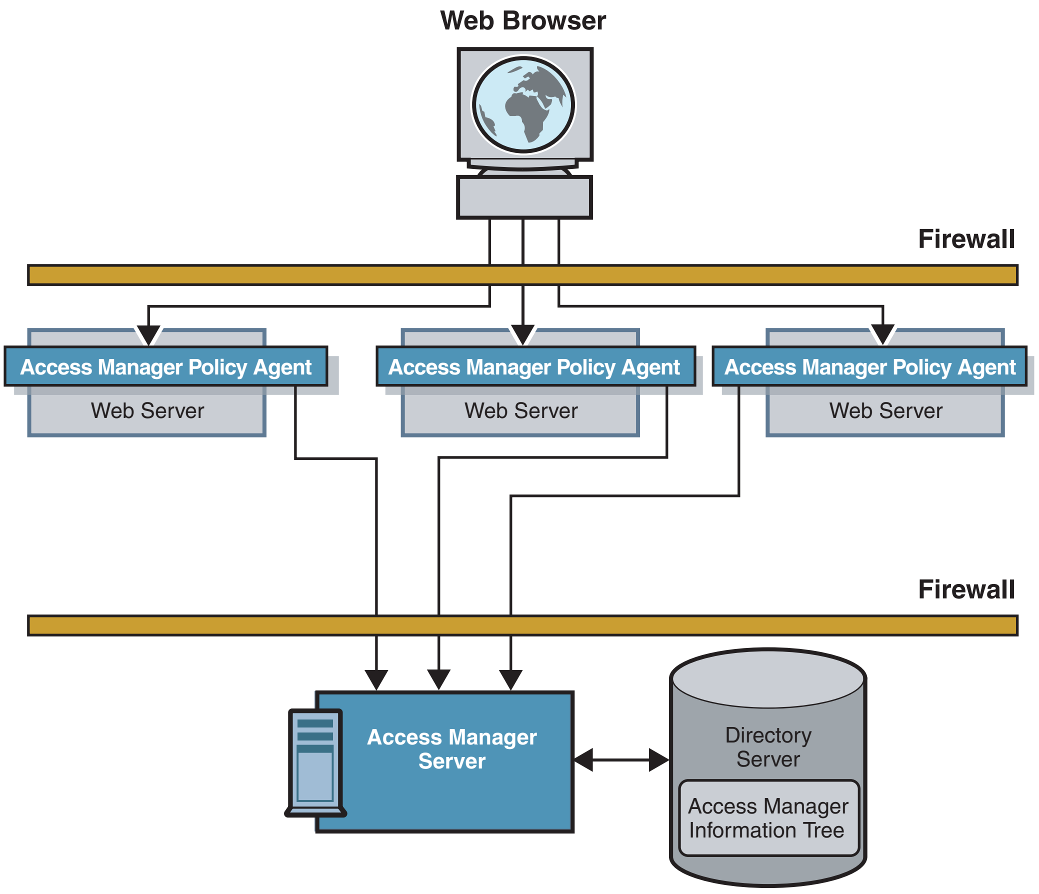

Web Browser
Firewall
Access Manager Policy Agent
Web Server
Access Manager Policy Agent
Web Server
Access Manager Policy Agent
Web Server
Firewall
Access Manager Server
Directory Server
Access Manager Information Tree

FIGURE 1–9 Basic Access Manager Deployment

Access Manager integrates the following functions into one product. They can be viewed and configured using a single administration console:

- "Authentication Service" on page 38
- "Policy Service" on page 38
- "User Session Management" on page 38
- "SAML Service" on page 39
- "Federation Service" on page 39
- "Logging" on page 39

Authentication Service

Authentication is the first step in determining whether a user is allowed to access a resource protected by Access Manager. The Access Manager Authentication Service verifies that a user really is the person he claims to be. It consists of the following components:

- Plug-in modules
- A framework for connecting plug-in modules
- A core authentication component
- A graphical user interface
- Client APIs

The Authentication Service interacts with the Authentication database to validate user credentials, and with Identity Repository Management plug-ins to retrieve user profile attributes. When the Authentication Service determines that a user's credentials are genuine, a valid user session token is issued, and the user is said to be *authenticated*.

Policy Service

Authorization is the process with which Access Manager evaluates policies associated with a user's identity, and determines whether an authenticated user has permission to access a protected resource. The Access Manager Policy Service enables authorization to take place. It consists of the following components:

- Policy plug-ins
- A framework for connecting policy plug-ins
- A core policy component
- A graphical user interface
- Client APIs

The Policy Service interacts with Access Manager service configurations, a delegation plug-in (which helps to determine a network administrator's scope of privileges), and identity repository plug-ins to verify that the user has access privileges from a recognized authority.

User Session Management

An Access Manager user session is the interval between the moment a user logs in to a network resource protected by Access Manager, and the moment the user logs out of the resource. During the user session, the Access Manager Session Service maintains information about the interactions the user has with the various applications. Access Manager uses this information to enforce time-dependent rules such as timeout limits. Also during the user session, Access Manager provides continuous proof of the user's identity. This proof of identity enables the user to access multiple enterprise resources without having to provide credentials each time.

The Access Manager Session Service enables the following types of user sessions:

- **Basic user session.** The user provides credentials to log in to one application, and then logs out of the same application.
- **Single sign-on (SSO) session.** The user provides credentials once, and can then access multiple applications within the same DNS domain.
- **Cross domain SSO (CDSSO) session.** The user provides credentials once, and can then access applications among multiple DNS domains.

SAML Service

Access Manager uses the Security Assertion Markup Language (SAML), an XML-based framework for exchanging security information. While the Session Service enables SSO sessions among different DNS domains within the same intranet, the SAML Service enables CDSSO sessions among different business domains. Using the SAML protocol, business partners can securely exchange authentication and authorization information over the Internet. The SAML Service consists of the following components:

- A framework to which web services can connect
- A core SAML component
- A graphical user interface

Federation Service

Identity federation allows a user to consolidate the many local identities he has configured among multiple service providers. With one federated identity, the user can log in at one service provider's site and move to an affiliated service provider site without having to re-authenticate or re-establish identity. The Federation Service uses SAML to enable SSO sessions among business partners over the Internet. It consists of the following components:

- A framework that complies with the Liberty Alliance Project specifications
- A core Federation component
- A graphical user interface

Logging

When a user logs in to a resource protected by Access Manager, the Logging component records information about the user's activity. You can write custom log operations and customize log plug-ins to generate log reports for auditing purposes.

◆ ◆ ◆ CHAPTER 2

User Session Management and Single Sign-On

This chapter explains how the Access Manager Session Service works with other core Access Manager components to process HTTP requests and to manage user session data. The chapter traces events in a basic user session, a single sign-on session (SSO), and a cross-domain single sign-on session (CDSSO) to give you an overview of Access Manager's features and process flows. Topics covered include:

User Sessions and the Session Service

The Session Service in Sun Java System Access Manager tracks a user's interaction with web applications. For example, the Session Service maintains information about how long a user has been logged in to Access Manager, and enforces time-out limits when necessary. Additionally, the Session Service performs the following actions:

- Generates session identifiers.
- Maintains a master copy of session state information.
- Implements time-dependent behavior of sessions.
- Implements session life cycle events such as logout and session destruction.
- Generates session life cycle event notifications.
- Generates session property change notifications.
- Implements session quota constraints.
- Implements session failover.

- Enables single sign-on (SSO) and cross-domain single sign-on (CDSSO) among applications external to Access Manager.

A *user session* is the interval between the moment a user logs in to Access Manager, and the moment the user logs out of Access Manager. In a typical user session, an employee attempts to access the corporate benefits administration application. The application is protected by Access Manager, and Access Manager prompts the user for a username and password. First, Access Manager *authenticates*, or verifies that the user is who he says he is. Following user authentication, Access Manager allows the user access to the application (providing the user has the appropriate permissions). For a more detailed explanation, see “Basic User Session” on page 44.

Oftentimes, in the same user session (without logging out of the corporate benefits application), the same employee attempts to access the corporate expense reporting application. Because the expense reporting application is also protected by Access Manager, the Session Service provides continued proof of the user’s authentication, and the employee is automatically allowed to access the expense reporting application. The employee has accessed more than one application in a single user session without having to re-authenticate. This functionality is called *Single Sign-On* (SSO). When SSO occurs among applications in more than one DNS domain, the functionality is called *Cross-Domain Single Sign-On* (CDSSO). For more detailed explanations, see “Single Sign-On Session” on page 53 and “Cross-Domain Single Sign-On Session” on page 56, respectively.

Sessions, Session Tokens, and Cookies

The Access Manager Session Service creates a session data structure to store information about a user session and uses cookies to store a token that identifies the session data structure. When a user logs in and is successfully authenticated, the user is assigned a *session*, a session data structure that, minimally, stores the following information about a user session:

Maximum Idle Time	Maximum number of minutes without activity before the session will expire and the user must reauthenticate.
Maximum Session Time	Maximum number of minutes (activity or no activity) before the session expires and the user must reauthenticate.
Maximum Caching Time	Maximum number of minutes before the client contacts Access Manager to refresh cached session information.

Internally, these session attributes are used to enforce Access Manager timeout limits. But a session can also contain additional attributes and properties which can be used by other applications. For example, a session data structure can store information about a user’s identity, or about a user’s browser preferences. You can configure Access Manager to include the following types of information in a session:

- Fixed session attributes

- Protected properties
- Custom properties

For a detailed summary of information that can be included in a session, see Configuring Access Manager Sessions in the *Sun Java System Access Manager 7.1 Postinstallation Guide*.

The Session Service also generates a session token for the new session data structure. The *session token*, also known as a *sessionID*, is an encrypted, unique string that identifies the specific session instance. If the session token is known to a protected resource such as an application, the application can access the session and all user information contained in it. In Access Manager, a session token is carried in a cookie. A *cookie* is an information packet generated by a web server and passed to a web browser. The fact that a web server generates a cookie for a user does not guarantee that the user is allowed access to protected resources. The cookie simply points to user information in a data store from which an access decision can be derived.

Note – Cookies are domain-specific. For example, a cookie generated by a web server within Domain A cannot be used by a web server in Domain B. Cookies can be passed only between servers in the same domain in which the cookie was set. Similarly, servers can set cookies only on servers within in their own domain.

Policy Agents

Policy agents are an integral part of SSO and CDSSO sessions. They are programs that police the web server or application server that hosts protected resources. When a user requests access to a protected resource such as a server or an application, the policy agent intercepts the request and redirects it to the Access Manager Authentication Service for authentication. Following this, the policy agent will also enforce the authenticated user's assigned policies. (A *policy* defines the rules that specify a user's access privileges to a protected resource.) Access Manager supports two types of policy agents:

- The *web agent* enforces URL-based policy for C applications.
- The *J2EE/Java agent* enforces URL-based policy and J2EE-based policy for Java applications on J2EE containers.

Both types of agents are available for you to install as programs separate from Access Manager. For an overview of the available policy agents and links to specific information on installation, see the *Sun Java System Access Manager Policy Agent 2.2 User's Guide*.

Note – When Access Manager policy agents are implemented, all HTTP requests are implicitly denied unless explicitly allowed by the presence of two things:

1. A valid session
2. A policy allowing access

You can modify this default configuration so that Access Manger implicitly allows access unless explicitly denied.

Basic User Session

The following sections describe the process behind a basic user session by tracing what happens when a user logs in to a resource protected by Access Manager. In these examples, the server which hosts an application is protected by an Access Manager policy agent. The Basic User Session includes the following phases:

- "Initial HTTP Request" on page 44
- "User Authentication" on page 46
- "Session Validation" on page 48
- "Policy Evaluation and Enforcement" on page 50
- "Logging Results" on page 52

Initial HTTP Request

When a user initiates a user session by using a browser to log in to a web-based application, the events in the following illustration occur. The accompanying text describes the process.

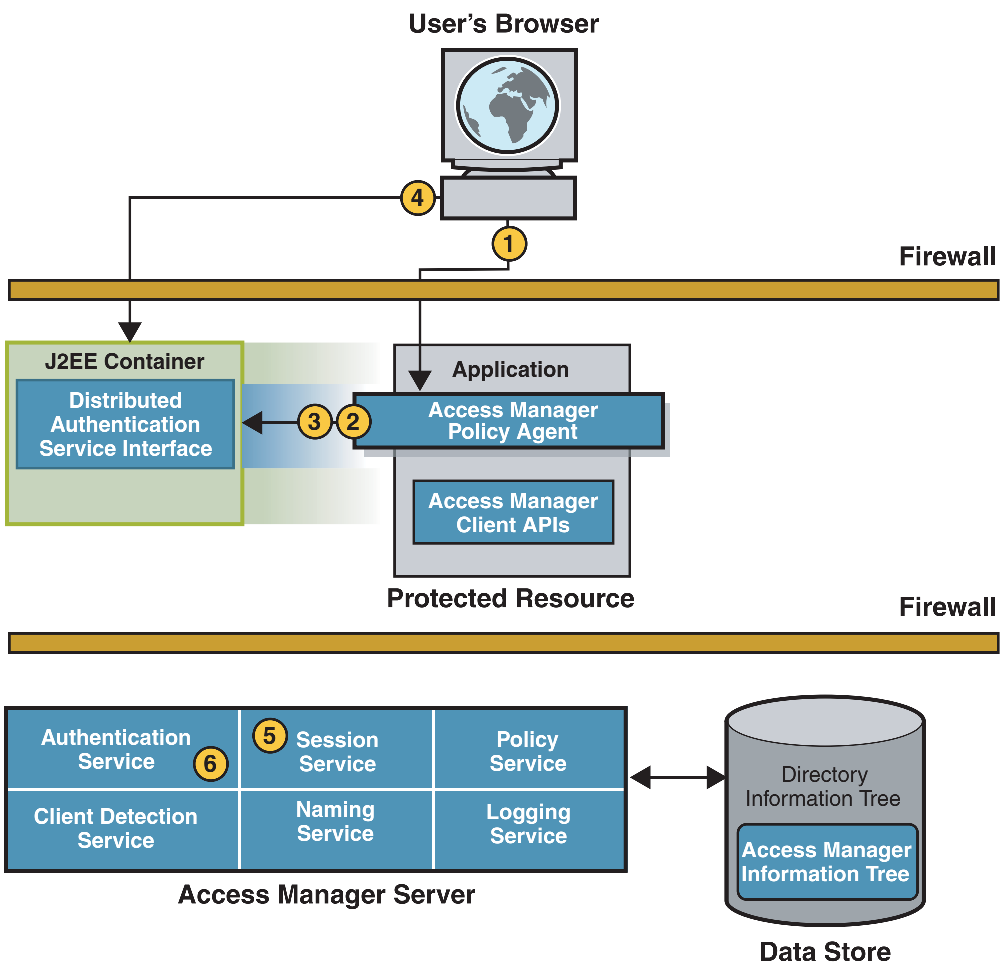


FIGURE 2–1 Initial HTTP Request

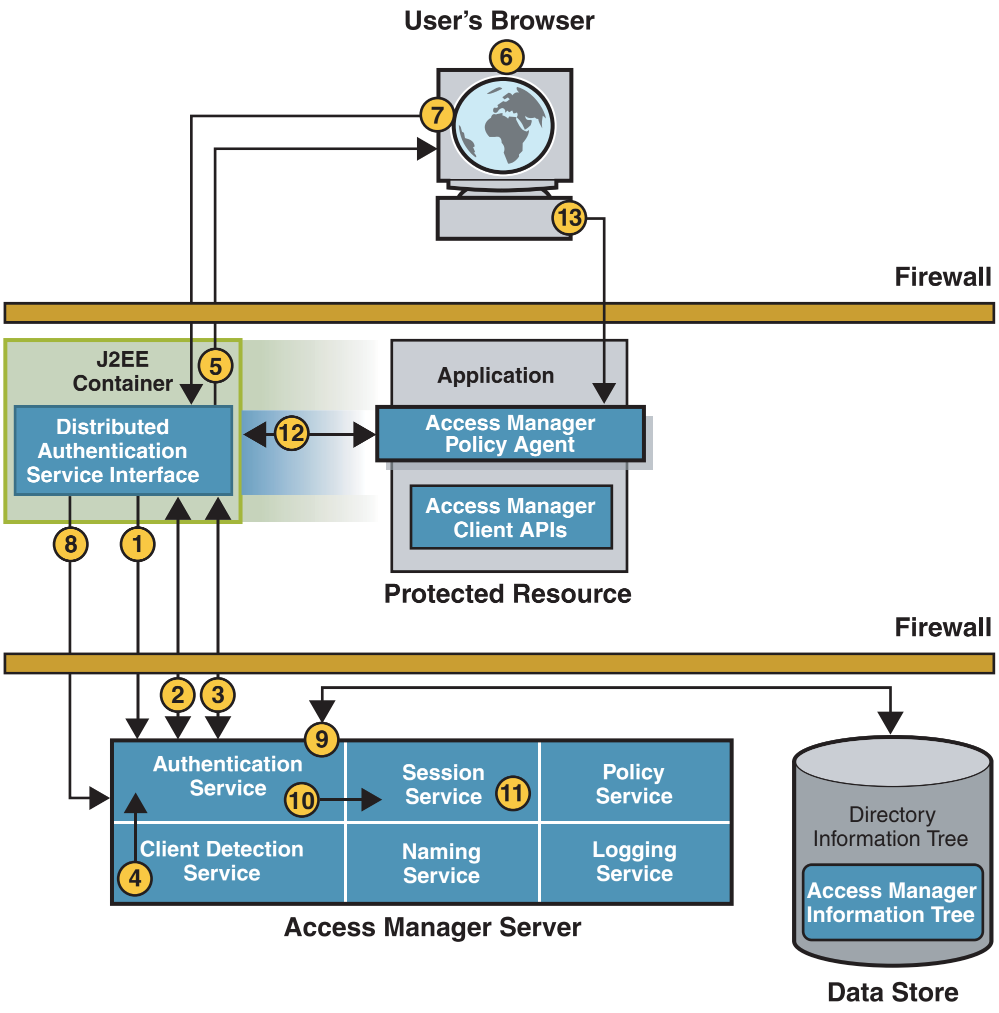
1. The user's browser sends an HTTP request to the protected resource.
2. The policy agent inspects the user's request and finds no session token.
3. The policy agent contacts the configured authentication URL.

   In this example, the authentication URL it is set to the URL of the Distributed Authentication User Interface Service.
4. The browser sends a GET request to the Distributed Authentication User Interface.
5. The Session Service creates a new session (session data structure) and generates a session token. The session token is a randomly-generated string that represents the user.
6. The Authentication Service sets the session token in a cookie.

The next part of the user session is User Authentication.

User Authentication

When the browser sends a GET request to the Distributed Authentication User Interface, the events in the following illustration occur. The accompanying text describes the process.



1. Using the parameters in the GET request, the Distributed Authentication User Interface contacts the Authentication Service installed on the Access Manager server.

2. The Authentication Service determines the appropriate authentication module to use based upon Access Manager configuration and the request parameters passed by the Distributed Authentication User Interface using the Authentication client APIs.

   For example, if Access Manager is configured to use the LDAP Authentication type of module, the Authentication Service determines that the LDAP Authentication login page will be used.
3. The Authentication Service determines which presentation callbacks should be presented, and sends to the Distributed Authentication User Interface all necessary credentials, requirements, and callbacks for use by the presentation framework layer.
4. The Client Detection Service determines which protocol, such as HTML or WML, to use to display the login page.
5. The Distributed Authentication User Interface returns to the Web browser a dynamic presentation extraction page along with the session cookie.

   The presentation extraction page contains the appropriate credentials request and callbacks info obtained from the Access Manager server.
6. The user's browser displays the login page.

   The user enters information in the Username and Password fields of the login page.
7. The browser replies to the Distributed Authentication User Interface with a POST that contains the required credentials.
8. The Distributed Authentication User Interface uses the Authentication client APIs to pass credentials to the Access Manager server.
9. The Authentication Service uses the appropriate authentication module type to validate the user's credentials.

   For example, if the LDAP authentication module type is used, the Authentication Service verifies that the username and password provided exist in the LDAP directory. Other authentication module types have different requirements.
10. Assuming authentication is successful, the Authentication Service activates the session by calling the appropriate methods in the Session Service.

    The Authentication Service stores information such as Login time, Authentication Scheme, and Authentication Level in the session data structure.
11. Once the session is activated, the Session Service changes the state of the session token to valid.
12. The Distributed Authentication User Interface replies to the protected resource with an SSOToken in a set-cookie header.
13. Now, the browser makes another request to the original resource protected by a policy agent.

    This time, the request includes a valid session token, created during the authentication process.

The next part of the user session is Session Validation.

Session Validation

After authentication, when the user's browser redirects the initial HTTP request to the server for a second time, the events in the following illustration occur. The accompanying text describes the process.

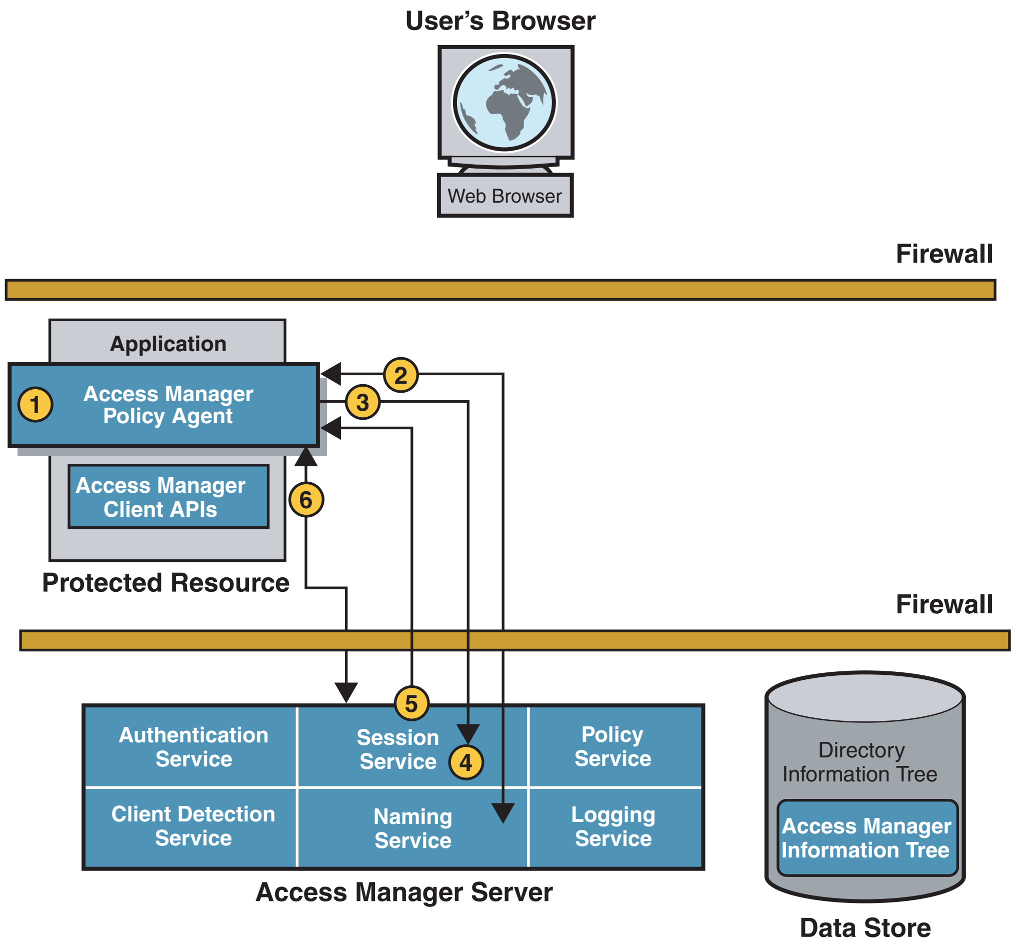


FIGURE 2–2 Session Validation

1. The policy agent intercepts the second access request.

   The request now contains a session token in the same DNS domain as Access Manager.

2. The policy agent determines the validity of the session token.
   a. The policy agent contacts the Naming Service to learn where the session token originated.

      The Naming Service allows clients to find the service URL for the internal services used by Access Manager. This information can then be used for communication regarding a session.
   b. The Naming Service decrypts the session token and returns the corresponding URLs . The URLs will be used by other services to obtain information about the user session.
3. The policy agent, using the information provided by the Naming Service, makes a POST request to the Session Service to validate the included session token.
4. The Session Service receives the request and determines whether the session token is valid based on the following criteria:
   a. Has the user been authenticated?
   b. Does a session data structure associated with the session token exist?
5. If all criteria are met, the Session Service responds that the session token is valid.

   This assertion is coupled with supporting information about the user session itself.
6. The policy agent creates a Session Listener and registers the Session Listener with the Session Service. This enables notification to the policy agent when a change in the session token state or validity occurs.

The next part of the user session is Policy Evaluation.

Policy Evaluation and Enforcement

After a session token has been validated, the policy agent determines if the user can be granted access to the server by evaluating it's defined policies. The following illustration and accompanying text describes the process.

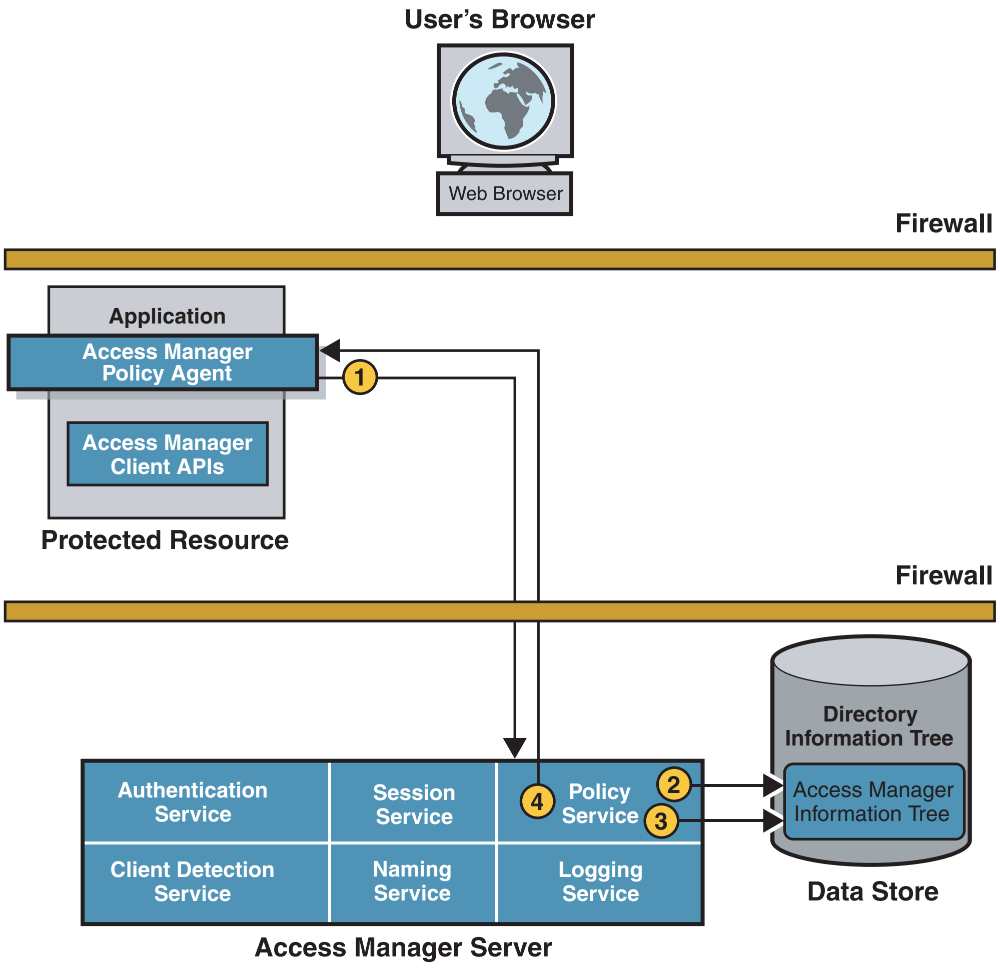


FIGURE 2–3 Policy Evaluation

1. The policy agent sends a request to the Policy Service.

   The request asks for decisions regarding resources in the policy agent's portion of the HTTP namespace. The request also includes additional environmental information. For example, IP address or DNS name could be included in the request because they might impact conditions set on a configuration policy.
2. The Policy Service checks for policies that apply to the request.

   Policies are cached in Access Manager. If the policies have not been cached already, then the policies are loaded from the Access Manager information tree in the Identity Repository.
3. If policies that apply to the request are found, the Policy Service checks if the user identified by the session token is a member of any of the Policy Subjects.
   a. If no policies that match the resource are found, the user is denied access. Skip to step 5.
   b. If policies are found that match the resource, and the user is a valid subject, the Policy Service evaluates the conditions of each policy. For example, *Is it the right time of day?* or *Are requests coming from the correct network?*
      - If the conditions are met, the policy applies.
      - If the conditions are not met, the policy is skipped.
4. The Policy Service aggregates all policies that apply, encodes a final decision to grant or deny access, and responds to the policy agent with the appropriate decision.

The next part of the user session is logging the policy evaluation results.

Logging Results

When the policy agent receives an allow decision from the Policy Service, the events in the following illustration occur. The accompanying text describes the process.

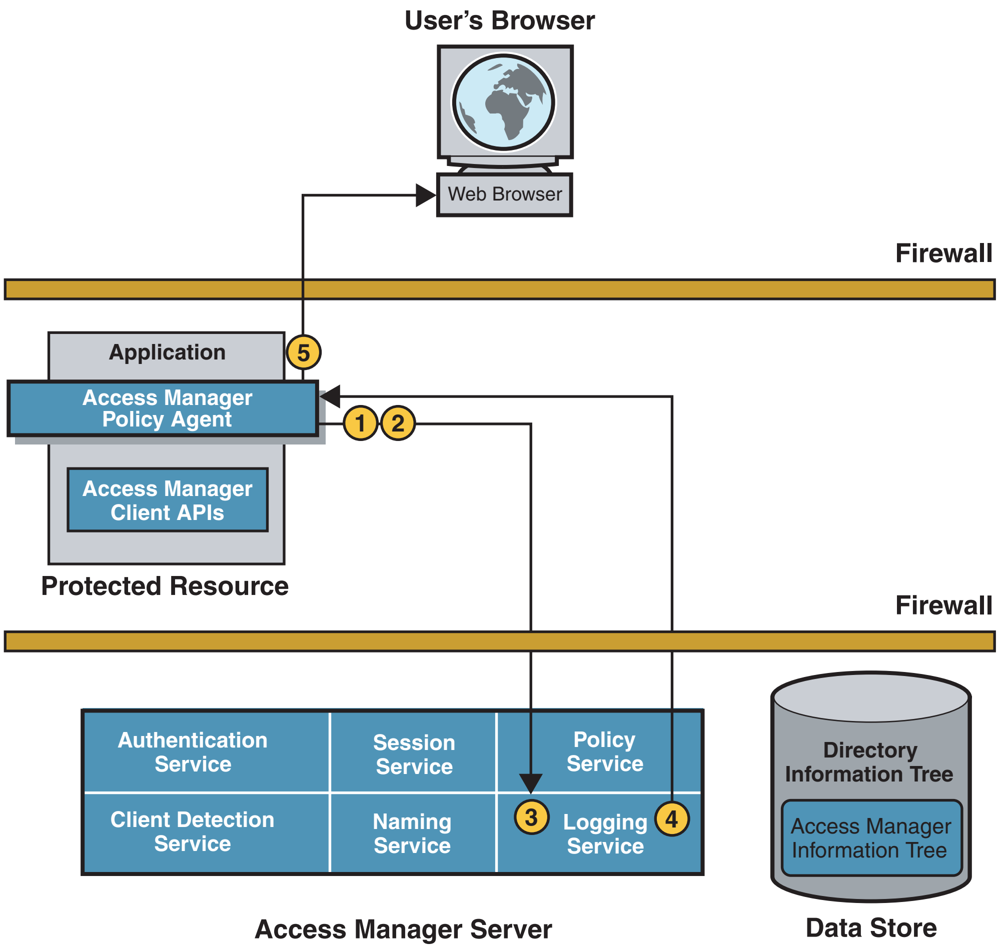


FIGURE 2–4 Logging the Policy Evaluation Results

1. The allow decision is cached in the policy agent, along with the session token, so that subsequent requests can be checked using the cache.

   It is no longer necessary for the policy agent to contact Access Manager. The cache will expire after an interval has passed or upon an explicit notification of change in policy or session status. The interval is configurable.
2. The policy agent issues a logging request to the Logging Service.
3. The Logging Service logs the policy evaluation results to a flat file (which can be signed) or to a JDBC store, depending upon the log configuration.
4. The Logging Service notifies the policy agent of the new log.
5. The policy agent allows or denies the user access to the application.
   a. If the user is denied access, the policy agent displays an "access denied" page.
   b. If the user is granted access, the resource displays its access page.

   Assuming the browser displays the application interface, this basic user session is valid until it is terminated. See "Session Termination" on page 58.

While the user is still logged in, if he attempts to log into another protected resource, the Single Sign-On session begins.

Single Sign-On Session

SSO is always preceded by a basic user session in which a session is created, its session token is validated, the user is authenticated, and access is allowed. SSO begins when the authenticated user requests a protected resource on a second server in the same DNS domain. The following example describes an SSO session by tracing what happens when an authenticated user accesses a second application in the same DNS domain as the first application. Because the Session Service maintains user session information with input from all applications participating in an SSO session, in this example, it maintains information from the application the user was granted access to in "Basic User Session" on page 44. We will assume the application previously accessed was a corporate benefits administration application. For this SSO session, the user is attempting access to an expense reporting application.

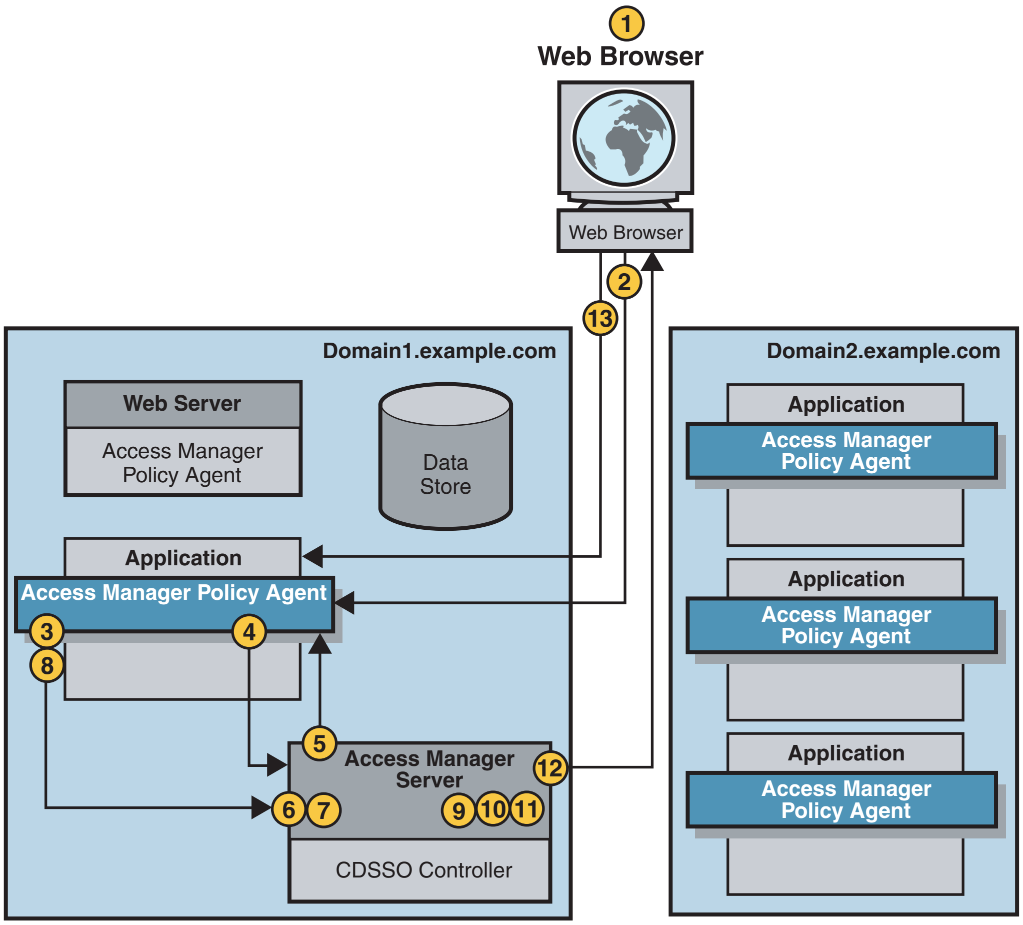


FIGURE 2–5 Single Sign-On Session

1. The user attempts to access an expense reporting application.

   Both the expense reporting application and the corporate benefits administration application are hosted on servers in the same domain.
2. The user's browser sends an HTTP request to the expense reporting application. The request includes the user's session token.
3. The policy agent intercepts and inspects the request to determine whether a session token exists.

   A session token indicates the user is already authenticated. Since the user was authenticated when the user logged in to the corporate benefits administration application, the Authentication Service is not required at this time. The SSO APIs retrieve the session data structure using the session token imbedded in the cookie. The session data structure is referred to as the *SSOToken* by the SSO APIs. The session token is referred to as the *SSOTokenID*.

4. The policy agent determines the validity of the session.

   For detailed steps, see "Session Validation" on page 48.
5. The Session Service sends a reply to the policy agent indicating whether the *SSOToken* is valid.
   - If the *SSOToken* is not valid, the user is redirected to the Authentication page.
   - If the *SSOToken* is valid, the Session Service creates a Session Listener.

     A Session Listener allows notification to the policy agent when a change in the *SSOToken* state or validity occurs.
6. The policy agent sends a request to the Policy Service.

   The request asks for a decision regarding resources in the policy agent's portion of the HTTP namespace.
7. The Policy Service checks for policies that apply to the request.
   - If Policy Service does not find policy allowing access to the protected resource, the user is denied access and the following events occur:
     a. The Logging Service logs a denial of access.
     b. The policy agent issues a *Forbidden* message to the user.

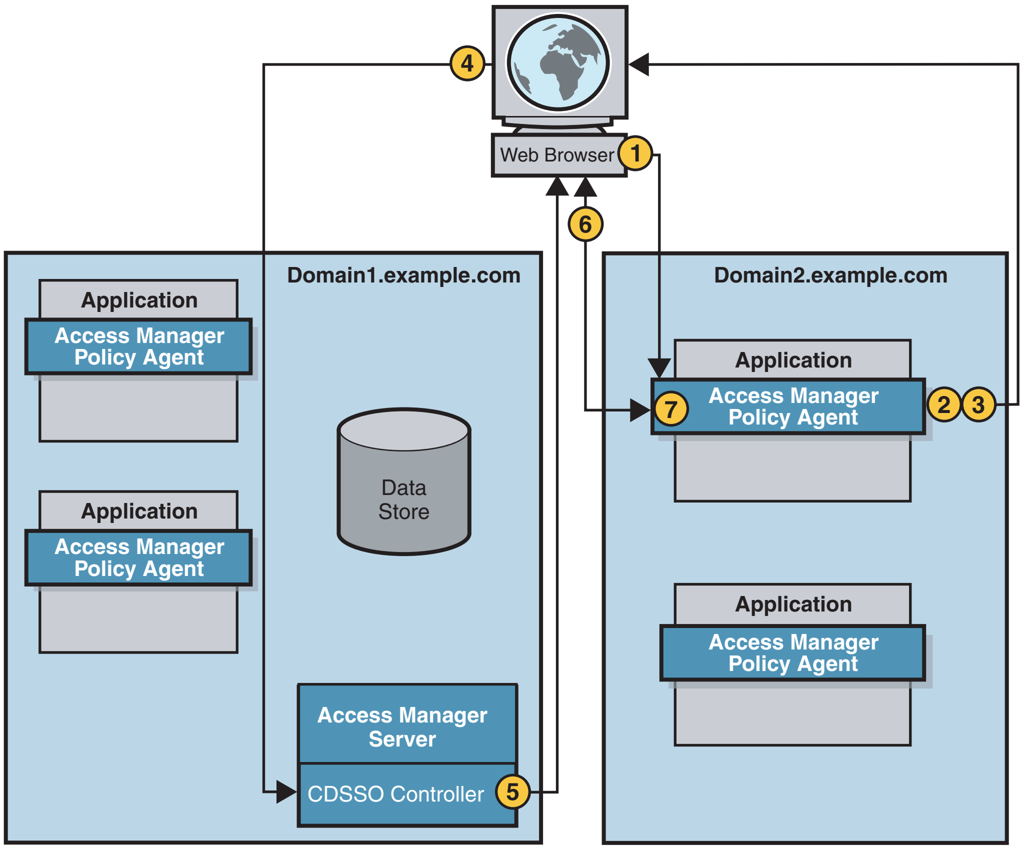
        The user can then be redirected to an administrator-specified page indicating the user was denied access.
   - If the Policy Service finds policy allowing access to the protected resource, the user is granted access to the protected resource and the SSO session is valid until it is terminated. See "Session Termination" on page 58.

While still logged in, if the user decides to attempt to log in to another protected resource located in a different DNS domain, Cross-Domain Single Sign-On takes place.

Cross-Domain Single Sign-On Session

CDSSO occurs when an authenticated user requests a protected resource on a different server in a different DNS domain. The user in the previous sections, "Basic User Session" on page 44 and "Single Sign-On Session" on page 53, for example, accessed applications in his company's DNS domain. In the following example, the same user will log in to a travel administration application supplied to his company by an external company. The travel administration application is hosted on the external company's DNS domain. In this scenario, the CDSSO Controller Service within Access Manager transfers the user's session information from the initial domain, making the it available to applications in this second domain.

When the user logs in to the travel administration application in the external DNS domain, the events in the following illustration occur. The process is described in the accompanying text.



1. The user's browser sends an HTTP request to the travel administration application.

2. The policy agent intercepts the request and inspects it to determine if a session token exists for the domain in which the travel administration application exists. One of the following occurs:
   - If a session token is present, the policy agent validates the session.
   - If no session token is present, the policy agent (which is configured for CDSSO) redirects the HTTP request to the CDSSO Controller Service.

     Note – The CDSSO Controller Service uses Liberty Alliance Project protocols to transfer sessions so the relevant parameters are included in the redirect.

   In this example, no session token for the second domain is found.
3. The policy agent redirects the HTTP request to the CDSSO Controller Service.
4. The user's browser allows the redirect to the CDSSO Controller Service.

   Recall that earlier in the user session the session token was set in a cookie in the initial domain which is now part of the redirect.
5. The CDSSO Controller Service's CDC Servlet receives the session token from the initial domain, extracts the user's session information, and formulates a Liberty POST profile response containing the information. The response is returned to the browser.
6. The user's browser automatically submits the form containing the Liberty document to the policy agent.

   The form is based upon the Action and the Javascript included in the Body tags `onLoad`.
7. The policy agent receives the document, extracts the session information, validates the session, and sets a session token in the cookie for the new DNS domain.

   For detailed information, see "Session Validation" on page 48.
8. The process continues with policy evaluation and results logging.

   See the following sections for detailed information.
   - "Policy Evaluation and Enforcement" on page 50
   - "Logging Results" on page 52
9. The policy agent allows or denies the user access to the application.
   a. If the user is denied access, the policy agent displays an "access denied" page.
   b. If the user is granted access, the resource displays its access page.

Assuming proper authorization, the policy agent responds to the user by presenting the travel administration application screen. This new cookie can now be used by all agents in the new domain, and the session is valid until it is terminated. (See "Session Termination" on page 58.)

Session Termination

A user session can be terminated in any of following ways:

- "User Ends Session" on page 58
- "Administrator Ends Session" on page 58
- "Access Manager Enforces Timeout Rules" on page 59
- "Session Quota Constraints" on page 59

User Ends Session

When a user explicitly logs out of Access Manager by clicking on a link to the Logout Service the following events occur:

1. The Logout Service receives the Logout request, and performs the following steps:
   a. Marks user's session as destroyed.
   b. Destroys session.
   c. Returns a successful logout page to the user.
2. The Session Service notifies applications which are configured to interact with the session.

   In this case, each of the policy agents was configured for Session Notification, and each is sent a document instructing the agent that the session is now invalid.
3. The policy agents flush the session from cache and the user session ends.

Administrator Ends Session

Access Manager administrators with appropriate permissions can terminate a user session at any time. When an administrator uses the Sessions tab in the Access Manager console to end a user's session, the following events occur:

1. The Logout Service receives the Logout request, and performs the following steps:
   a. Marks user's session as destroyed.
   b. Destroys session.
2. The Session Service notifies applications which are configured to interact with the session.

   In this case, each of the policy agents was configured for Session Notification, and each is sent a document instructing the agent that the session is now invalid.
3. The policy agents flush the session from cache and the user session ends.

Access Manager Enforces Timeout Rules

When a session timeout limit is reached, the Session Service completes the following steps:

1. Changes session status to `invalid`.
2. Displays time-out message to user.
3. Starts timer for purge operation delay (default is 60 minutes).
4. When purge operation delay time is reached, purges or destroys the session.
5. If a session validation request comes in after the purge delay time is reached, displays login page to user.

Session Quota Constraints

Access Manager allows administrators to constrain the amount of sessions one user can have. If the user has more sessions than the administrator will allow, one (or more) of the existing sessions can be destroyed.

CHAPTER 3

Authentication

The Sun Java System Access Manager Authentication Service determines whether a user is the person he claims to be. User authentication is the first step in controlling access to web resources within an enterprise. This chapter explains how the Authentication Service works with other Access Manager components to authenticate a user, or prove that the user's identity is genuine. Topics covered include:

Authentication Overview

The following example demonstrates how the Authentication Service works from the perspective of the user. A company employee must look up a colleague's phone number, so he uses a browser to access the company's online phone book. To log in to the phone book service, the employee provides a user name and password. Access Manager compares the user's input with data stored in a central user repository. If Access Manager finds a match for the user name, and if the given password matches the stored password, Access Manager authenticates the user's identity. After authentication, the policy evaluation process occurs. If the policy agent allows access to the user, the corporate phone book is displayed. The "Basic User Session" on page 44 section in the previous chapter contains a detailed description and illustration of the authentication process within a basic user session.

Note – The Authentication Service is client-type aware and supports all configured client types such as cookieless and cookie-enabled client types.

Authentication Modules

An *authentication module* is a plug-in that collects user information such as a user ID and password, and compares the information against entries in a database. If a user provides information that meets the authentication criteria, the user is granted access to the requested resource. If the user provides information that does not meet the authentication criteria, the user is denied access to the requested resource. Access Manager is installed with a number of authentication modules. The following table provides a brief description of them.

TABLE 3–1 Access Manage Authentication Module Types

Authentication Module Name	Description
Active Directory	Uses an Active Directory operation to associate a user ID and password with a particular Active Directory entry. You can define multiple Active Directory authentication configurations for a realm. Allows both LDAP and Active Directory to coexist under the same realm.
Anonymous	Enables a user to log in without specifying credentials. You can create an Anonymous user so that anyone can log in as Anonymous without having to provide a password. Anonymous connections are usually customized by the Access Manager administrator so that Anonymous users have limited access to the server.
Certificate	Enables a user to log in through a personal digital certificate (PDC). The user is granted or denied access to a resource based on whether or not the certificate is valid. The module can optionally require the use of the Online Certificate Status Protocol (OCSP) to determine the state of a certificate.
Data Store	Enables authentication against one or more configuration data stores within a realm.
HTTP Basic	Enables authentication to occur with no data encryption. Credentials are validated internally using the LDAP authentication module.
Java Database Connectivity (JDBC)	Enables authentication through any Structured Query Language (SQL) databases that provide JDBC-enabled drivers. The SQL database connects either directly through a JDBC driver or through a JNDI connection pool.

TABLE 3–1 Access Manage Authentication Module Types *(Continued)*

Authentication Module Name	Description
LDAP	Enables authentication using LDAP bind, a Directory Server operation which associates a user ID password with a particular LDAP entry. You can define multiple LDAP authentication configurations for a realm.
Membership	Enables user to self-register. The user create an account, personalizes it, and accesses it as a registered user without the help of an administrator. Implemented similarly to personalized sites such as `my.site.com`, or `mysun.sun.com`.
MSISDN	The Mobile Station Integrated Services Digital Network (MSISDN) authentication module enables authentication using a mobile subscriber ISDN associated with a device such as a cellular telephone. It is a non-interactive module. The module retrieves the subscriber ISDN and validates it against the user repository to find a user that matches the number.
RADIUS	Uses an external Remote Authentication Dial-In User Service (RADIUS) server to verify identities.
Security Assertion Markup Language (SAML)	Receives and validates SAML assertions on a target server by using either a web artifact or a POST response.
SafeWord®	Uses Secure Computing's SafeWord PremierAccess™ server software and SafeWord tokens to verify identities.
SecurID™	Uses RSA ACE/Server software and RSA SecurID authenticators to verify identities.
UNIX®	Solaris and Linux modules use a user's UNIX identification and password to verify identities.
Windows Desktop Single Sign-On (SSO)	Allows a user who has already authenticated with a key distribution center to be authenticated with Access Manager without having to provide the login information again. Leverages Kerberos authentication and is specific to the Windows operating system.
Windows NT	Uses a Microsoft Windows NT™ server to verify identities.

You can use the Access Manager Console to enable and configure the authentication modules that are installed with Access Manager by default. You can also create and configure multiple instances of a particular authentication module. (An *authentication module instance* is a child entity that extends the schema of a parent authentication module and adds its own subschema.)

Finally, you can write your own custom authentication module (or plug-in) to connect to the Access Manager authentication framework. See Chapter 4, “Managing Authentication,” in *Sun Java System Access Manager 7.1 Administration Guide* for detailed information about enabling and configuring default authentication modules and authentication module instances. See Chapter 2, “Using Authentication APIs and SPIs,” in *Sun Java System Access Manager 7.1 Developer's Guide* for more information about writing custom authentication modules.

Authentication Configuration Services

The Authentication framework includes the following pluggable and customizable services:

- “General Authentication Service” on page 64
- “Authentication Configuration Service” on page 64

General Authentication Service

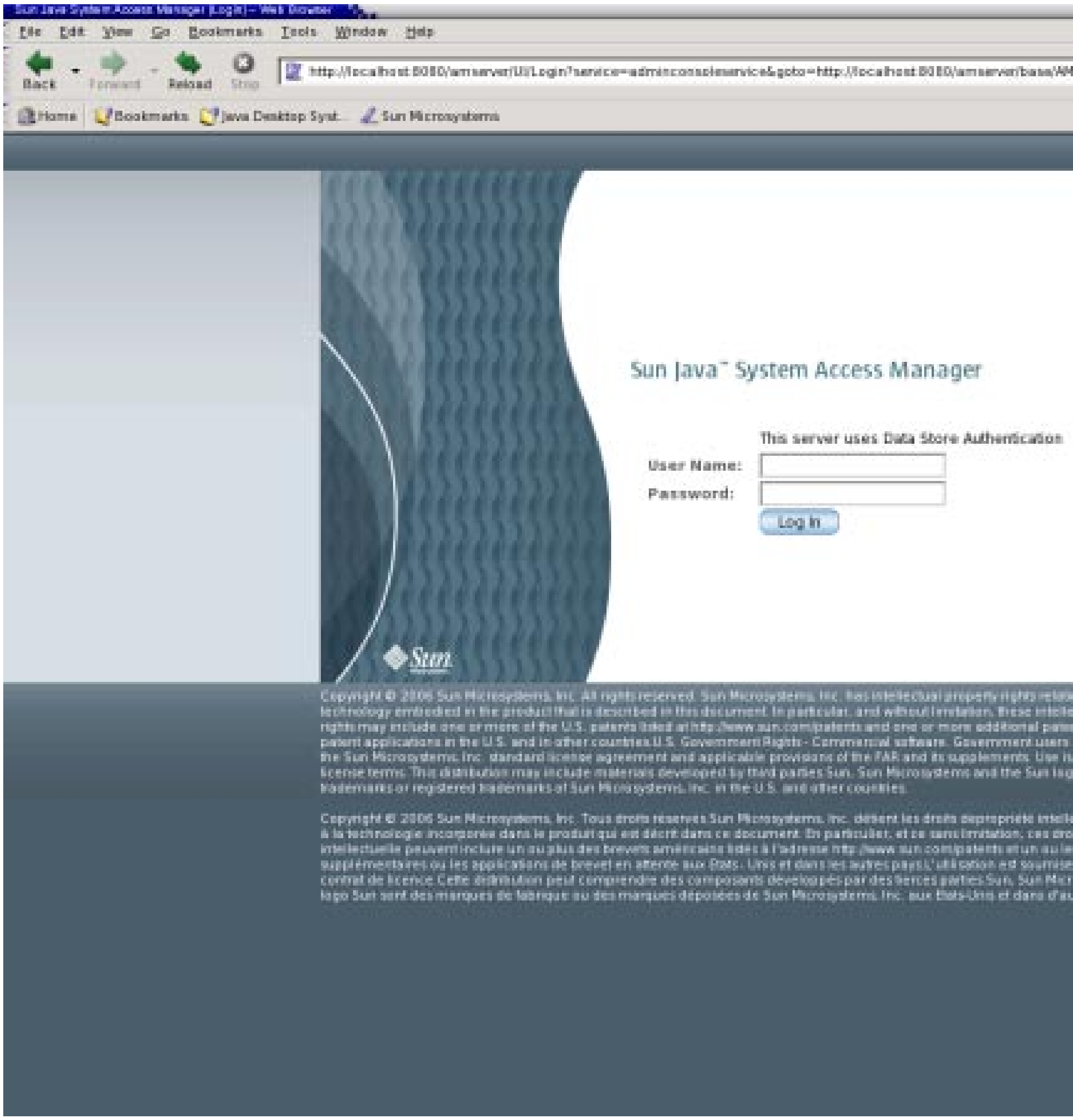
The General Authentication Service is used for server-related attribute configuration. (Some of the attributes described in this service are default attributes for all Access Manager authentication modules.) Configuration is available in each configured realm. The General Authentication Service enables the Access Manager administrator to define default values for a realm's authentication parameters. These values can be used if no overriding value is defined in the specified authentication module. The default values for the General Authentication Service are defined in the `amAuth.xml` file and stored in the Access Manager information tree after installation. For more information, see Chapter 4, “Managing Authentication,” in *Sun Java System Access Manager 7.1 Administration Guide*.

Authentication Configuration Service

The Authentication Configuration Service describes all the dynamic attributes for service-based authentication. This service is used for configuring roles. When you assign a service to a role, you can also assign other attributes such as a success URL or an authentication post-processing class to the role. For more information, see “Role-based Authentication” in *Sun Java System Access Manager 7.1 Administration Guide*.

Authentication Service User Interface

The Authentication Service implements a user interface that is separate from the Access Manager administration console. The Authentication Service user interface provides a dynamic and customizable means for gathering authentication credentials. When a user requests access to a protected resource, the Authentication Service presents a web-based login page and prompts the user for user name and password. Once the credentials have been passed back to Access Manager and authentication is successful, the user can gain access based on the user's specific privileges:


Sun Java System Access Manager (Login) – Web Browser
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:8080/amserver/UI/Login?service=adminconsoleservice&goto=http://localhost:8080/amserver/base/AM
Home Bookmarks Java Desktop Syst... Sun Microsystems
Sun Java™ System Access Manager
Log In
Sun

This server uses Data Store Authentication

User Name:

Password:

The Authentication Service user interface can be used for the following:

- Administrators can access the administration portion of the Access Manager console to manage their realm's identity data.
- Users can access their own profiles to modify personal data.

- A user can access a resource defined as a redirection URL parameter appended to the login URL.
- A user can access the resource protected by a policy agent.

Access Manager 7.1 provides customization support for the Authentication Service user interface. You can customize JavaServer Pages™ (JSP™) and the file directory level for `/org/service/locale/client_type.` See Chapter 12, "Customizing the Authentication User Interface," in *Sun Java System Access Manager 7.1 Developer's Guide* for more information.

Distributed Authentication User Interface

Access Manager provides a remote authentication user interface component to enable secure, distributed authentication across two firewalls. A web browser communicates an HTTP request to the remote authentication user interface which, in turn, presents a login page to the user. The web browser then sends the user login information through a firewall to the remote authentication user interface which, in turn, communicates through the second firewall to the Access Manager server. The Distributed Authentication User Interface enables a policy agent or an application that is deployed in a non-secured area to communicate with an instance of the Access Manager Authentication Service installed in a secured area of the deployment. The following figure illustrates this scenario.

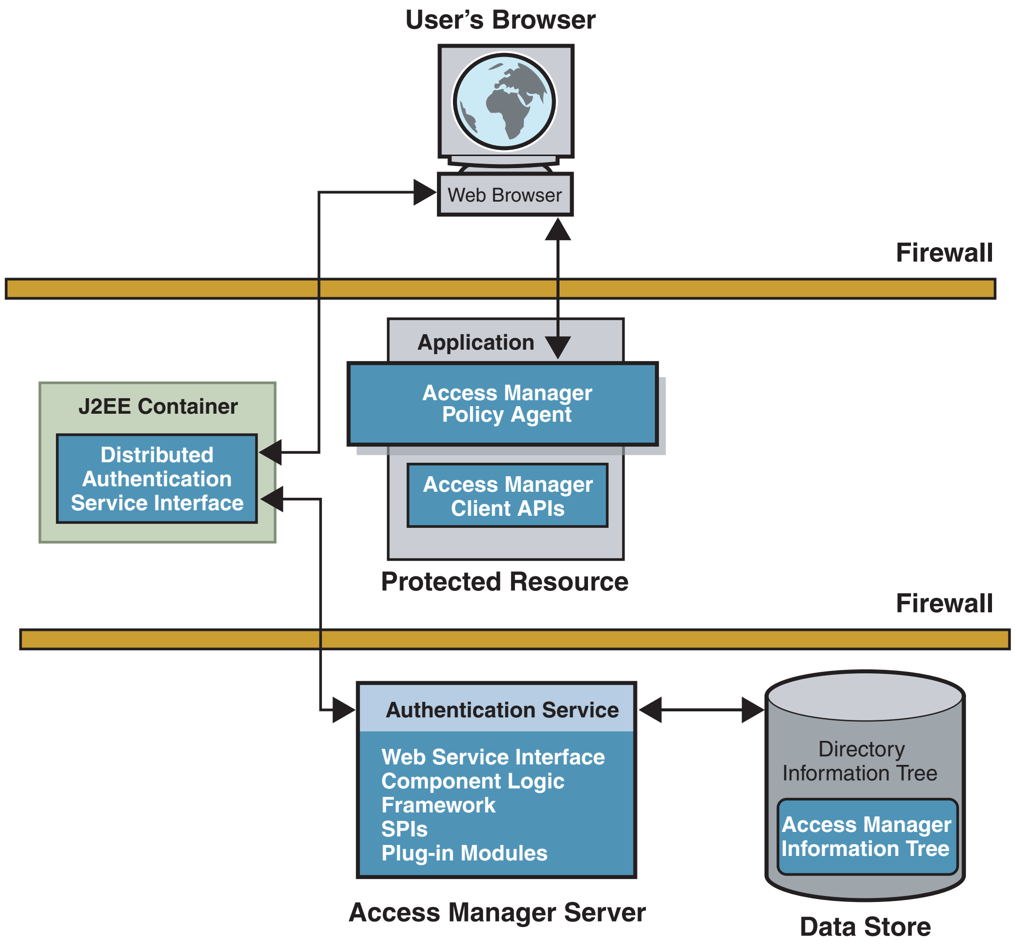


FIGURE 3–1 Distributed Authentication

The Distributed Authentication User Interface uses a JATO presentation framework and is customizable. (See screen capture in "Authentication Service User Interface" on page 65.) You can install the Distributed Authentication User Interface on any servlet-compliant web container within the non-secure layer of an Access Manager deployment. The remote component then works with the Authentication client APIs and authentication utility classes to authenticate web users. For a more detailed process flow, see "User Authentication" on page 46. For detailed installation and configuration instructions, see Chapter 11, "Deploying a Distributed Authentication UI Server," in *Sun Java System Access Manager 7.1 Postinstallation Guide*.

Inside the Core Authentication Component

The core Authentication component is where default configurations are stored and the following authentication processes are invoked.

Client Detection Service

An initial step in the authentication process is to identify the type of client making the HTTP(S) request. This Access Manager feature is known as *client detection*. The URL information in the HTTP(S) request is used to retrieve the client's characteristics. Based on these characteristics, the appropriate authentication pages are returned. For example, when a Netscape browser is used to request a web page, Access Manager displays an HTML login page. Once the user is validated, the client type `Netscape browser` is added to the session token. For more information, see Chapter 7, "Client Detection Service," in *Sun Java System Access Manager 7.1 Developer's Guide*.

Authentication Type Configurations

After granting or denying access to a resource, Access Manager checks for information about where to redirect the user. A specific order of precedence is used when checking this information. The order is based on whether the user was granted or denied access to the protected resource, and on the type of authentication specified. When you install Access Manager, a number of authentication types are automatically configured for you. Following is a list of authentication type configurations. For more information, see "Authentication Types" in *Sun Java System Access Manager 7.1 Administration Guide*.

Realm-based Authentication.	User authenticates to a realm or subrealm in the Access Manager information tree.
Role-based Authentication.	User authenticates to a role within a realm or subrealm of the directory information tree. A *role* is a grouping of like items in the repository. A *static role* is created when an attribute is

	assigned to a specific user or container. A *filtered role* is dynamically generated based on an attribute contained in the a user's or container's entry. For example, all users that contain a value for the `employee` attribute can be automatically included in a filtered role named *employees*.
Service-based Authentication.	User authenticates to a specific service or application registered to a realm or subrealm.
User-based Authentication.	User authenticates using an authentication process configured specifically for him or her.
Authentication Level-based Authentication	Administrator specifies the security level of the modules to which identities can authenticate.
Module-based Authentication.	User specifies the module instance to which the user will authenticate.
Organization-based Authentication.	User authenticates to an organization or suborganization. **Note –** This authentication type only applies to Access Manager when installed in Legacy mode.

Login URLs and Redirection URLs

In the last phase of the authentication process, Access Manager either grants or denies access to the user. If access is granted, Access Manager uses a login URL to display a page in the browser. If access is denied, Access Manager uses a redirection URL to display an alternate page in the browser. A typical alternate page contains a brief message indicating the user has been denied access.

Each authentication type (realm-based, role-based, and so forth) uses a login URL or redirection URL based on a specific order of precedence, and on whether the authentication succeeded or failed. For a detailed description of how Access Manager proceeds through the order of precedence, see "Authentication Types" in *Sun Java System Access Manager 7.1 Administration Guide*.

Account Locking

The Authentication Service provides an account locking feature that prevents a user from completing the authentication process after a specified number of failures. Only modules that throw an Invalid Password Exception can leverage the Account Locking feature. Access Manager sends email notifications to administrators when account lockouts occur. Account locking activities are also logged. Access Manager supports the following types of account locking:

Physical Locking.	By default, user accounts are `active` or physically unlocked. You can initiate physical locking by changing the status of an LDAP attribute in the user's profile to `inactive`. The account remains physically locked until the attribute is changed to `active`.
Memory Locking.	You can enable memory locking by changing the Login Failure Lockout Duration attribute to a value greater then `0`. The user's account is locked in memory for the number of minutes you specified. The account is unlocked after the time period elapses. You can configure Memory Locking so that a user account is locked in memory after a specified number of tries. The user account will be locked when AM server is restarted.

The account locking feature is disabled by default. You can enable it by using the Access Manager console. For more information, see "Account Locking" in *Sun Java System Access Manager 7.1 Administration Guide*.

Authentication Chaining

You can configure one or more authentication module instances so that a user must pass authentication credentials to all of them before the user is allowed access. This feature is called *authentication chaining*. Access Manager uses the Java Authentication and Authorization Service (JAAS) framework (already integrated in the Authentication Service) to implement authentication chaining. The JAAS framework validates all user IDs used during the authentication process, and maps them all to one user. (The mapping is based on the configuration of the User Alias List attribute in the user's profile.) If all the maps are correct, then a valid session token is issued to the user. If all the maps are not correct, the user is denied a valid session token. Once authentication to all modules in the chain succeeds or fails, control is returned to the Authentication Service from the JAAS framework.

You can configure authentication chaining by realm, user, role, or service. Determining access is based upon control flags you specify for the chain. Authentication modules use one of the following control flags to indicate requirements for successful authentication.

Requisite.	The LoginModule is required to succeed. If it succeeds, authentication continues down the LoginModule list. If it fails, control immediately returns to the application (authentication does not proceed down the LoginModule list).
Required.	Authentication to this module is required to succeed. If any of the required modules in the chain fails, the whole authentication chain will fail and the user will be notified of this.
Sufficient.	The LoginModule is not required to succeed. If it does succeed, control immediately returns to the application (authentication does not proceed down the LoginModule list). If it fails, authentication continues down the LoginModule list.
Optional.	The LoginModule is not required to succeed. Whether it succeeds or fails, authentication still continues to proceed down the LoginModule list.

For more information, see "Authentication Chaining" in *Sun Java System Access Manager 7.1 Administration Guide*.

Fully Qualified Domain Name Mapping

Fully Qualified Domain Name (FQDN) mapping enables the Authentication Service to take corrective action in the case where a user may have typed in an incorrect URL. This is necessary, for example, when a user specifies a partial host name or IP address to access protected resources. This feature is also used to allow access to one instance of Access Manager using many different aliases. For example, you might configure one instance of Access Manager as `intranet.example.com` for employees and `extranet.example.com` for partners. For more information, see "Fully Qualified Domain Name Mapping" in *Sun Java System Access Manager 7.1 Administration Guide*.

Persistent Cookie

A persistent cookie is an information packet that is written to the user's hard drive and, therefore, continues to exist after the web browser is closed. The persistent cookie enables a user to log into a new browser session without having to reauthenticate. For more information, see "Persistent Cookie" in *Sun Java System Access Manager 7.1 Administration Guide*.

Session Upgrade

The Authentication Service allows for the upgrade of a valid session token based on a second, successful authentication performed by the same user. If a user with a valid session token attempts to authenticate to a second resource secured under the realm to which he is currently authenticated, and this second authentication request is successful, the Authentication Service updates the session with the new properties based on the new authentication. If the authentication fails, the current user session is returned without an upgrade. If the user with a valid session attempts to authenticate to a resource secured by a different realm, the user will receive a message asking whether the user would like to authenticate to the new realm. The user can choose to maintain the current session, or can attempt to authenticate to the new realm. Successful authentication will result in the old session being destroyed and a new one being created. For more information, see “Session Upgrade” in *Sun Java System Access Manager 7.1 Administration Guide*.

Validation Plug-in Interface

An administrator can write username or password validation logic appropriate for a particular realm, and plug the logic into the Authentication Service. Before authenticating a user or changing the user password, Access Manager will invoke this plug-in. If the validation is successful, authentication continues. If validation fails, an authentication failed page will be thrown. The plug-in extends the `com.iplanet.am.sdk.AMUserPasswordValidation` class which is part of the Service Configuration SPI. For more information, see “Validation Plug-in Interface” in *Sun Java System Access Manager 7.1 Administration Guide*.

Note – The Validation Plug-In Interface is only supported by the LDAP and Membership authentication module types.

JAAS Shared State

The JAAS shared state enables sharing of both user ID and password between authentication module instances. Options are defined for each authentication module type by realm, user, service and role. If an authentication fails with the credentials from the shared state, the authentication module restarts the authentication process by prompting for its required credentials. If it fails again, the module is marked failed. After a commit, an abort, or a logout, the shared state will be cleared. For more information, see “JAAS Shared State” in *Sun Java System Access Manager 7.1 Administration Guide*.

Authentication Programming Interfaces

Access Manager provides both Java APIs and C APIs for writing authentication clients that remote applications can use to gain access to the Authenticate Service. Communication between the APIs and the Authentication Service occurs by sending XML messages over HTTP(S). The Java and C APIs support all authentication types supported by the browser-based user interface. Clients other than Java and C clients can also use the XML/HTTP interface directly to initiate an authentication request.

Additionally, you can add custom authentication modules to Access Manager by using the `com.iplanet.authentication.spi` package. This SPI implements the JAAS LoginModule, and provides additional methods to access the Authentication Service and module configuration properties files. Because of this architecture, any custom JAAS authentication module will work within the Authentication Service.

For more information, see Chapter 2, “Using Authentication APIs and SPIs,” in *Sun Java System Access Manager 7.1 Developer’s Guide*.

CHAPTER 4

Authorization and the Policy Service

The Sun Java System Access Manager Policy Service determines if a user has been given permission by a recognized authority to access a protected resource. The process is referred to as user authorization. This chapter describes how the various parts of the Policy Service work together to perform authorization. Topics covered include:

Authorization Overview

A *policy* is a rule that defines who is authorized to access a resource. A single policy can define either binary or non-binary decisions. A binary decision is yes/no, true/false or allow/ deny. A non-binary decision represents the value of an attribute. For example, a mail service might include a `mailboxQuota` attribute with a maximum storage value set for each user. In general, a policy is configured to define what a subject can do to which resource and under what conditions.

The Access Manager Policy Service allows administrators to define, modify, and delete policies for protected resources within the Access Manager deployment. Configured policies are grouped into *realms* and stored in the Access Manager information tree. The Policy Service relies on the following:

- A Policy Enforcement Point (PEP) protects an enterprise's resources by enforcing access control. The policy agent is the PEP.
- A Policy Decision Point (PDP) is where the policy is evaluated and a determination is made. The Policy Service is the PDP.

- A data store in which configured policies are stored and from which they are retrieved. The proprietary Access Manager information tree is the data store.

The Access Manager Policy Service uses configured policies to determine if a user has been given permission by a recognized authority to access a protected resource. When a user attempts to access a resource protected by a PEP, the PEP contacts the PDP to get a policy decision. The Policy Service evaluates the policies that protect the resource and are applicable to the requesting user. This results in a policy decision indicating whether the user is allowed to access the resource. Upon receiving the decision, the PEP allows or denies access accordingly. This whole process is referred to as *authorization*.

Access Control and Realms

When a user logs into an application, Access Manager plug-ins retrieve all user information, authentication properties, and authorization policies that the Access Manager framework needs to form a temporary, virtual user identity. The Authentication Service and the Policy Service use this virtual user identity to authenticate the user and enforce the authorization policies, respectively. All user information, authentication properties, and authorization policies is contained in realms. You can create a realm when you want to apply policies to a group of related subjects, services or servers. For example, you can create a realm that groups all servers and services that are accessed regularly by employees in one region. And, within that regional grouping realm, you can group all servers and services accessed regularly by employees in a specific division such as Human Resources. A configured policy might state that all Human Resources administrators can access the URL `http://HR.example.com/HRadmins/index.html.`. You might also add constraints to this policy: it is applicable only Monday through Friday from 9:00 a.m. through 5:00 p.m. Realms facilitate the delegation of policy management privileges.

Note – Access control realms can be configured to use any user database.

Policy Types

The Policy Service authorizes access to a user based on the policies stored in the Access Manager information tree. The following sections contain information on the two types of policies you can create using Access Manager:

- "Normal Policy" on page 77
- "Referral Policy" on page 79

Normal Policy

A *normal policy* specifies a protected resource and who is allowed to access the resource. The protected resource can be anything hosted on a protected server. Examples of protected resources are applications, document files, images, or the server itself. Only a Top-Level Realm or Policy Administrator can create or manage polices that apply to a resource. A normal policy consists of rules, subjects, conditions, and response providers. The following sections contain information regarding these elements.

- "Rules" on page 77
- "Subjects" on page 77
- "Conditions" on page 78
- "Response Providers" on page 79

Rules

A *rule* defines the policy itself by specifying a resource, one or more sets of an action, and values for each action.

- A *resource* defines the specific object that is being protected. Examples of protected objects are an HTML page on a web site, or a user's salary information accessed using a human resources service.
- An *action* is the name of an operation that can be performed on the resource. Examples of web page actions are POST and GET. An allowable action for a human resources service might be `canChangeHomeTelephone`.
- A *value* defines the permission for the action. Examples are `allow` and`deny`.

Subjects

A *subject* specifies, by implication, the user or collection of users that the policy affects. You can assign the following subjects to policies:

Access Manger Roles	The roles you create and manage under the Realms Subject tab can be added as a value of the subject.
Access Manager Identity	The identities you create and manage under the Realms Subject tab can be added as a value of the subject.
Authenticated Users	Any user with a valid SSOToken is a member of this subject. All authenticated users would be member of this Subject, even if they have authenticated to a realm that is different from the realm in which the policy is defined.
LDAP Groups	Any member of an LDAP group can be added as a value of this subject.

LDAP Roles	Any LDAP role can be added as a value of this subject. An LDAP Role is any role definition that uses the Sun Java System Directory Server role capability. These roles have object classes mandated by Directory Server role definition. The LDAP Role Search filter can be modified in the Policy Configuration Service to narrow the scope and improve performance.
LDAP Users	Any LDAP user can be added as a value of this subject.
Organization	Any realm can be added as a value of this subject
Web Services Clients	Valid values are the DNs of trusted certificates in the local JKS keystore, which corresponds to the certificates of trusted web service clients (WSCs). A WSC identified by the SSOToken is a member of this subject, if the DN of any principal contained in the SSOToken matches any selected value of this subject. This subject has dependency on the Access Manager implementation of the Liberty Alliance Project Identity Web Services Framework and should be used only by web service providers to authorize WSCs.

Note – You can implement custom subjects by using the Policy APIs.

Conditions

A *condition* specifies additional constraints that must be satisfied for a policy be applicable. For example, you can define a condition to limit a user's network access to a specific time period. The condition might state that the subject can access the network only between 7:00 in the morning and 10:00 at night. Access Manager provides the following conditions:

Active Session Time	Sets a condition based on constraints configured for user session time such as maximum session time.
Authentication Chain	The policy is applicable if the user has successfully authenticated to the authentication chain in the specified realm. If the realm is not specified, authentication to any realm at the authentication chain will satisfy the condition.
Authentication Level	The Authentication Level attribute indicates the level of trust for authentication. The policy is applicable if the user's authentication level is greater than or equal to the Authentication Level set in the condition, or if the user's authentication level is less than or equal to the Authentication Level set in the condition, depending on the configuration.

Authentication Module Instance	The policy applies if the user has successfully authenticated to the authentication module in the specified realm. If the realm is not specified, authentication to any realm at the authentication module will satisfy the condition.
IP Address/DNS Names	Sets a condition based on a range of IP Addresses, or a DNS name.
Current Session Properties	Decides whether a policy is applicable to the request based on values set in the user's Access Manager session.
LDAP Filter Condition	The policy is applicable when the defined LDAP filter locates the user entry in the LDAP directory that was specified in the Policy Configuration service.
Realm Authentication	The policy applies if the user has authenticated to the specified realm.
Time	Sets the condition based on time constraints (time, day, date, time zone).

Note – You can implement custom conditions by using the Policy APIs.

Response Providers

Response providers are plug-ins that provide policy response attributes. Policy response attributes typically provide values for attributes in the user profile. The attributes are sent with policy decisions to the PEP which, in turn, passes them in headers to an application. The application typically uses these attributes for customizing pages such as a portal page. Access Manager includes one implementation, the `IDResponseProvider`.

Note – Custom response providers are not supported in this version of Access Manager.

Referral Policy

A Realm Administrator or Policy Administrator at the root or top level of the Access Manager information tree can create policy for any resource. A *referral policy* enables a Realm Administrator or a Policy Administrator to delegate policy configuration tasks. A referral policy delegates both policy creation and policy evaluation, and consists of one or more rules and one or more referrals.

- A *rule* defines the resource whose policy creation or evaluation is being referred.

- A *referral* defines the identity object to which the policy creation or evaluation is being referred.

Referral policies delegate policy management privileges to another entity such as a peer realm, a subrealm, or even a third-party product. (You can implement custom referrals by using the Policy APIs.) For example, a top-level realm exists named `ISP`. It contains two subrealms: `company1` and `company2`. The Top-Level Administrator for `ISP` can delegate policy management privileges so that a Realm Administrator in `company1` can create and manage policies only within the`company1` realm, and a Realm Administrator in `company2` can create and manage policies only within the `company2` realm. To do this, the Top-Level Administrator creates two referral policies, defining the appropriate realm in the rule and the appropriate administrator in the referral.

Note – An administrator or Policy Administrator for realms configured below the root level of the Access Manager information tree have permission to create policies only for resources delegated to that realm.

Policy Framework

The Policy framework in Access Manager are the services where policy management and administration are implemented. The Policy framework includes the following:

- "Policy Service" on page 80
- "Policy Configuration Service" on page 81

Policy Service

The Policy Service is defined using the `amPolicy.xml`. It performs the following functions:

- Provides a means for defining and managing access policies.
- Provides a means for defining custom policy plug-ins by providing names and class locations.
- Evaluates access policies.
- Acts as a PDP to deliver the result of a policy evaluation.

In order to configure for custom policy plug-ins, modify `amPolicy.xml` and use `amadmin` to reload it. See "Developing Custom Subjects, Conditions, Referrals, and Response Providers" in *Sun Java System Access Manager 7.1 Developer's Guide*.

Policy Configuration Service

The Policy Configuration Service provides a means to specify how policies are defined and evaluated. The Policy Configuration Service enables you to specify, for example:

- Which directory to use for subject lookup
- The directory password
- Which search filters to use
- Which subjects, conditions, and response providers to use

This configuration can be done within a realm or a subrealm and is accessible through the Access Manager console.

Policy SPIs and Plug-Ins Layer

Access Manager includes SPIs that work with the Policy framework to create and manage policies. You can develop customized plug-ins for creating custom policy subjects, referrals, conditions, and response providers. For information on creating custom policy plug-ins, see the *Sun Java System Access Manager 7.1 Developer's Guide*.

The following table summarizes the Policy service provider interfaces (SPIs), and lists the specialized Policy plug-ins that come bundled with Access Manager.

TABLE 4–1 Policy Service Provider Interfaces

Interface	Description
Subject	Defines a set of authenticated users for whom the policy applies. The following Subject plug-ins come bundled with Access Manager: Access Manager Identity Subject, Access Manager Roles, Authenticated Users, LDAP Groups, LDAP Roles, LDAP Users, Organization Web, and Services Clients.
Referral	Delegates management of policy definitions to another access control realm.
Condition	Specifies applicability of policy based on conditions such as IP address, time of day, authentication level. The following Condition plug-ins come bundled with Access Manager: Authentication Level, Authentication Scheme, IP Address, LE Authentication Level, Session, SessionProperty, and Time.
Resource Name	Allows a pluggable resource.
Response Provider	Gets attributes that are sent along with policy decision to the policy agent, and used by the policy agent to customize the client applications. Custom implementations of this interface are now supported in Access Manager 7.1.

Policy Client APIs

Access Manager provides client APIs that implement policy evaluation logic on a remote web server or application server. For policy client API information, see the *Sun Java System Access Manager 7.1 Developer's Guide*.

CHAPTER 5

Federation, SAML, and Web Services

This chapter explains the concept of identity federation, and describes the role of the Federation feature in Access Manager. For detailed information about enabling or managing identity federation, or using the Federation Management APIs and SPIs, see the *Sun Java System Access Manager 7.1 Federation and SAML Administration Guide*.

This chapter includes the following topics:

Federating Identities

Consider the many times an individual accesses services on the Internet in a single day. At work, he uses the company intranet to perform a multitude of tasks including reading and sending email, looking up information in the company phone book, searching internal databases, and submitting expense reports and other online forms. At home, he checks personal email, logs into an online news service, finalizes travel plans via a travel agent's web site, and shops. Each time he accesses one of these services, he must log in and identify himself.

A *local identity* refers to the set of attributes or information that identify the user to a particular service provider. These attributes typically include a name and password, plus an email address, account number or other identifier. Most users have many local identities. For example, the individual in our scenario might log in at work using an employee number but, at home, he might log in to his travel agent as Joe Smith. He might use an account number to log in to the car rental agency he uses frequently, and he might log in to an airline using a frequent flyer number.

Identity federation allows a user to consolidate the many local identities he has configured among multiple service providers. With a *federated identity*, the individual can log in at one service provider site and move to an affiliated service provider site without having to re-authenticate or re-establish his identity. For example, with a federated identity, the individual might want to access both his personal email account and his business email account from his workplace, and move back and forth between the two services without having to log in each time. Or at home he might want to log in to an online travel agency to book airline tickets and make hotel reservations. It is a convenience for the user to be able to access all of these services without having to provide different user names and passwords at each service site. It is a valuable benefit to the user when he can do so safely, knowing that his identity information is secure.

The Liberty Alliance Project

The Liberty Alliance Project develops and delivers specifications that enable federated network identity management and supporting web services. Using web redirection and open-source technologies such as SOAP and XML, they enable distributed, cross-domain interactions. The specifications include:

- Liberty Identity Federation Framework
- Liberty Identity Web Services Framework
- Liberty Identity Services Interface Specifications
- Schema Files and Service Definition Documents
- Support Documents

An overview of these specifications as well as background information on the Liberty Alliance Project can be found in Chapter 1, "Introduction to the Liberty Alliance Project," in *Sun Java System Access Manager 7.1 Federation and SAML Administration Guide*. The specifications themselves can be found on the Liberty Alliance Project web site.

How Federation Works

The goal of the Liberty Alliance Project specifications is to enable individuals and multiple organizations to easily conduct network transactions while protecting the individual's identity. When organizations form a *circle of trust*, they agree to exchange user authentication information using web service technologies. A circle of trust must contain at least one identity provider, a service provider that maintains and manages identity information. It also includes multiple service providers that offer web-based services to users. Once a circle of trust is established, single sign-on is enabled between all the providers and users can federate their multiple identities.

In Access Manager, the circle of trust is referred to as an *authentication domain*. An authentication domain contains entities that are grouped together for the purpose of identity federation. A travel portal is a good example of an authentication domain. Typically, a travel portal is a web site designed to help you access various travel-related service providers from one location. The travel portal forms a partnership with each hotel, airline, and car rental agency displayed on its web site. The user registers with the travel portal which, in effect, is the authentication domain's identity provider. After logging in, the user looks for a flight using the airline service provider. After booking a flight, the user looks for a hotel using the accommodations service provider. This time, because of the agreements established among the travel portal partners, the airline web site shares the authentication information obtained earlier in the user's online session. The user moves from the hotel reservations web site to the airline reservations web site without having to re-authenticate. All of this is transparent to the user who must initially choose to unite his local identities. The following figure illustrates the travel portal example.

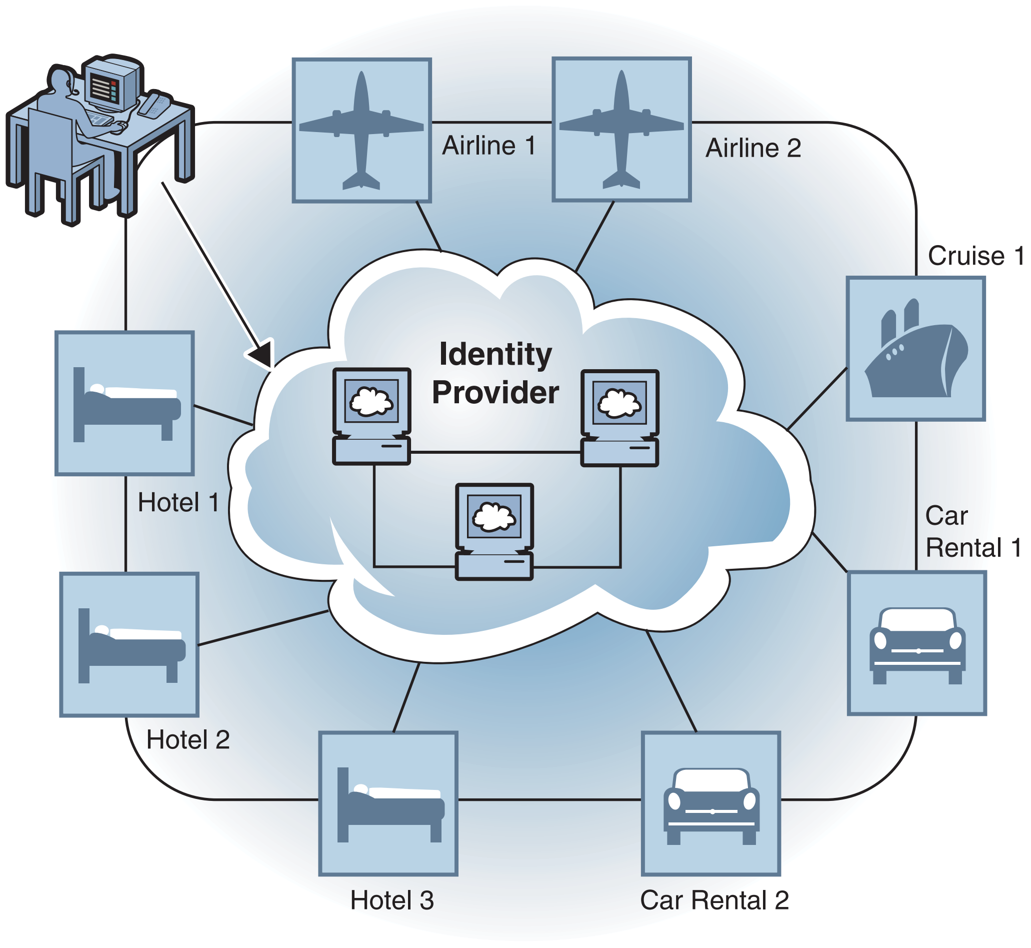


FIGURE 5–1 Federation Within a Travel Portal

Note – Account federation occurs when a user chooses to unite distinct service accounts and identity provider accounts. The user retains individual account information with each provider in the circle. At the same time, the user establishes a link that allows the exchange of authentication information between them. Users can choose to federate any or all identities they might have with the service providers. After account federation, when a user successfully authenticates with one service provider, he can access any of the his accounts within the authentication domain in a single session *without having to reauthenticate*.

The Web Services Stack

In Access Manager, the Federation framework enables the secure exchange of authentication and authorization information by providing an interface for creating, modifying, and deleting authentication domains and configuring service and identity providers (both remote and hosted types) as entities. Additionally, the implemented web services define a stack to support the Federation framework. The following figure illustrates the architecture of the web services stack and how a web service consumer communicates with the web service provider (Access Manager).

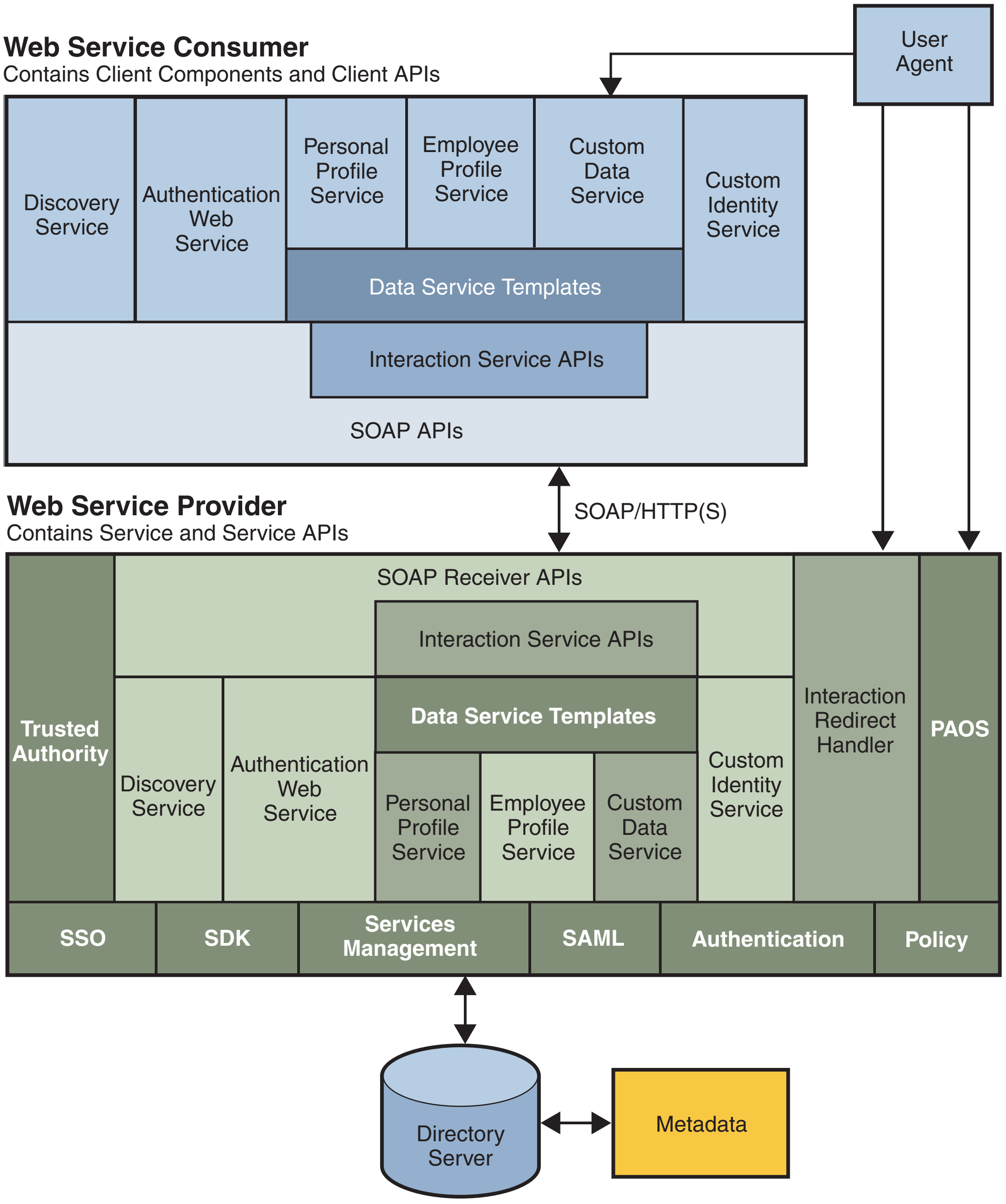


FIGURE 5–2 Web Services Architecture

Implemented Services

Access Manager includes the following web services based on the Liberty Alliance Project specifications:

Authentication Web Service
: Provides authentication to a WSC, allowing the WSC to obtain security tokens for further interactions with other services at the same provider. Upon successful authentication, the final Simple Authentication and Security Layer (SASL) response contains the resource offering for the Discovery Service.

Discovery Service
: A web service that allows a requesting entity, such as a service provider, to dynamically determine a principal's registered attribute provider. Typically, a service provider queries the Discovery Service, which responds by providing a *resource offering* that describes the requested attribute provider. The implementation of the Discovery Service includes Java and web-based interfaces.

SOAP Binding
: A set of Java APIs used by the developer of a Liberty-enabled identity service. The APIs are used to send and receive identity-based messages using SOAP, an XML-based messaging protocol.

Liberty Personal Profile Service
: A data service that supports storing and modifying a principal's identity attributes. Identity attributes might include information such as first name, last name, home address, and emergency contact information. The Liberty Personal Profile Service is queried or updated by a WSC acting on behalf of the principal.

Web Services Process

The following figure provides a high-level view of the process between the various components in the web services stack. In this example:

- The web browser represents a user.
- The service provider also acts as a web services consumer (WSC), invoking a web service on behalf of the user. The service provider relies on the identity provider for authentication.
- The identity provider acts as an authentication provider by authenticating the user. It also acts as a trusted authority, issuing security tokens through the Discovery Service.
- The web services provider (WSP) serves requests from web services clients such as the Liberty Personal Profile Service.

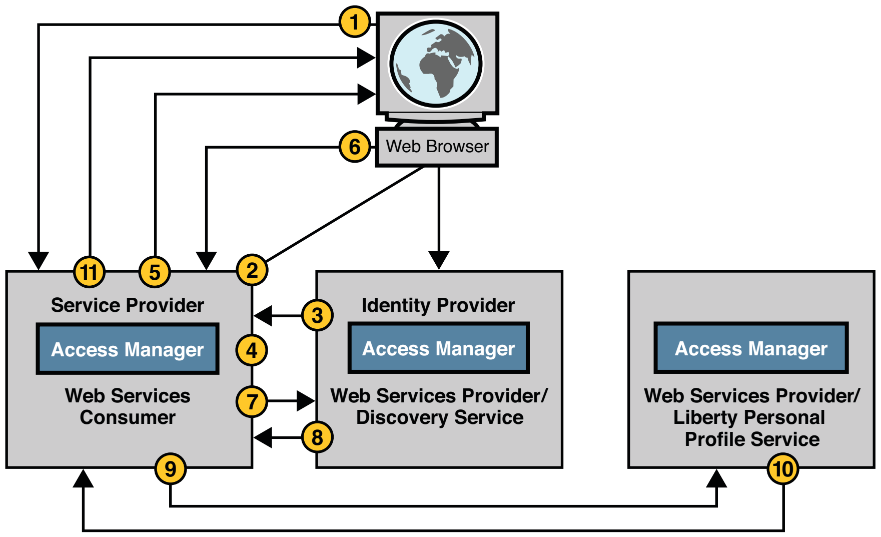


FIGURE 5–3 Web Services Stack Process

The following process assume that the user, the identity provider, and the service provider have already been federated.

1. The user attempts to access a resource hosted on the service provider server.
2. The service provider redirects the user to the identity provider for authentication.
3. The identity provider authenticates the user successfully and sends the single sign-on assertion to the requesting service provider.
4. The service provider verifies the assertion and the user is issued a session token.
5. The service provider redirects the user to the requested resource.
6. The user requests access to another service hosted on the WSC server.

   For example, it might need that value of an attribute from the user's Liberty Personal Profile Service.
7. The WSC sends a query to the Discovery Service to determine where the user's Liberty Personal Profile Service instance is hosted.

   The WSC bootstraps the Discovery Service with the resource offering from the assertion obtained earlier.
8. The Discovery Service returns a response to the WSC containing the endpoint for the user's Liberty Personal Profile Service instance and a security token that the WSC can use to access it.
9. The WSC sends a query to the Liberty Personal Profile Service instance.

The query asks for the user's personal profile attributes, such as home phone number. The required authentication mechanism specified in the Liberty Personal Profile Service resource offering must be followed.

10. The Liberty Personal Profile Service instance authenticates and validates authorization for the requested user or the WSC, or both.

    If user interaction is required for some attributes, the Interaction Service will be invoked to query the user for consents or for attribute values. The Liberty Personal Profile Service instance returns a response to the WSC after collecting all required data.
11. The WSC processes the Liberty Personal Profile Service response, and renders the service pages containing the information.

For detailed information about all these components, see the *Sun Java System Access Manager 7.1 Federation and SAML Administration Guide*.

SAML Service

SAML defines an eXtensible Markup Language (XML) framework to achieve interoperability across different vendor platforms that provide SAML assertions. SAML is an XML framework for exchanging security information over the Internet. Access Manager SAML Service consists of a web service interface, a SAML core component, and a SAML framework that web services can connect to.

The Access Manager SAML Service enables the following functionality:

- Users can authenticate against Access Manager and access trusted partner sites without having to reauthenticate. This single sign-on process independent of the process enabled by Access Manager user session management.
- Access Manager acts as a policy decision point (PDP), allowing external applications to access user authorization information for the purpose of granting or denying access to their resources.
- Access Manager acts as both an attribute authority (allowing trusted partner sites to query a subject's attributes) and an authentication authority (allowing trusted partner sites to query a subject's authentication information.)
- Two parties in different security domains can validate each other for the purpose of performing business transactions.
- Access Manager SAML APIs can be used to build Authentication, Authorization Decision and Attribute Assertions.
- The Access Manager SAML Service provides pluggable XML-based digital signature signing and verifying.

CHAPTER 6

Logging and the Java Enterprise System Monitoring Framework

Sun Java System Access Manager provides its own logging feature that records information such as user login, user logout, session creation, and policy evaluation. This chapter describes how Access Manager logging works, and provides some information about the Java Enterprise System Monitoring Framework. It contains the following sections:

Logging Overview

The Logging Service enables Access Manager services to record information such as access denials, access approvals, authentication events, and authorization violations. Administrators can use the logs to track user actions, analyze traffic patterns, audit system usage, review authorization violations, and troubleshoot. The logged information from all Access Manager services is recorded in one centralized directory. The default location for all Access Manager log files is `/var/opt/SUNWam/logs`. Logging client APIs enable external applications to access the Logging framework. This section contains the following:

Logging Service

The Logging Service stores the attributes and values for the logging function. A global service configuration file named `amLogging.xml` defines the Logging attributes. Examples of Logging Service attributes are maximum log size, log location, and log format (flat file or relational database). The attribute values are applied across the Access Manager deployment and inherited by every configured realm. By default, `amLogging.xml` is located in the directory `/etc/opt/SUNWam/config/xml` when Access Manager is installed in a Solaris environment. (When installed on Windows, the directory is *jes-install-dir*`\identity\config\xml`; on HP-UX the directory is `/etc/opt/sun/identity/config/xml`.) The structure of `amLogging.xml` is defined by file `sms.dtd`.

Logging Configuration

When Access Manager starts or when any logging configuration data is changed using the Access Manager console, the logging configuration data is loaded (or reloaded) into the Logging Service. This data includes the log message format, log file name, maximum log size, and the number of history files. Applications can use the client APIs to access the Logging features from a local or remote server. The client APIs use an XML-over-HTTP layer to send logging requests to the Logging component on the server where Access Manager is installed.

Recorded Events

The client passes the Logging Service logs information to the `com.sun.identity.log.LogRecord` class. The following table summarizes the items logged by default in the `LogRecord`.

TABLE 6–1 Events Recorded in LogRecord

Event	Description
Time	The date (`YYYY-MM-DD`) and time (`HH:MM:SS`) at which the log message was recorded. This field is not configurable.
Data	Variable data pertaining to the log records's `MESSAGE ID`. This field is not configurable.
Module Name	Name of the Access Manager service or application being logged. Additional information on the value of this field can be found in “Adding Log Data” on page 88.
Domain	Access Manager domain to which the user belongs.
Log Level	The Java 2 Platform, Standard Edition (J2SE) version 1.4 log level of the log record.
Login ID	ID of the user as the subject of the log record. The user ID is taken from the session token.
IP Address	IP address from which the operation was performed.

TABLE 6–1 Events Recorded in LogRecord *(Continued)*

Event	Description
Logged By	User who writes the log record. The information is taken from the session token passed during `logger.log(logRecord, ssoToken)`.
Host Name	Host name associated with the IP Address above.
MessageID	Non-internationalized message identifier for this log record's message.
ContextID	Identifier associated with a particular login session.

Log Files

The following sections contain information about Access Manager log files:

- "Log File Formats" on page 93
- "Error and Access Logs" on page 95

Log File Formats

Access Manager can record events in either of the following formats:

- "Flat File Format" on page 93
- "Relational Database Format" on page 93

Flat File Format

The default flat file format is the W3C Extended Log Format (ELF). Access Manager uses this format to record the default fields in each log record. See "Recorded Events" on page 92 for a list of default fields and their descriptions. The following example illustrates an authentication log record formatted for a flat file. The fields are in this order: `Time`, `Data`, `ModuleName`, `MessageID`, `Domain`, `ContextID`, `LogLevel`, `LoginID`, `IPAddr`, `LoggedBy`, and `HostName`.

EXAMPLE 6–1 Flat File Record From `amAuthentication.access`

```
"2005-08-01 16:20:28"   "Login Success" LDAP    AUTHENTICATION-100
   dc=example,dc=com       e7aac4e717dda1bd01      INFO
uid=amAdmin,ou=People,dc=example,dc=com 192.18.187.152
"cn=exampleuser,ou=Example Users,dc=example,dc=com" exampleHost
```

Relational Database Format

When Access Manager uses a relational database to log messages, the messages are stored in a database table. Access Manager uses Java Database Connectivity (JDBC) to access the database table. JDBC provides connectivity to a wide range of SQL databases. JDBC also provides access to other tabular data sources such as spreadsheets or flat files. Oracle® and MySQL databases are currently supported.

For log records generated by Access Manager, the `Data` and `MessageID` fields are used slightly differently than in previous versions of Access Manager. Starting with this version of Access Manager, the `MessageID` field is introduced as a template for types of log messages. For example, in previous versions, Access Manager would generate the following message in the `Data` field:

```
Data: "Created group
cn=agroupSubscription1,ou=Groups,dc=iplanet,dc=com"
```

In this version of Access Manager, two log records are recorded for the one event:

```
Data:       agroupSubscription1|group|/
MessageID:    CONSOLE-1
```

and

```
Data:       agroupSubscription1|group|/
MessageID:    CONSOLE-2
```

These log records reflect the use of identities and realms. In this example, `CONSOLE-1` indicates an attempt to create an identity object, and `CONSOLE-2` indicates the attempt to create an identity object was successful. The root organization notation (`dc=iplanet,dc=com`) is replaced with a forward slash (`/`). The variable parts of the messages (*agroupSubscription1*, *group*, and */*) are separated by a pipe character (|), and continue to go into the `Data` field of each log record. The `MessagID` string is not internationalized in order to facilitate machine-readable analysis of the log records in any locale.

The following table summarizes the schema for a relational database.

TABLE 6–2 Relational Database Log Format

Column Name	Data Type	Description
`TIME`	VARCHAR(30)	Date of the log in the format `YYYY-MM-DD HH:MM:SS`.
`DATA`	VARCHAR(1024)	The variable data part of the log record pertaining to the MESSAGE ID. For MySQL, the Data Type is VARCHAR(255).
`MODULENAME`	VARCHAR(255)	Name of the Access Manager component invoking the log record.
`DOMAIN`	VARCHAR(255)	Access Manager domain of the user.
`LOGLEVEL`	VARCHAR(255)	JDK 1.4 log level of the log record.

TABLE 6–2 Relational Database Log Format *(Continued)*

Column Name	Data Type	Description
`LOGINID`	VARCHAR(255)	Login ID of the user who performed the logged operation.
`IPADDR`	VARCHAR(255)	IP Address of the machine from which the logged operation was performed.
`LOGGEDBY`	VARCHAR(255)	Login ID of the user who writes the log record.
`HOSTNAME`	VARCHAR(255)	Host name of machine from which the logged operation was performed.
`MESSAGE ID`	VARCHAR(255)	Non-internationalized message identifier for this log record's message.
`CONTEXT ID`	VARCHAR(255)	Identifier associated with a particular login session.

Error and Access Logs

There are two types of Access Manager log files:

- Access log files
- Error log files

Access log files record general auditing information concerning the Access Manager deployment. An access log may contain a single record for an event such as a successful authentication, or multiple records for the same event. For example, when an administrator uses the console to change an attribute value, the Logging Service logs the attempt to change in one record but, the Logging Service also logs the results of the execution of the change in a second record. *Error log files* record errors that occur within the application. While an operation error is recorded in the error log, the operation attempt is recorded in the access log file.

Flat log files are appended with the `.error` or `.access` extension. Database column names end with `_ERROR` or `_ACCESS`. For example, a flat file logging console events is named `amConsole.access` while a database column logging the same events is named `AMCONSOLE_ACCESS` or `amConsole_access`.

Note – The period (.) separator in a log filename is converted to an underscore (_) in database formats. Also in databases, table names may be converted to all upper case. For example, `amConsole.access` may be converted to `AMCONSOLE_ACCESS`, or it may be converted to `amConsole_access`.

Access Manager Component Logs

The log files record a number of events for each of the Access Manager components using the Logging Service. Administrators typically review these log files on a regular basis. The default location for all Access Manager log files is `/var/opt/SUNWam/logs` when Access Manager is installed in a Solaris environment. (When installed on Windows, the directory is *jes-install-dir*`\identity\logs`; on HP-UX the directory is `/var/opt/sun/identity/logs`.) The following table provides a brief description of the log files produced by each Access Manager component.

TABLE 6–3 Access Manager Component Logs

Component	Log Filename	Information Logged
Session	■ `amSSO.access`	Session management attributes values such as login time, logout time, and time out limits. Also session creations and terminations.
Administration Console	■ `amConsole.access` ■ `amConsole.error`	User actions performed through the administration console such as creation, deletion and modification of identity-related objects, realms, and policies. `amConsole.access` logs successful console events while `amConsole.error` logs error events.
Authentication	■ `amAuthentication.access` ■ `amAuthentication.error`	User logins and log outs, both successful and failed.
Federation	■ `amFederation.access` ■ `amFederation.error` ■ `amLiberty.access` ■ `amLiberty.error`	Federation-related events such as the creation of an authentication domain or the creation of a hosted provider entity.
Authorization (Policy)	■ `amPolicy.access` ■ `amPolicy.error` ■ `amAuthLog`	Policy-related events such as policy creation, deletion, or modification, and policy evaluation. `amPolicy.access` logs policy allows, `amPolicy.error` logs policy error events, and `amAuthLog` logs policy denies.
Policy Agent	`amAgent`	Exceptions regarding resources that were either accessed by a user or denied access to a user. `amAgent` logs reside on the server where the policy agent is installed. Agent events are logged on the Access Manager machine in the Authentication logs.
SAML	■ `amSAML.access` ■ `amSAML.error`	SAML-related events such as assertion and artifact creation or removal, response and request details, and SOAP errors.

TABLE 6–3 Access Manager Component Logs *(Continued)*

Component	Log Filename	Information Logged
Command-line	■ `amAdmin.access` ■ `amAdmin.error`	Event successes and errors that occur during operations using the command line tools. Examples are: loading a service schema, creating policy, and deleting users.
Password Reset	■ `amPasswordReset.access`	Password reset events.

For detailed reference information about events recorded in each type of Access Manager log, see the *Sun Java System Access Manager 7.1 Administration Guide*.

Additional Logging Features

You can enable a number of logging features for added functionality. The additional features include:

- "Secure Logging" on page 97
- "Remote Logging" on page 97
- "Log Reading" on page 98

Secure Logging

This feature adds an extra measure of security to the Logging Service. When secure logging is enabled, the Logging component can detect unauthorized changes to the security logs. No special coding is required to leverage this feature. However, secure logging uses a certificate that you must create and install in the container that runs Access Manager. When secure logging is enabled, a Manifest Analysis and Certification (MAC) is generated and stored for every log record, and a special signature record is periodically inserted in the log. The signature record represents the signature for the contents of the log written up to that point. The combination of the certificate and the signature record ensures that the logs have not been tampered. For detailed information about enabling secure logging, see the *Sun Java System Access Manager 7.1 Administration Guide*.

Remote Logging

Remote logging allows a client using the Client APIs to create log records on an instance of Access Manager deployed on a remote machine. Remote logging is useful in the following situations:

- When the login URL in the Naming Service of an Access Manager instance points to a remote Access Manager instance, and a trust relationship between the two instances has been configured.

- When the Access Manager APIs are installed in a remote Access Manager instance, and a client application or a simple Java class running on the Access Manager server uses the logging APIs.
- When logging APIs are used by Access Manager agents.

Log Reading

Access Manager provides Logging APIs for writing your own custom log reading program. You can set up queries to retrieve specific records from the log file or database. This is useful for auditing purposes. For more information, see the *Sun Java System Access Manager 7.1 Developer's Guide*.

Java Enterprise System Monitoring Framework

Access Manager 7.1 integrates with the Java Enterprise System (JES) monitoring framework through Java Management Extensions (JMX). JMX technology provides the tools for building distributed, web-based, modular, and dynamic solutions for managing and monitoring devices, applications, and service-driven networks. Typical uses of the JMX technology include: consulting and changing application configuration, accumulating statistics about application behavior, notification of state changes and erroneous behaviors. Data is delivered to centralized monitoring console. Access Manager 7.1 uses the Java ES Monitoring Framework to capture statistics and service-related data such as:

- Number of attempted, successful, and failed authentications
- Number of active sessions, statistics from session failover DB
- Session failover database statistics
- Policy caching statistics
- Policy evaluation transaction times
- Number of assertions for a given provider in a SAML/Federation deployment

For comprehensive information about how the JES monitoring framework works and how you can use the monitoring framework with Access Manager, see the *Sun Java Enterprise System 5 Monitoring Guide*.

Index