Sun Federated Access Manager 8.0 Technical Overview

Beta

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

Part No: 820–3740–10
September 2008

080704@20490

List of Remarks

Contents

Preface

Sun Federated Access Manager is an access management product that includes a set of software components to provide the authentication and authorization services needed to support enterprise applications distributed across a network or Internet environment. This book, *Sun Federated Access Manager 8.0 Technical Overview*, describes the features of Federated Access Manager, explains what it does, and illustrates how it 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 servers and software. Readers of this guide should be familiar with the following:

- Web containers in which Federated 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:

- "Federated Access Manager Core Documentation" on page 10
- "Adjunct Product Documentation" on page 11

Federated Access Manager Core Documentation

The Federated Access Manager core documentation set contains the following titles:

- The *Sun Federated Access Manager 8.0 Early Access (EA) 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 Federated Access Manager 8.0 Technical Overview* (this guide) provides an overview of how Federated Access Manager components work together to protect enterprise assets and web-based applications. It also explains basic concepts and terminology.
- The XXXXX provides planning and deployment solutions for Sun Java System Access Manager based on the solution life cycle
- The *Sun Federated Access Manager 8.0 Installation and Configuration Guide* provides information for installing and configuring Federated Access Manager.
- The *Sun Java System Access Manager 7.1 Performance Tuning and Troubleshooting Guide* provides information on how to tune Access Manager and its related components for optimal performance.
- The *Sun Federated Access Manager 8.0 Administration Guide* describes administrative tasks such as *how to create a realm* and *how to configure a policy*. Most of the tasks described can be performed using the administration console as well as the `famadm` command line utilities.
- The *Sun Federated Access Manager Administration Reference* is a look-up guide containing information about the command line interfaces, configuration attributes, internal files, and error codes.
- The *Sun Federated Access Manager 8.0 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 Federated Access Manager 8.0 C API Reference* provides summaries of data types, structures, and functions that make up the public Access Manager C APIs.
- The *Federated Access Manager 8.0 Java API Reference* provides information about the implementation of Java packages in Access Manager.

- The *Sun Java System Federated Access Manager Policy Agent 3.0 User's Guide* provides an overview of the policy functionality and the policy agents available for Federated Access Manager.

Updates to the *Release Notes* and links to modifications of the core documentation can be found on the Federated Access Manager page at docs.sun.com. Updated documents will be marked with a revision date.

Adjunct 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

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–1 Typographic Conventions

Typeface	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`

`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`**
		`Password:`

AaBbCc123	A placeholder to be replaced with a real name or value	The command to remove a file is `rm` *filename*.

TABLE P–1 Typographic Conventions *(Continued)*

Typeface	Meaning	Example
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–2 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–3 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.

TABLE P–3 Symbol Conventions *(Continued)*

Symbol	Description	Example	Meaning
+	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.

PART I

An Overview of Federated Access Manager

This part of the Sun Federated Access Manager Technical Overview contains introductory material concerning Federated Access Manager. It includes the following sections:

- Chapter 1, “Introducing Federated Access Manager”
- Chapter 2, “Examining Federated Access Manager”
- Chapter 3, “Simplifying Federated Access Manager”

CHAPTER 1

Introducing Federated Access Manager

Sun Java™ System Federated Access Manager integrates authentication and authorization services, single sign-on, and open, standards-based federation protocols to provide a comprehensive solution for protecting network resources by preventing unauthorized access to web services, applications and web content, and securing identity data. This introductory chapter contains a high-level description of Federated Access Manager and what it does. It contains the following sections:

- “What is Federated Access Manager?” on page 17
- “What Does Federated Access Manager Do?” on page 18
- “What Are the Functions of Federated Access Manager?” on page 19
- “What Else Does Federated Access Manager Offer?” on page 21

What is Federated Access Manager?

Sun Java System Federated Access Manager is a single product that combines the features of Sun Java System Access Manager, Sun Java System Federation Manager, and the Sun Java System SAML v2 Plug-in for Federation Services; additionally, it is enhanced with new functionality developed specifically for this release. Federated Access Manager provides access management by allowing the implementation of authentication, policy-based authorization, federation, SSO, and web services security from a single, unified framework. The core application is delivered as a simple web archive (WAR) that can be easily deployed in a supported web container.

Note – Federated Access Manager is Sun Microsystems' commercial distribution of the open source code available at OpenSSO.

To assist the core application, policy agents, the Client SDK, and (possibly) other disparate pieces must be installed remotely and be able to communicate with the Federated Access

Manager server. See "What Does Federated Access Manager Do?" on page 18 for a high-level explanation of the deployment architecture and Chapter 2, "Examining Federated Access Manager," for more specific information.

What Does Federated Access Manager Do?

The following types of interactions occur daily in a corporate environment.

- An employee looks up a colleague's phone number in the corporate phone directory.
- A manager retrieves employee salary histories to 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 a company's web site and looks for a product in their online catalog.
- A vendor submits an invoice to the company's accounting department.
- A corporate human resources administrator accesses an outsourced benefits application.

For each of these transactions, the company must determine who is allowed to view the information or use the application. Some information such as product descriptions and advertising can be made available to everyone in a public online catalog. Other information such as accounting and human resources data must be restricted to employees only. And other sensitive information such as pricing models and employee insurance plans is appropriate to share only with partners, suppliers, and employees. This need for access determination is met by Sun Java System Federated Access Manager, an access management product with authentication, authorization, and single sign-on (SSO) services provided out of the box.

When a user or an external application requests access to content stored on a company's server, a *policy agent* (available in a separate download and installed on the same machine as the resource you want to protect) intercepts the request and directs it to Federated Access Manager which, in turn, requests credentials (such as a username and password in the case of a user) for authentication. If the credentials returned match those stored in the appropriate identity data store, Federated Access Manager determines that the user is authentic. Following authentication, access to the requested content is determined by the policy agent which evaluates the policies associated with the authenticated identity. Policies are created using Federated Access Manager and identify which identities are allowed to access a particular resource, specifying the conditions under which this authorization is valid. Based upon the results of the policy evaluation, the policy agent either grants or denies the user access. Figure 1–1 illustrates a high-level deployment architecture of Federated Access Manager.

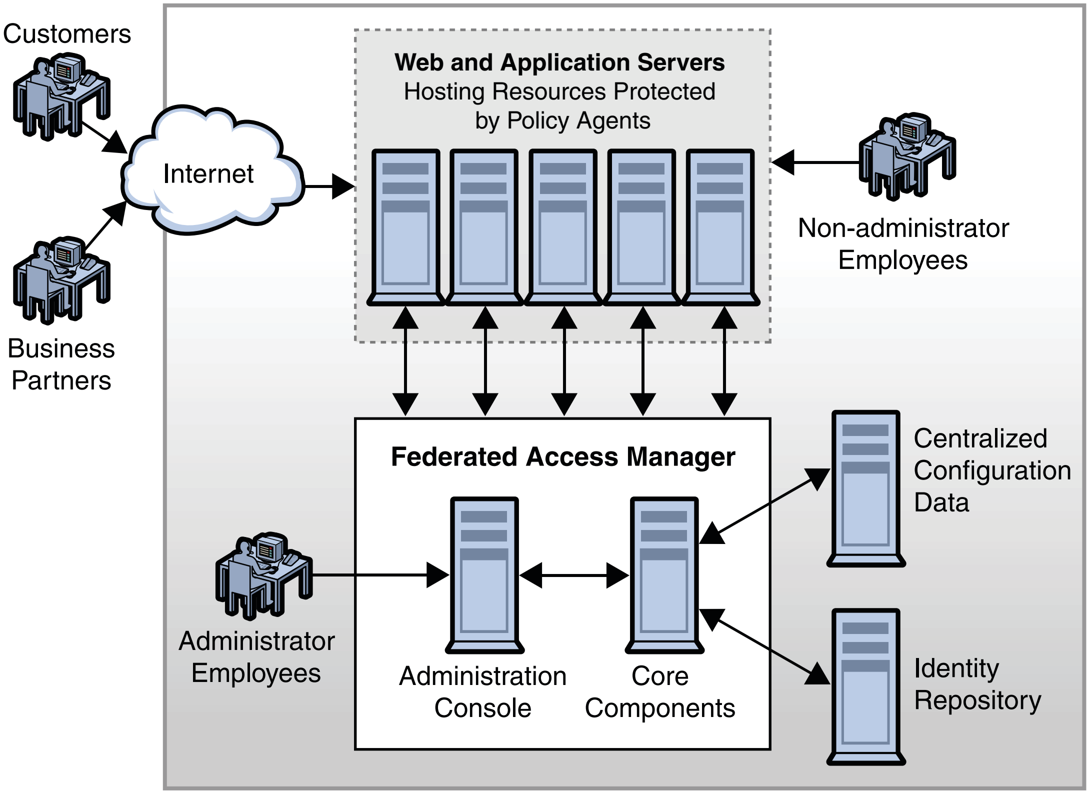


FIGURE 1–1 High-level Deployment Architecture of Federated Access Manager

What Are the Functions of Federated Access Manager?

The following sections contain an overview of the functions of Federated Access Manager.

- "Access Control" on page 19
- "Federation Management" on page 20
- "Web Services Security" on page 20
- "Identity Services" on page 21

Access Control

Federated Access Manager manages authorized access to network services and resources. By implementing authentication and authorization, Federated Access Manager (along with an installed policy agent) ensures that access to protected resources is restricted to authorized users. In a nutshell, a policy agent intercepts a request for access to a resource and communicates with Federated Access Manager to authenticate the requestor. If the user is successfully authenticated, the policy agent then evaluates the policies associated with the requested resource and the user to determine if the authenticated user is authorized to access the resource. If the user is authorized, the policy agent allows access to the resource, also providing identity data to the resource to personalize the interaction. For more information on access control, see "Core Services" on page 29 and Part II.

Federation Management

With the introduction of federation protocols into the process of access management, identity information and entitlements can be communicated across security domains, spanning multiple trusted partners. By configuring a *circle of trust* and defining applications and services as *entity providers* in the circle (either identity providers or service providers), users can opt to associate, connect or bind the various identities they have configured locally for these providers. The linked local identities are federated and allow the user to log in to one identity provider site and click through to an affiliated service provider site without having to reauthenticate; in effect, single sign-on (SSO). Federated Access Manager supports several open federation technologies including the Security Access Markup Language (SAML) versions 1 and 2, WS-Federation, and the Liberty Alliance Project Identity Federation Framework (Liberty ID-FF), therefore encouraging an interoperable infrastructure among providers. For more information on federation management, see "Core Services" on page 29 and Part III.

Web Services Security

A *web service* is a component service or application that exposes some type of business or infrastructure functionality through a language-neutral and platform-independent, callable interface; enterprises might use this web service to build larger service-oriented architectures. In particular, the service defines its interface (for example, the format of the message being exchanged) using the Web Services Description Language (WSDL), and communicates using SOAP and eXtensible Markup Language (XML) messages. The web service client (WSC) communicates with the web service provider (WSP) through an intermediary — usually a firewall or load balancer.

Although web services enable open, flexible, and adaptive interfaces, its openness creates security risks. Without proper security protections, a web service can expose vulnerabilities that might have dire consequences. Hence, ensuring the integrity, confidentiality and security of web services through the application of a comprehensive security model is critical for both enterprises and consumers. A successful security model associates identity data with the web services and creates secure service-to-service interactions. The security model adopted by Federated Access Manager identifies the user and preserves that identity through multiple interactions, maintains privacy and data integrity, uses existing technologies, and logs the interactions. In Federated Access Manager, the following web service security standards are implemented:

- Liberty Alliance Project Identity Web Services Framework (Liberty ID-WSF)

- WS-I Basic Security Profile
- WS-Trust (from which the Security Token Service was developed)

For more information on Federated Access Manager web services and web services security, see "Core Services" on page 29 and Part IV.

Identity Services

For some time, Federated Access Manager has provided client interfaces for access to core features and functionality. These interfaces are used by policy agents and custom applications developed by customers. With the recent advancements in web services though (service-oriented architecture [SOA] style and the Representational State Transfer [REST] style), Federated Access Manager now exposes these back-end functions as simple identity services allowing developers to easily invoke them when developing applications using one of the supported integrated development environment (IDE) products. (The IDE generates the stub code that wraps a call to the web service.) The Identity Services solution allows the client interfaces to be exposed using:

- Service-oriented Architecture (SOA) using SOAP and Web Services Description Language (WSDL)
- Representational State Transfer (REST) style using Web Application Description Language (WADL)

Identity Services do not require the deployment of an agent or a proxy and includes the following capabilities:

- Authentication to validate user credentials.
- Authorization to permit access to protected resources.
- Provisioning for user attribute management and self registration.
- Logging to keep track of it all.

For more information on identity services, see "Identity Services" on page 46 and Part IV.

What Else Does Federated Access Manager Offer?

Federated Access Manager allows for:

1. **Ease of Deployment:** Federated Access Manager is delivered as a WAR that can be easily deployed as a Java EE application in different web containers. All configuration files and required libraries are inside the WAR to avoid the manipulation of the classpath in the web container's configuration file. The Federated Access Manager WAR is supported on:
    - Sun Java System Web Server 7.0 — Update 1 &2
    - Sun Java System Application Server 9.1 (and Glassfish v2)

- BEA WebLogic Application Server 9.2 &10
- IBM WebSphere Application Server 6.1
- Oracle Application Server 10g
- JBoss 4.2.x
- Tomcat 5.5.x & 6.x
- Geronimo (supported on the Sun Solaris™ 10 Operating Environment for SPARC, x86 & x64 and the Sun Solaris 9 Operating Environment for SPARC & x86 systems only)

> **Note –** Geronimo can install Jetty Application Server and Tomcat Application Server; Federated Access Manager supports only Tomcat.

See the *Sun Federated Access Manager 8.0 Early Access (EA) Release Notes* for updates to this list.

2. **Portability:** Federated Access Manager is supported on the following operating systems:
   - Sun Solaris 10 Operating Environment for SPARC, x86 & x64 systems
   - Sun Solaris 9 Operating Environment for SPARC & x86 systems
   - Windows Server 2003 and Windows XP (development only) operating systems
   - Red Hat Enterprise Linux 4 Server (Base)
   - Red Hat Enterprise Linux 4 Advanced Platform
   - Red Hat Enterprise Linux 5 Server (Base)
   - Red Hat Enterprise Linux 5 Advanced Platform
   - Windows 2003 Standard Server
   - Windows 2003 Enterprise Server
   - Windows 2003 Datacenter Server
   - Windows Vista
   - IBM AIX 5.3 (supported with the IBM WebSphere Application Server 6.1 container only)

   See the *Sun Federated Access Manager 8.0 Early Access (EA) Release Notes* for updates to this list.
3. **Open Standards:** Federated Access Manager is built using open standards and specifications as far as possible. For example, features designed for federation management and web services security are based on the Security Assertion Markup Language (SAML), the Liberty Alliance Project specifications, and the WS-Security standards.

4. **Ease of Administration:** Federated Access Manager contains a web-based, graphical administration console as well as command line interfaces for configuration tasks and administrative operations. Additionally, an embedded, centralized data store allows for one place to store server and agent configuration data.
5. **Security:**
   - Runtime security enables an enterprise's resources to be protected as configured and Federated Access Manager services to be accessed by authorized entities only.
   - Administration security ensures only authorized updates are made to the Federated Access Manager configuration data.
   - Deployment security implements best practices for installing Federated Access Manager on different operating systems, web containers, and so forth.

   Additionally, all security actions are logged.
6. **Configuration Data Store:** Federated Access Manager can write server configuration data to an embedded configuration data store. You can also point to instances of Sun Java System Directory Server 5.1, 5.2 & 6.2 during configuration of Federated Access Manager for use as a configuration data store. See “Data and Data Stores” on page 52 for more information.
7. **User Data Store Independence:** Federated Access Manager allows you to view and retrieve user information without making changes to an existing user database. Supported directory servers include Directory Server 5.1, 5.2 & 6.2, IBM Tivoli Directory 6.1, and Microsoft Active Directory 2003. See “Data and Data Stores” on page 52 for more information.

Caution – The configuration data store embedded with Federated Access Manager should only be used as a user data store for proof of concepts and deployments in development.

8. **Web and Non-Web-Based Resources:** The core design of Federated Access Manager caters to SSO for both web and non-web applications.
9. **Performance, Scalability and Availability:** Federated Access Manager can be scaled horizontally and vertically to handle increased workloads, and as security needs change over time. There is no single point of failure.
10. **Distributed Architecture** Server and client components can be deployed across the enterprise or across domain boundaries as all application programming interfaces (API) provide remote access to Federated Access Manager based on a service-oriented architecture.
11. **Flexibility and Extensibility:** Many Federated Access Manager services expose a service provider interface (SPI) allowing expansion of the framework to provide for specific deployment needs.
12. **Internationalization** Federated Access Manager contains a framework for multiple language support. Customer facing messages, API, command line interfaces, and user interfaces are localized in the supported languages.

CHAPTER 2

Examining Federated Access Manager

Federated Access Manager provides a pluggable architecture to deliver access management, secure web services, and federation capabilities. This chapter contains information on the internal architecture and features of Federated Access Manager.

Federated Access Manager Architecture

Federated Access Manager is written in Java, and leverages many industry standards, including the HyperText Transfer Protocol (HTTP), the eXtensible Markup Language (XML), the Security Assertion Markup Language (SAML), and SOAP, to deliver access management, secure web services, and federation capabilities in a single deployment. It consists of client application programming interfaces (a Client SDK), a framework of services that implement the business logic, and service provider interfaces (SPI) that are implemented by concrete classes and can be used to extend the functionality of Federated Access Manager as well as retrieve information from data stores. Figure 2–1 illustrates the internal architecture of Federated Access Manager.

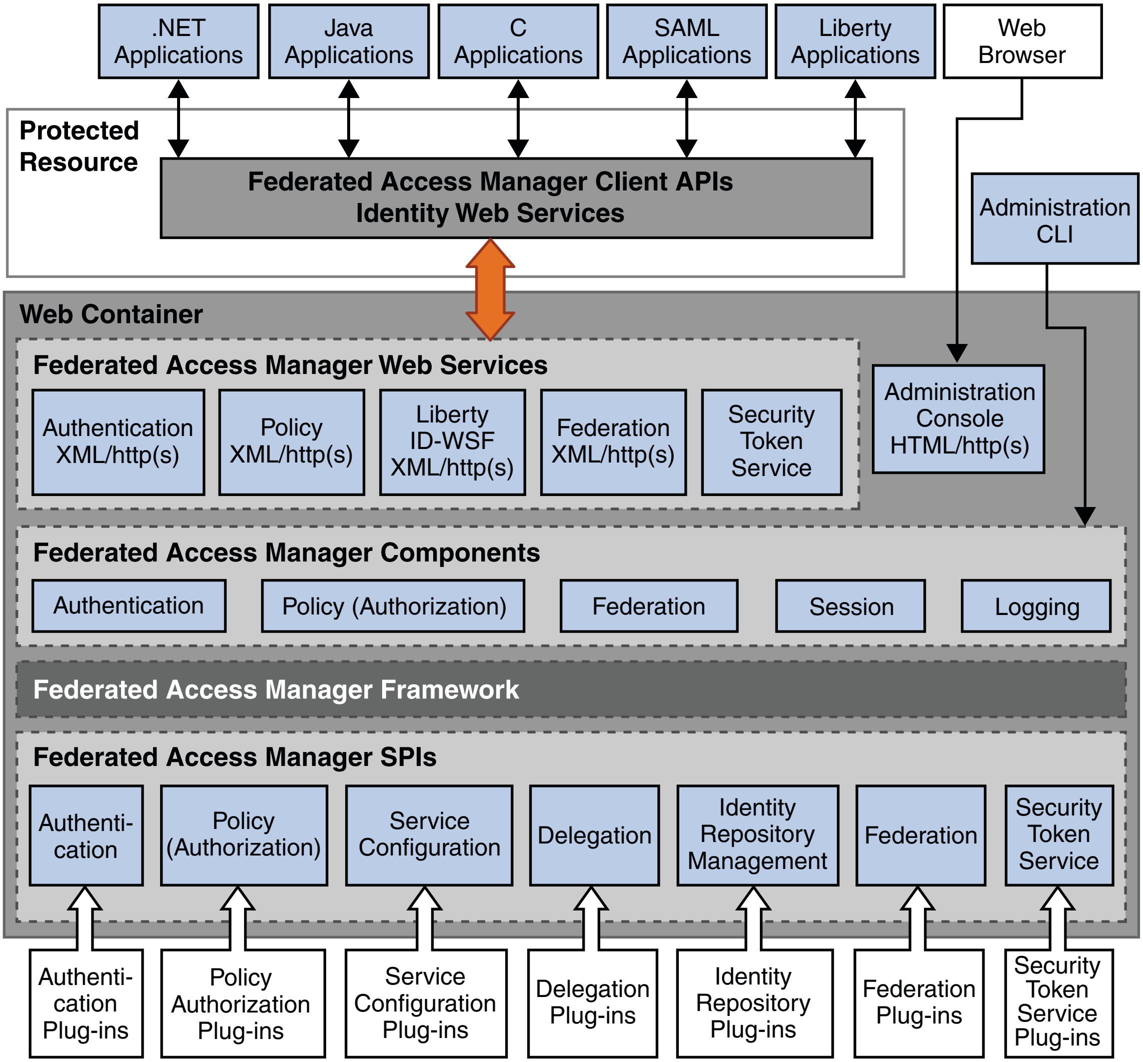


FIGURE 2–1 Internal Architecture of Federated Access Manager

Each component of Federated Access Manager uses its own framework to retrieve customer data from the plug-in layer and to provide data to other components. The Federated Access Manager framework integrates all of the application logic into one layer that is accessible to all components and plug-ins. The Client SDK and Identity Web Services are installed on a machine remote to the Federated Access Manager server that holds the resource to be protected. (The policy agent, also installed on the remote machine, is basically a client written using the Client SDK and Identity Web Services.) Applications on the remote machine access Federated Access Manager using the Client SDK. Custom plug-in modules are installed on the machine local to Federated Access Manager and interact with the Federated Access Manager SPI to retrieve required information from the appropriate data store and deliver it to the plug-ins and, in turn, the Federated Access Manager framework for processing.

How Federated Access Manager Works

To gain access to a protected resource, the requestor needs to be authenticated and have the authorization to access the resource. When someone (using a browser) sends an HTTP request for access to a protected resource, a policy agent (separately downloaded and installed on the same machine as the resource you want to protect) intercepts the request and examines it. If no valid Federated Access Manager session token is found, the policy agent contacts the server which then invoke the authentication and authorization processes. Figure 2–2 illustrates one way in which the policy agents can be situated to protect an enterprise's servers by directing HTTP requests to a centralized Federated Access Manager for processing.

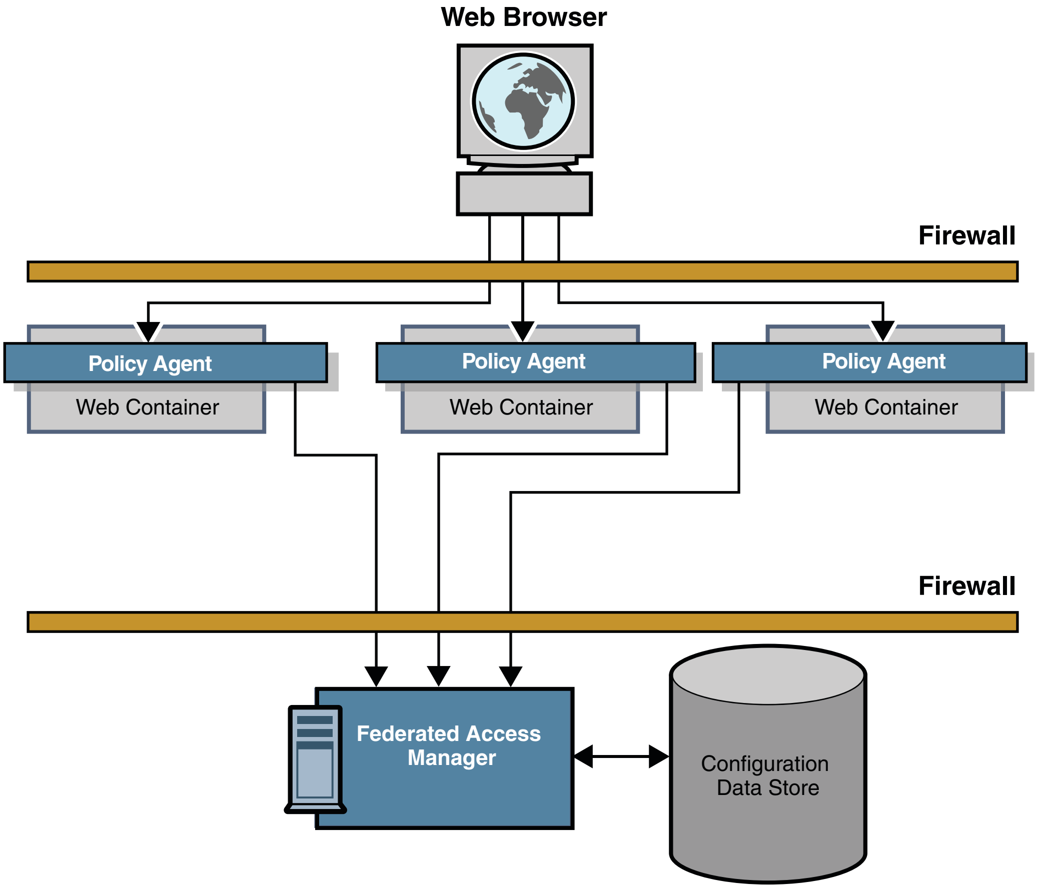


FIGURE 2–2 Basic Federated Access Manager Deployment

Federated Access Manager integrates core features such as access control through authentication and authorization processes, and federation. These functions can be configured using the administration console or the `famadm` command line utility. Figure 2–3 is a high-level illustration of the interactions that occur between Federated Access Manager, a policy agent, browser, and protected application during authentication and authorization.

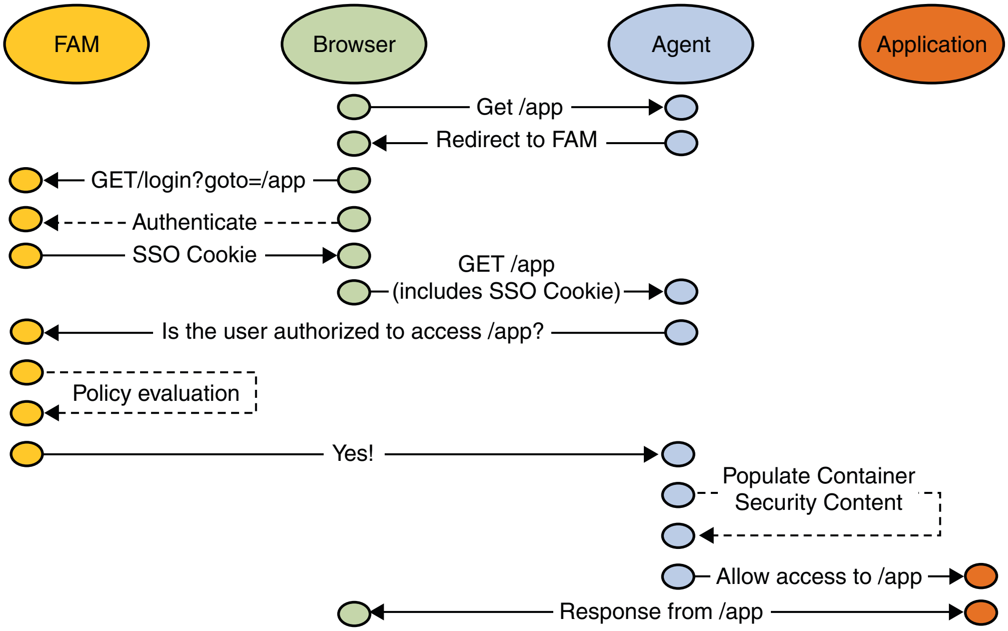


FIGURE 2–3 Federated Access Manager Authentication and Authorization Interactions

For more information on the core functions of Federated Access Manager, see "Core Services" on page 29.

Core Services

Services developed for Federated Access Manager generally contain both a server component and a client component. The server component is a simple Java servlet developed to receive XML requests and return XML responses. (The deployment descriptor `web.xml` defines the servlet name and description, the servlet class, initialization parameters, mappings, and other startup information.) The client component is provided as Java application programming interfaces (API), and in some cases C API, that allow remote applications and other Federated Access Manager services to communicate with and consume the particular functionality.

Each core service uses its own framework to retrieve customer and service data and to provide it to other Federated Access Manager services. The Federated Access Manager framework integrates all of these service frameworks to form a layer that is accessible to all product components and plug-ins. The following sections contain information on the Federated Access Manager core services.

- "Authentication Service" on page 30
- "Policy Service" on page 33
- "Session Service" on page 35

Note – Many services also provide a public SPI that allows the service to be extended. See the *Sun Federated Access Manager 8.0 Developer's Guide*, the *Sun Federated Access Manager 8.0 C API Reference*, and the *Federated Access Manager 8.0 Java API Reference* for information.

Authentication Service

The Authentication Service provides the functionality to request user credentials and validate them against a specified authentication data store. Upon successful authentication, it creates a session data structure for the user that can be validated across all web applications participating in an SSO environment. Several authentication modules are supplied with Federated Access Manager, and new modules can be plugged-in using the Java Authentication and Authorization Service (JAAS) SPI.

Note – The Authentication Service is based on the JAAS specification, a set of API that enables services to authenticate and enforce access controls upon users. See the Java Authentication and Authorization Service Reference Guide for more information.

Components of the Authentication Service include:

- The Distributed Authentication User Interface allows the Authentication Service user interface to be deployed separately from Federated Access Manager, if desired. By deploying this authentication proxy in the DMZ and using the authentication interfaces provided in the Client SDK to pass user credentials back and forth, you can protect Federated Access Manager data (for example, the login URL information and hence the host information). JavaServer Pages (JSP) represent the interface displayed to users for authentication and are completely customizable.
- The Core Authentication Service executes common processes across all authentication modules. Key responsibilities of this service include identification of the appropriate plan to authenticate the user (identify the authentication module, load the appropriate JSP) and creation of the appropriate session for the authenticated user.
- The Authentication API are *remoteable* interfaces that don't need to reside on the same machine as the Federated Access Manager server. This allows remote clients to access the Authentication Service. remote-auth.dtd defines the structure for the XML communications that will be used by the Authentication Service, providing definitions to initiate the process, collect credentials and perform authentication.

- A number of authentication modules are installed and configured (including, but not limited to, LDAP, Unix, Windows Desktop, Certificate, and Active Directory). A configured authentication level for each module is globally defined. Mechanisms are also provided to upgrade a user's session after authenticating the user to an additional authentication module that satisfies the authentication level of the resource. New modules can be plugged-in using the JAAS SPI.

The Authentication Service interacts with both the database that stores user credentials (authentication data store) to validate the user, and with the Identity Repository Service 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*. Figure 2–4 illustrates how the authentication components interact within the Federated Access Manager framework.

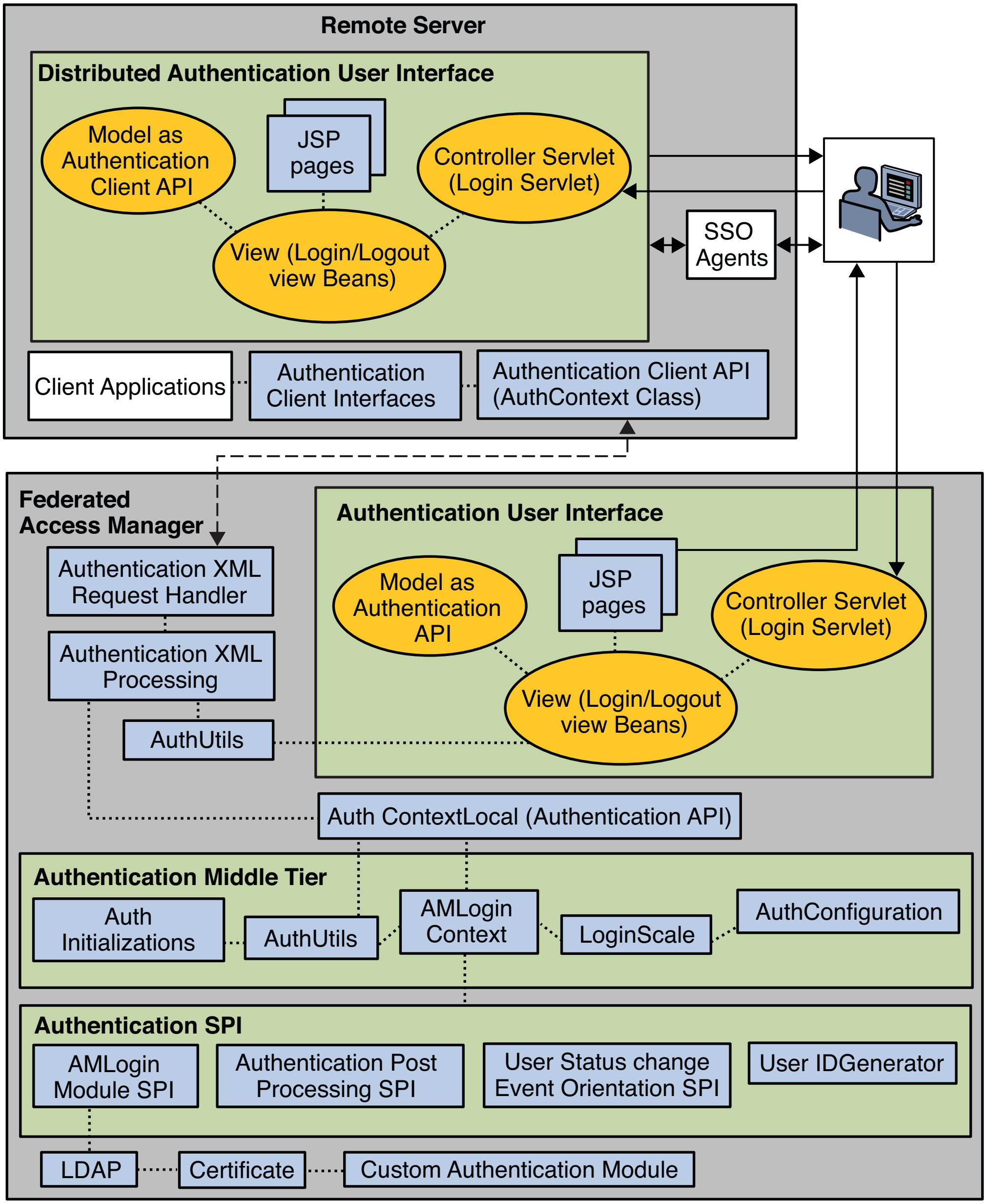

Remote Server
Distributed Authentication User Interface
Model as Authentication Client API
JSP pages
Controller Servlet (Login Servlet)
View (Login/Logout view Beans)
SSO Agents
Client Applications
Authentication Client Interfaces
Authentication Client API (AuthContext Class)
Federated Access Manager
Authentication XML Request Handler
Authentication XML Processing
AuthUtils
Authentication User Interface
Model as Authentication API
JSP pages
Controller Servlet (Login Servlet)
View (Login/Logout view Beans)
Auth ContextLocal (Authentication API)
Authentication Middle Tier
Auth Initializations
AuthUtils
AMLogin Context
LoginScale
AuthConfiguration
Authentication SPI
AMLogin Module SPI
Authentication Post Processing SPI
User Status change Event Orientation SPI
User IDGenerator
LDAP
Certificate
Custom Authentication Module

FIGURE 2–4 Authentication Components Within the Authentication Service Framework

More information on the architecture of the Authentication Service can be found in the Authentication Service Architecture document on the OpenSSO web site.

Policy Service

Authorization is the process with which Federated Access Manager evaluates the policies associated with an authenticated user’s identity, and determines whether the user has permission to access a protected resource. (A *policy* defines the rules that specify a user's access privileges to a protected resource.) The Policy Service provides the authorization functionality using a rules-based engine. It interacts with the Federated Access Manager configuration data store, a delegation plug-in (which helps to determine a network administrator’s scope of privileges), and Identity Repository Service plug-ins to verify that the user has access privileges from a recognized authority. Policy can be configured using the administration console, and comprises the following:

- A *Schema* for the policy type (normal or referral) that describes the syntax of policy.
- A *Rule* which defines the policy itself and is made up of a *Resource*, an *Action* and a *Value*.
- *Condition(s)* to define constraints on the policy.
- *Subject(s)* to define the user or collection of users which the policy affects.
- A *ResponseProvider(s)* to send requested attribute values, typically based on the user profile, with the policy decision.

Figure 2–5 illustrates the framework of the Policy Service. Note that the `PolicyServiceRequestHandler` maps to the `PolicyRequest` XML element.

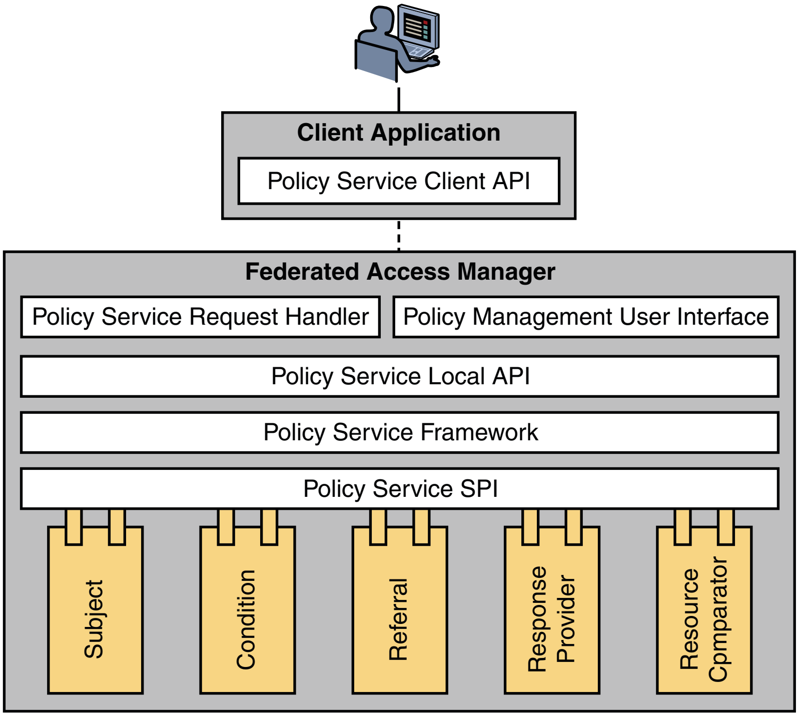


FIGURE 2–5 Policy Components within the Policy Service Framework

Policy agents are an integral part of authorization. They are programs, available for installation separate from Federated Access Manager, that police the web container which hosts the protected resources. When a user requests access to the protected resource (such as a server or an application), the policy agent intercepts the request and redirects it to the Federated Access Manager Authentication Service. Following authentication, the policy agent will enforce the authenticated user's assigned policies. Federated Access Manager supports two types of policy agents:

- The *web agent* enforces URL-based policy for C applications.
- The *Java Platform, Enterprise Edition (Java EE) agent* enforces URL-based policy and Java EE-based policy for Java applications on Java EE containers.

Note – When 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

Note – If the resource is in the Not Enforced list defined for the policy agent, access is allowed even if there is no valid session.

More information on the architecture of the Policy Service can be found in the Policy Service Architecture document on the OpenSSO web site. For an overview of the available policy agents and links to specific information on installation, see the *Sun Java System Federated Access Manager Policy Agent 3.0 User's Guide*.

Session Service

The mission of the Federated Access Manager Session Service is to maintain information about an authenticated user's session across all web applications participating in a user session. Additionally, Federated Access Manager provides continuous proof of the user's identity, enabling the user to access multiple enterprise resources without having to provide credentials each time. This 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.
- **SSO session.** The user provides credentials once, and then accesses multiple applications within the same DNS domain.
- **Cross domain SSO (CDSSO) session.** The user provides credentials once, and then accesses applications among multiple DNS domains.

A *user session* is the interval between the time a user successfully authenticates through Federated Access Manager and is issued a session token, and the moment the user logs out of the session. In what might be considered a typical user session, an employee accesses the corporate benefits administration service. The service, monitored by Federated Access Manager, prompts the user for a username and password. With the credentials Federated Access Manager can *authenticate*, or verify that the user is who he says he is. Following authentication, Federated Access Manager allows the user access to the service providing authorization is affirmed. Successful authentication through Federated Access Manager results in the creation of a *session data structure* for the user or entity by the Session Service. Generally speaking, the Session Service performs some or all of the following:

- Generates unique session identifiers, one for each user's session data structure

Note – A session data structure is initially created in the `INVALID` state with default values for certain attributes and an empty property list. Once the session is authenticated, the session state is changed to `VALID` and session data is updated with the user's identity attributes and properties.

- Maintains a master copy of session state information

Note – The session state maintained on the client side is a cached view of the actual session data structure. This cache can be updated by either the active polling mechanism or the session notification triggered by the Session Service.

- Implements time-dependent behavior of sessions — for example, enforces timeout limits
- Implements session life cycle events such as logout and session destruction
- Notifies all participants in the same SSO environment of session state changes
- Enables SSO and cross-domain single sign-on (CDSSO) among applications external to Federated Access Manager by providing continued proof of identity.
- Allow participating clients to share information across deployments
- Implement high availability facilities

Figure 2–6 illustrates the components of the Session Service.

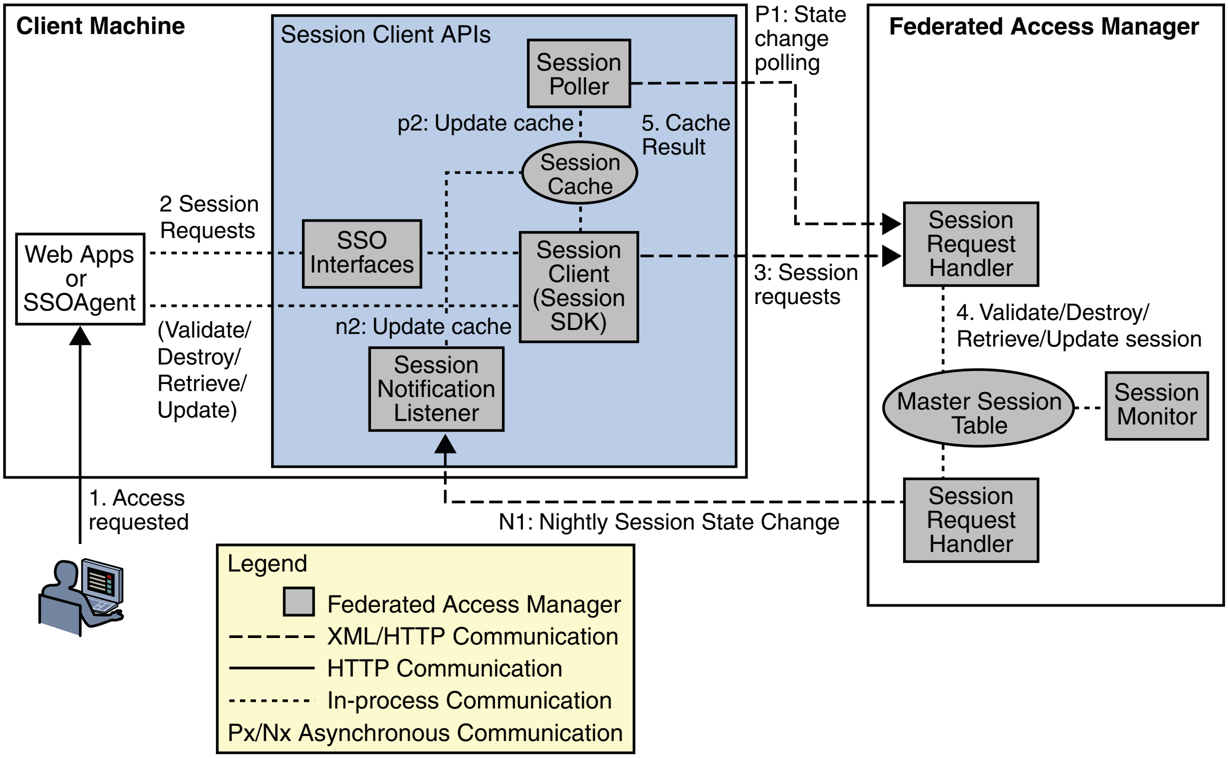


FIGURE 2–6 Components within the Session Service Framework

Additionally, Figure 2–7 illustrates how the messaging capabilities of Message Queue can be used to push session information to a persistent store based on the Berkeley DataBase (DB). Using Federated Access Manager in this manner enables the following key feature:

- Session Failover allows an alternative Federated Access Manager server to pick up a given user session when the server owning the original session fails.
- Session Constraints allow deployments to specify constraints on a sessions, such as one session per user.

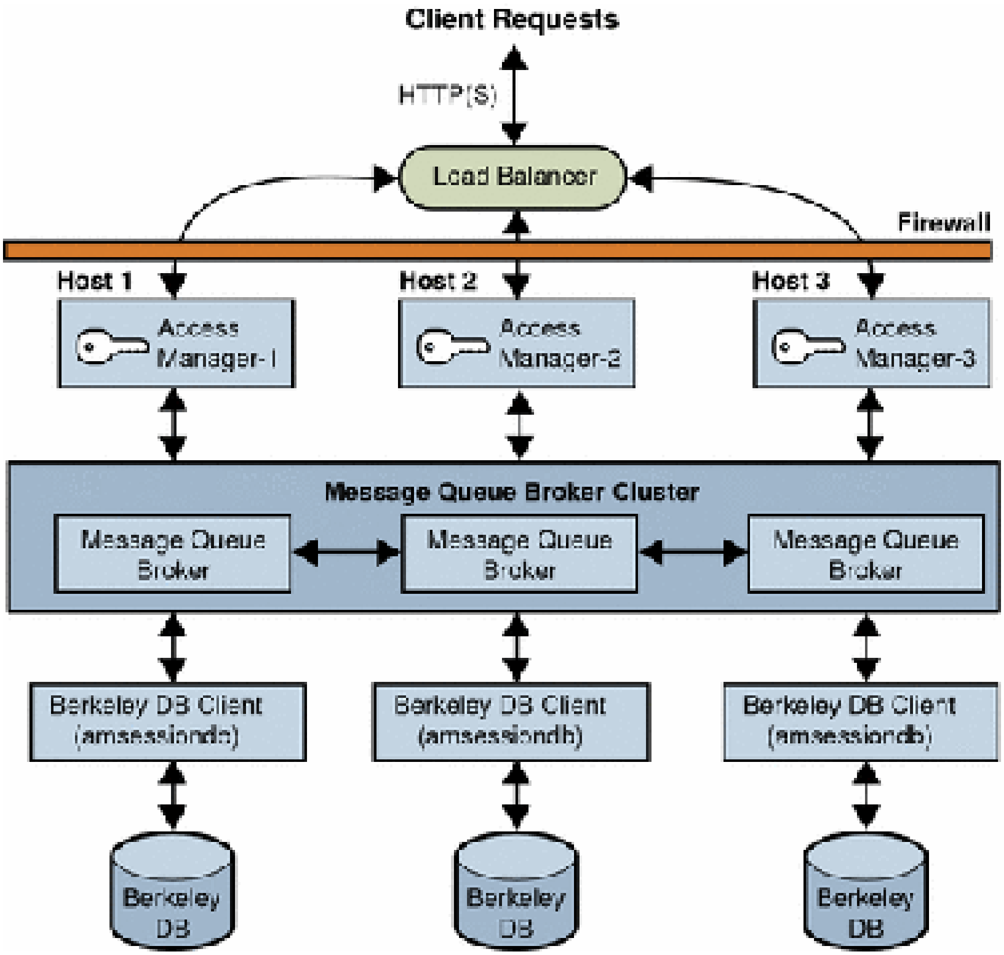


FIGURE 2–7 Session Persistence Deployment Architecture

More information on the architecture of the Session Service can be found in the Session Service Architecture document on the OpenSSO web site. For more information on session failover, see Chapter 5, “Implementing Federated Access Manager Session Failover,” in *Sun Federated Access Manager 8.0 Installation and Configuration Guide*.

Logging Service

When a user logs in to a resource protected by Federated Access Manager, the Logging Service records information about the user's activity. The common Logging Service can be invoked by components residing on the same server as Federated Access Manager as well as those on the client machine, allowing the actual mechanism of logging (such as destination and formatting) to be separated from the contents which are specific to each component. You can write custom log operations and customize log plug-ins to generate log reports for specific auditing purposes.

Administrators can control log levels, authorize the entities that are allowed to create log entries and configure secure logging. Logged information includes the name of the host, an IP address, the identity of the creator of the log entry, the activity itself, and the like. Currently, the fields logged as a *log record* are controlled by the Configurable Log Fields selected in the Logging Configuration page located under the System tab of the Federated Access Manager console. The Logging Service is dependent on the client application (using the Logging APIs) creating a programmatic `LogRecord` to provide the values for the log record fields. The logging interface sends the logging record to the Logging Service which determines the location for the log record from the configuration. A list of active logs can also be retrieved using the Logging API. Figure 2–8 illustrates logging communications.

Caution – Generally speaking, writing log records can be done remotely, using the Client SDK, but anything involving the reading API can only be done on the machine on which Federated Access Manager is installed. Using the reading API uses a great deal of system resources, especially when database logging is involved.

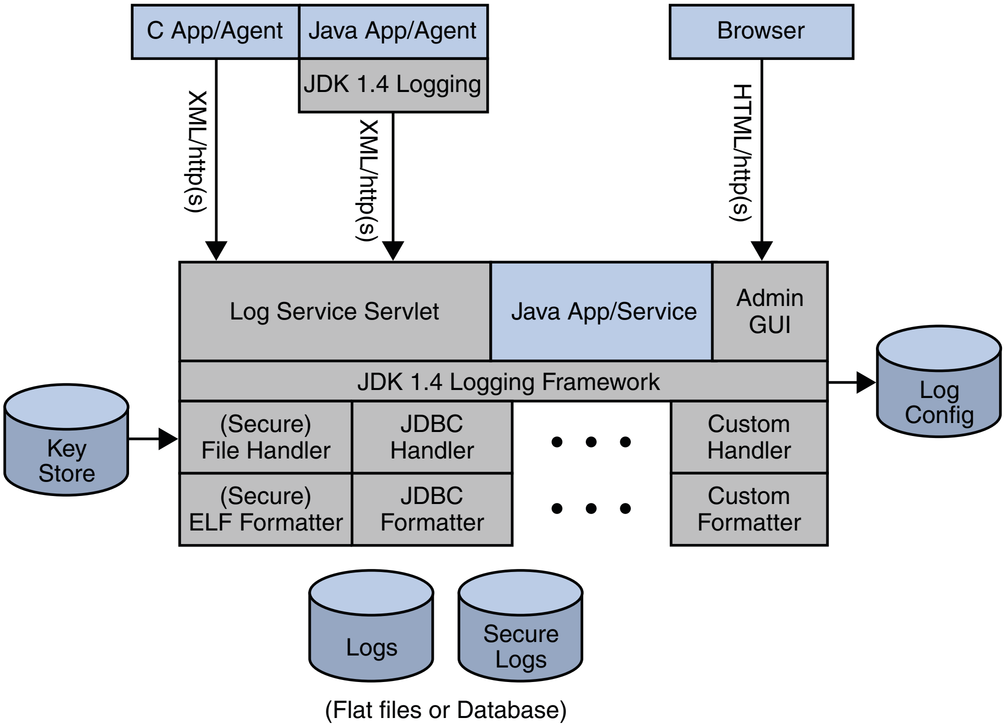


FIGURE 2–8 Components within the Logging Service Framework

Logs can be written to flat files or to one of the supported databases (Oracle and MySQL). See Chapter 11, "Logging and the Java Enterprise System Monitoring Framework," for more information.

Identity Repository Service

The Identity Repository Service allows Federated Access Manager to integrate an existing user data store (such as a corporate LDAP server) into the environment. The Identity Repository Service is able to access user profiles (as well as group and role assignments if supported) and is capable of spanning multiple repositories — even of different types. The current implementation supports Sun Java System Directory Server, IBM Tivoli Directory and Microsoft Active Directory.

Access to the Identity Repository Service is provided by the `com.sun.identity.idm` Java package. The `AMIdentityRepository` class represents a realm that has one or more identity repositories configured and provides interfaces for searching, creating and deleting identities. The `AMIdentity` class represents an individual identity such as a user, group or role and provides interfaces to set, modify and delete identity attributes and assign and unassign services. IdRepo is an abstract class that contains the methods that need to be implemented by plug-ins when building new adapters for repositories not currently supported.

The Identity Repository Service is configured per realm under the Data Stores tab and it's main functions are:

- To specify an identity repository that will store service configurations and attributes for users, groups and roles.
- To provide a list of identity repositories that can provide user attributes to the Policy Service and Federation Services frameworks.
- To combine the attributes obtained from different repositories.
- To provide interfaces to create, read, edit, and delete identity objects such as a realm, role, group, user, and agent.
- To map identity attributes using the Principal Name from the SSOToken object.

Note – Default administrator roles are also defined by the Identity Repository Service. These include the top-level Administrator, the Realm Administrator, the top-level Policy Administrator, the Realm Policy Administrator, the Realm Agent Administrator, and the Realm Log Administrator. For example, the Realm Administrator can access all data in all configured realms while the Subrealm Administrator can access data only within the specified realm. The Policy Administrator can access all policies in all configured realms while the Policy Realm Administrator can access policies only within the specified realm. For more information, see XXXXXXX Deployment Planning Guide. For information on realm privileges, see “Privileges” in *Sun Federated Access Manager 8.0 Administration Guide*.

Federation Services

Federated Access Manager provides an open and extensible framework for identity federation and associated web services to resolve the problems of identity-enabling web services, web service discovery and invocation, security, and privacy. Federation Services are built on the following standards:

- Liberty Alliance Project Identity Federation Framework (Liberty ID-FF) 1.1 and 1.2
- OASIS Security Assertion Markup Language (SAML) 1.0 and 1.1
- OASIS Security Assertion Markup Language (SAML) 2.0
- WS-Federation (Passive Requestor Profile)

Federation Services allows organizations to share a identity information (for example, which organizations and users are trusted, and what types of credentials are accepted) securely. Once this data can be exchanged securely, identity federation is possible, allowing a user to consolidate the many local identities 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 site without having to re-establish identity. The following figure illustrates the actions and services common to federation and how they interact with other non-federation FAM components.

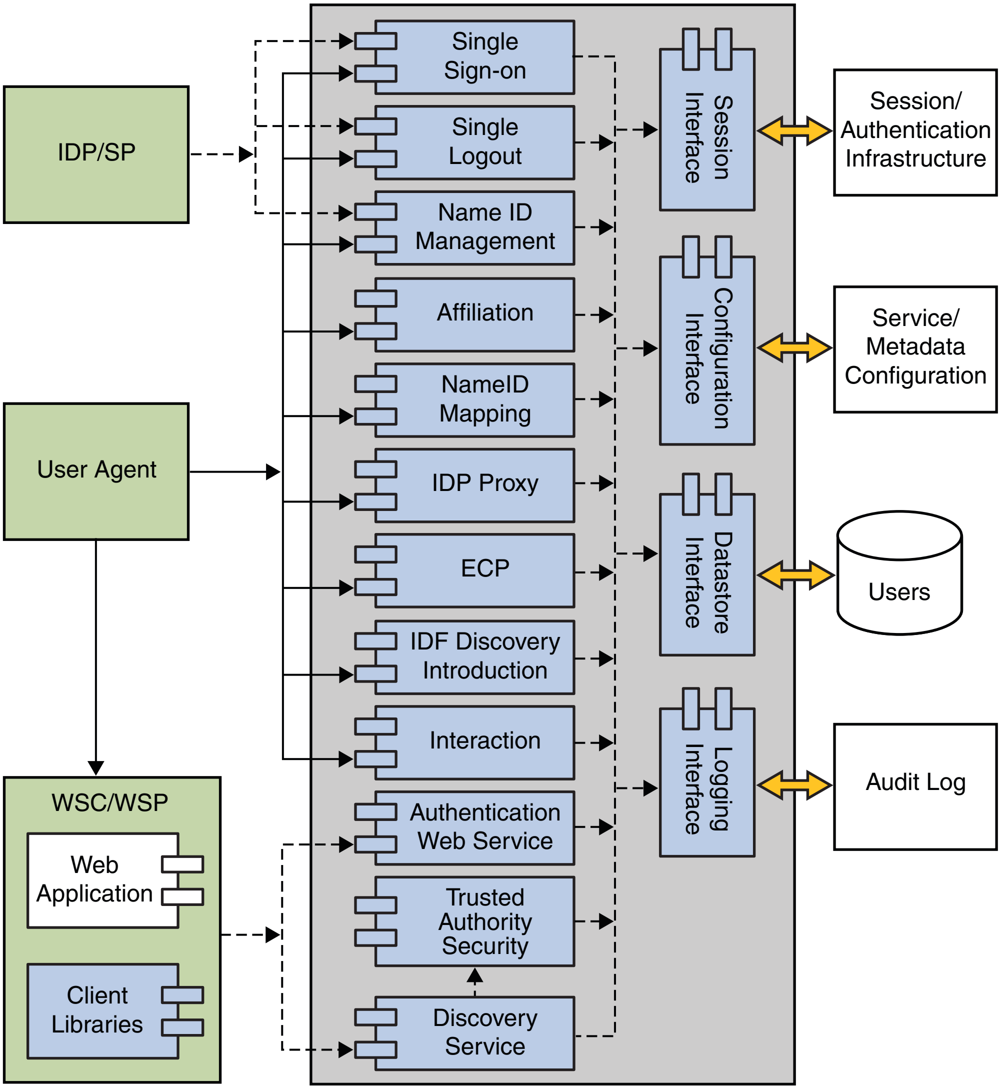


FIGURE 2–9 Federation Components Interaction within the Federated Access Manager Infrastructure

More information on the Federation Services can be found in the Open Federation Architecture and the Federation Use Case documentation on the OpenSSO web site. Also, see Part III in this book.

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. They 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. Federated Access Manager provides web services that primarily 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. Federated Access Manager implements the follow web service specifications.

- Liberty Alliance Project Identity Web Services Framework (Liberty ID-WSF) 1.0, 1.1, and 2.0
- Web Services-Interoperability (WS-I) Basic Security Profile

The following table lists the Federated Access Manager web services.

TABLE 2–1 Federated Access Manager Web Services

Web Service Name	Description
Authentication Web Service	Provides authentication to a web service client (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.
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.

TABLE 2–1 Federated Access Manager Web Services *(Continued)*

Web Service Name	Description
Security Token Service	The centralized Security Token Service that issues, renews, cancels, and validates security tokens.
SOAP Binding Service	A set of Java APIs implemented 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.

Federated Access Manager uses both XML files and Java interfaces to manage web services and web services configuration data. A Federated Access Manager XML file is based on the structure defined in the Federated Access Manager Document Type Definition (DTD) files located in *path-to-context-root*`/fam/WEB-INF`. The main `sms.dtd` file defines the structure for all Federated Access Manager service files (located in *path-to-context-root*`/fam/WEB-INF/classes`).

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

For more information, see Part IV.

Web Services Security

In message security, security information is applied at the message layer and travels along with the web services message. Message layer security differs from transport layer security in that it can be used to decouple message protection from message transport so that messages remain protected after transmission, regardless of how many hops they travel on. This message security is available as Web Services Security in Federated Access Manager and through the installation of an *authentication agent*. Web Services Security is the implementation of the WS-Security specifications and the Liberty Alliance Project Identity Web Services Framework (Liberty ID-WSF). Web Services Security allows communication with the Security Token Service to insert security tokens in outgoing messages and evaluate incoming messages for the same. Towards this end, authentication agents based on the Java Specification Request (JSR) 196 must be downloaded and installed on the web services client (WSC) machine and the web services provider (WSP) machine.

To secure web services communications, the requesting party must first be authenticated with a security token which is added to the SOAP header of the request. Additionally, the WSC needs to be configured to supply message level security in their SOAP requests and the WSP needs to be configured to enable message level security in their SOAP responses. Figure 2–10 illustrates the components used during a secure web services interaction.

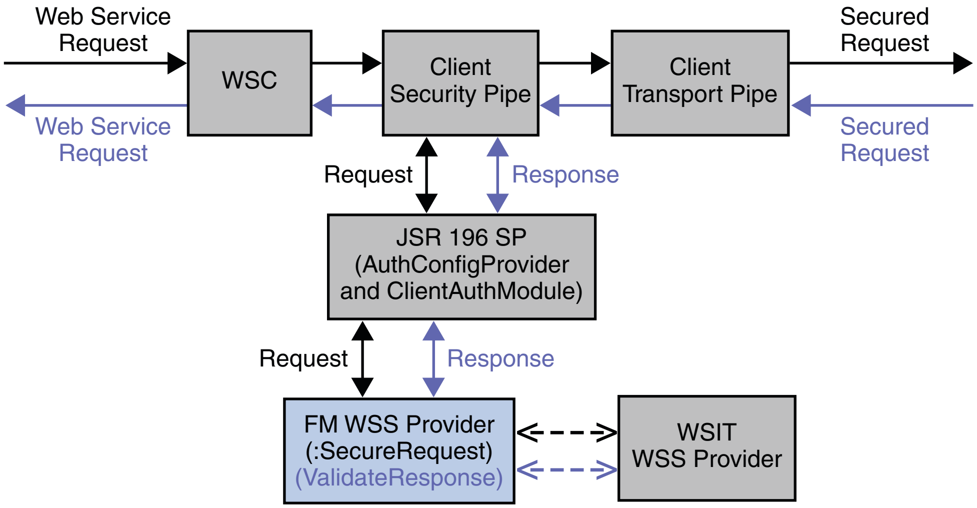


FIGURE 2–10 Components within the Web Services Security Interactions

Note – The stand alone applications can directly invoke the interfaces (secure request by WSC, and validate response by WSP) from the WS-Security Library and establish message-level end-to-end web service security. Standalone Java applications do not need the WS-Security Provider Plugin.

For more information, see Part IV.

Identity Services

Federated Access Manager contains client interfaces for authentication, authorization, session management, and logging in Java, C, and C++ (using the proprietary XML and SOAP over HTTP or HTTPs). These interfaces are used by policy agents and custom applications. Development using these interfaces, though, is labor-intensive. Additionally, the interfaces cause dependencies on Federated Access Manager. Therefore, Federated Access Manager now has simple interfaces that can be used for efficient development of:

- Authentication (verification of user credentials, password management)
- Authorization (policy evaluation for access to secured resources)
- Provisioning (self-registration, creating or deleting identity profiles, retrieve or update identity profile attributes)
- Logging (auditing, recording operations)

- Token validation
- Search (return a list of identity profile attributes that match a search filter)

These Identity Services are offered using either SOAP and the Web Services Description Language (WSDL) or Representational State Transfer (REST). They are implemented by pointing an integrated development environment (IDE) application project to the appropriate URL and generating the stub code that wraps the function calls to the services.

Note – Federated Access Manager supports Eclipse, NetBeans, and Visual Studio.

The user interacts with the presentation logic of the application which could be, for example, a calendar, a human resources applications, or a banking account. The application calls either of the Identity Services to authenticate and authorize the identity, create personalized services, and log the actions. When contacted at the respective URL, Federated Access Manager obtains the user profile from the appropriate identity repository for authentication and the policy configuration from the appropriate configuration data store, and writes the actions to the configured log file. Figure 2–11 illustrates the components of the Identity Services.

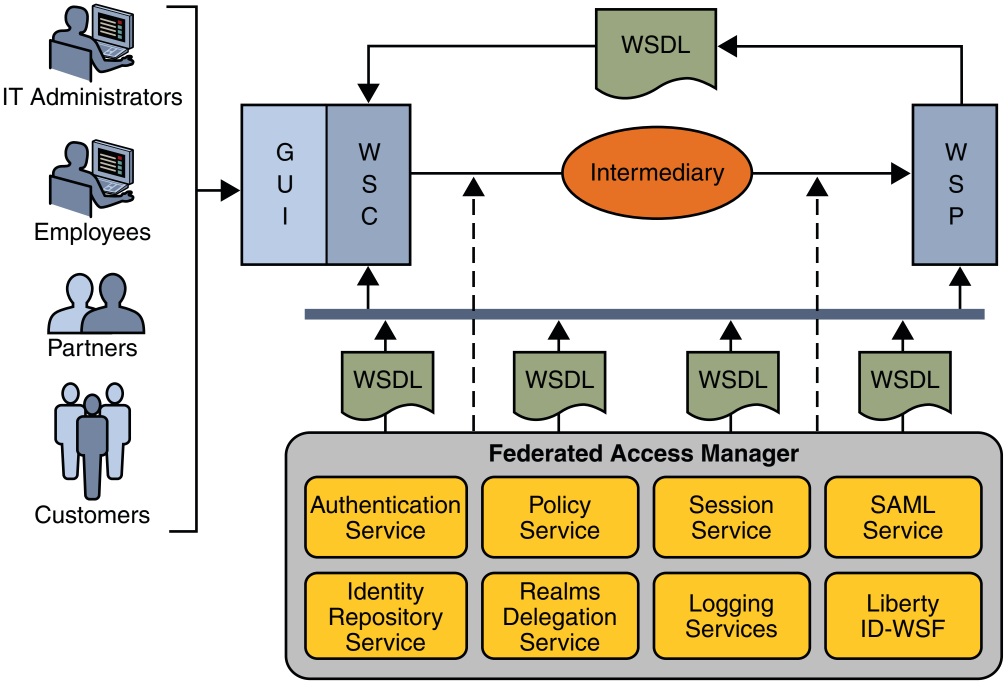


FIGURE 2–11 Components within the Identity Services Interactions

---

Note – Identity Services does not require the Client SDK or deployment of an agent or proxy to protect a resource.

For more information, see Part IV.

Global Services

Global services take configuration values and perform functions for Federated Access Manager on a global basis. The following table lists the global services with brief descriptions.

TABLE 2–2 Global Federated Access Manager Services

Service	What it Does
Common Federation Configuration	Contains configuration attributes for Federation Services.
Liberty ID-FF Service Configuration	Contains configuration attributes for the Liberty Alliance Project Identity Federation Framework.

TABLE 2–2 Global Federated Access Manager Services (Continued)

Service	What it Does
Liberty ID-WSF Security Service	Contains configuration attributes for the Liberty Alliance Project Identity Web Services Framework.
Liberty Interaction Service	Contains configuration attributes for the Liberty Alliance Project Interaction Service — used to get consent from an owner to expose data, or to get additional data.
Multi-Federation Protocol	Contains configuration attributes for multi-federation protocol circles of trust.
Password Reset	Contains configuration attributes for the Password Reset Service.
Policy Configuration	Contains configuration attributes for the Policy Service.
SAML v2 Service Configuration	Contains configuration attributes for the SAML v2 interactions.
SAML v2 SOAP Binding	Contains configuration attributes for SAML v2 SOAP Binding Service.
Security Token Service	Contains configuration attributes for the Security Token Service.
Session	Contains configuration attributes for the Session Service.
User	Contains configuration attributes for user profiles.

Realms

A *realm* is the unit that Federated Access Manager uses to organize configuration information. Authentication properties, authorization policies, data stores, subjects (including a user, a group of users, or a collection of protected resources) and other data can be defined within the realm. The data stored in a realm can include, but is not limited to:

- One or more subjects (a user, a group of users, or a collection of protected resources)
- A definition of one or more data stores to store subject (user) data
- Authentication details 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 Federated Access Manager the subjects can access.
- Responder configurations that allows applications to personalize the user experience, once the user has successfully authenticated and been given access.
- Administration data for realm management

You create a top-level realm when you deploy Federated Access Manager. The top-level realm (by default `fam`) is the root of the Federated Access Manager instance and contains Federated Access Manager configuration data; it cannot be changed after it is created. All other realms are configured under the `fam` realm. These sub-realms may contain other sub-realms and so on. Sub-realms identify sets of users and groups that have different authentication or authorization requirements.

The Federated Access Manager framework aggregates realm properties as part of the configuration data. Figure 2–12 illustrates how configuration data can use a hierarchy of realms to distribute administration responsibilities. Region 1, Region 2, and Region 3 are realms; Development, Operations, and Sales are realms sub to Region 3.

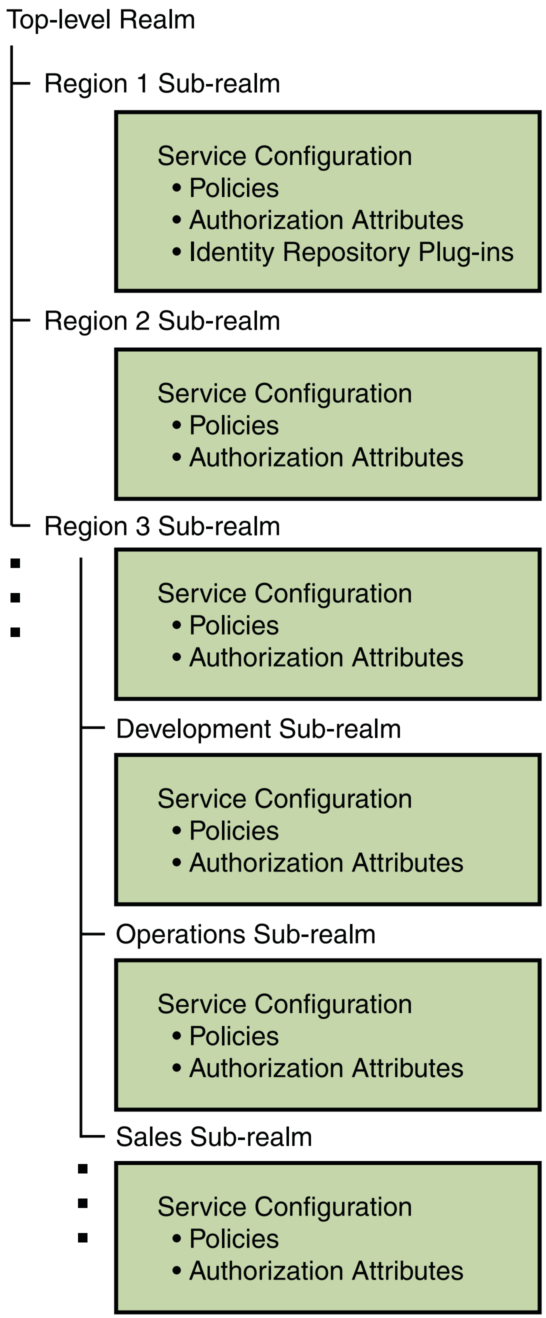


FIGURE 2–12 Realm Hierarchy for Configuration Data

Note – Federated Access Manager 8.0 supports the Sun Java System Access Manager Legacy mode (which contains no realms) with a provided interface.

Additional Components

The following sections provide information on additional components used in a Federated Access Manager deployment.

- "Data and Data Stores" on page 52
- "The `bootstrap` File" on page 58
- "Policy Agents" on page 59
- "Authentication Agents" on page 60
- "Federated Access Manager Tools" on page 61
- "Client SDK" on page 61
- "Service Provider Interfaces for Plug-ins" on page 61

Data and Data Stores

Federated Access Manager services need to interact with a number of different data stores. The following distinct repositories can be configured.

- A configuration repository provides server and service specific data.
- One or more identity repositories provide user profile information.
- Authentication repositories provide authentication credentials to a particular module of the Authentication Service.

A common LDAP connection pooling facility allows efficient use of network resources. In the simplest *demonstration* environment, a single LDAP repository is sufficient for all data however, the typical *production* environment tends to separate configuration data from other data. The following sections contain more specific information.

- "Configuration Data" on page 52
- "Identity Data" on page 55
- "Authentication Data" on page 58

Configuration Data

The default configuration of Federated Access Manager creates a branch in a fresh installation of a configuration data store for storing service configuration data and other information pertinent to the server's operation. Federated Access Manager components and plug-ins access the configuration data and use it for various purposes including:

- Accessing policy data for policy evaluation.

- Finding location information for identity data stores and Federated Access Manager services.
- Retrieving authentication configuration information that define how users and groups authenticate.
- Finding which partner servers can send trusted SAML assertions.

Federated Access Manager supports Microsoft Active Directory, Sun Java System Directory Server, and the open source OpenDS as configuration data stores. Flat files (supported in previous versions of the product) are no longer supported but configuration data store failover is by using `bootstrap`. (See "The `bootstrap` File" on page 58.) Figure 2–13 illustrates how configuration data in the embedded data store is accessed.

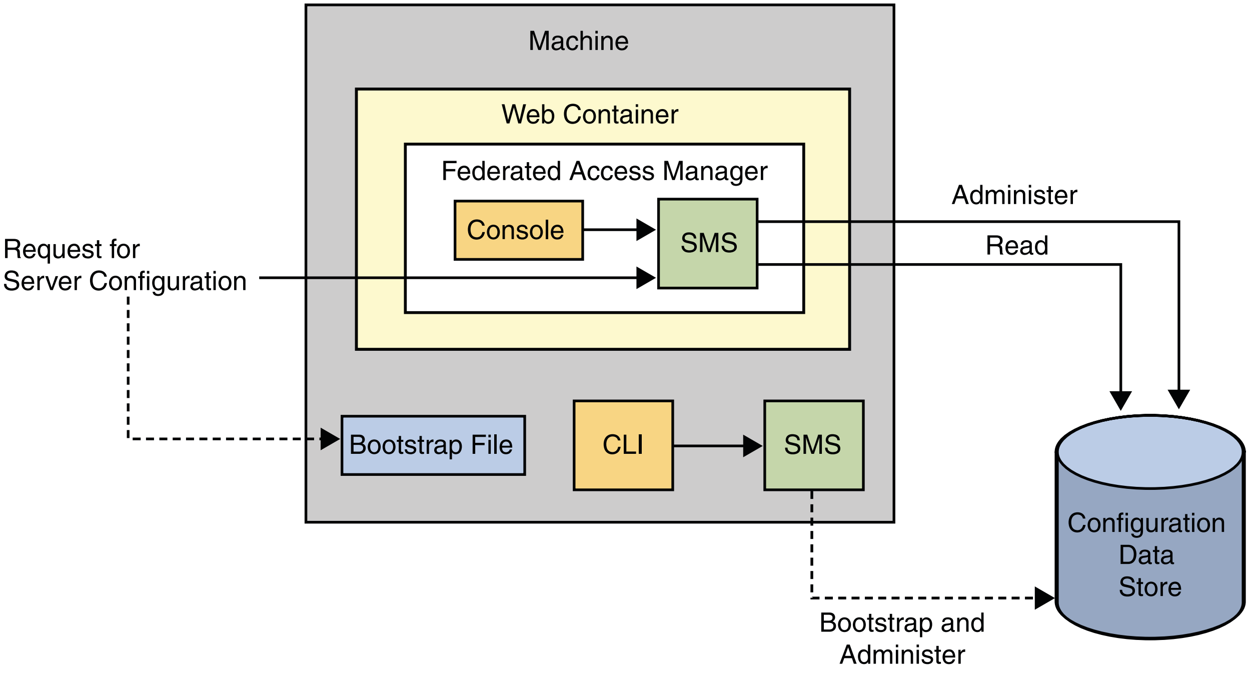


FIGURE 2–13 Accessing Configuration Data

Previous releases of Access Manager and Federation Manager stored product configuration data in a property file named `AMConfig.properties` that was installed local to the product instance directory. This file is deprecated for Federated Access Manager although supported for backward comparability. See the *Sun Federated Access Manager 8.0 Installation and Configuration Guide* for more information.

Configuration data comprises the attributes and values in the Federated Access Manager configuration services, as well as default Federated Access Manager users like `amadmin` and `anonymous`. It is stored under **`ou=services,`***ROOT SUFFIX* in the configuration data store. Following is a partial listing of the XML service files that contribute to the data. They can be found in the *path-to-context-root*`/fam/WEB-INF/classes` directory.

Note – The data in this node branch is private and is mentioned here for information purposes only.

- `AgentService.xml`
- `amAdminConsole.xml`
- `amAgent70.xml`
- `amAuth.xml`
- `amAuth-NT.xml`
- `amAuthAD.xml`
- `amAuthAnonymous.xml`
- `amAuthCert.xml`
- `amAuthConfig.xml`
- `amAuthDataStore.xml`
- `amAuthHTTPBasic.xml`
- `amAuthJDBC.xml`
- `amAuthLDAP.xml`
- `amAuthMSISDN.xml`
- `amAuthMembership.xml`
- `amAuthNT.xml`
- `amAuthRADIUS.xml`
- `amAuthSafeWord-NT.xml`
- `amAuthSafeWord.xml`
- `amAuthSecurID.xml`
- `amAuthWindowsDesktopSSO.xml`
- `amClientData.xml`
- `amClientDetection.xml`
- `amConsoleConfig.xml`
- `amDelegation.xml`
- `amEntrySpecific.xml`
- `amFilteredRole.xml`
- `amG11NSettings.xml`
- `amLogging.xml`
- `amNaming.xml`
- `amPasswordReset.xml`
- `amPlatform.xml`
- `amPolicy.xml`
- `amPolicyConfig.xml`
- `amRealmService.xml`

- `amSession.xml`
- `amUser.xml`
- `amWebAgent.xml`
- `idRepoEmbeddedOpenDS.xml`
- `idRepoService.xml`
- `identityLocaleService.xml`
- `ums.xml`

Caution – By default, the Federated Access Manager configuration data is created and maintained in the embedded configuration data store apart from any identity data. Although users can be created in the configuration data store this is only recommended for demonstrations and development environments.

For more information, see "Embedded Configuration Data" on page 67.

Identity Data

An *identity repository* is a data store where information about users, roles, and groups in an organization is stored. User profiles can contain data such as a first name, a last name, a phone number, group membership, and an e-mail address; an identity profile template is provided out-of-the-box but it can be modified to suit specific deployments.

Identity data stores are defined per realm. Because more than one identity data store can be configured per realm Federated Access Manager can access the many profiles of one identity across multiple data repositories. Sun Java System Directory Server, Microsoft Active Directory and IBM Tivoli Directory are the currently supported identity repositories. Plug-ins can be developed to integrate other types of repositories (for example, a relational database). Figure 2–14 illustrates a Federated Access Manager deployment where the identity data and the embedded configuration data are kept in separate data stores.

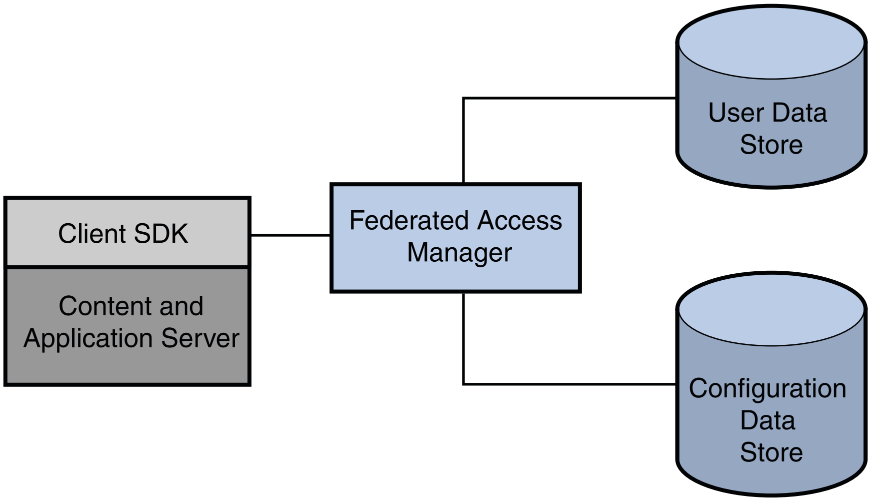


FIGURE 2–14 Federated Access Manager Deployment with Two Data Stores

---

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

Federated Access Manager provides out-of-the-box plug-in support for some identity repositories. Each default plug-in configuration includes details about what operations are supported on the underlying data store. Once a realm is configured to use a plug-in, the framework can instantiate it and execute the operations on the appropriate identity repository. Each new plug-in developed must have a corresponding service management schema defining its configuration attributes. This schema would be integrated as a sub schema into `idRepoService.xml`, the service management file for the Identity Repository Service that controls the identity data stores available under a realm's Data Stores tab. The following sections contain information on the out-of-the-box plug-ins.

- "Generic Lightweight Directory Access Protocol (LDAP) version 3" on page 56
- "LDAPv3 Plug-in for Active Directory" on page 57
- "LDAPv3 Plug-in for Active Directory" on page 57
- "Sun Directory Server With FAM Core Services" on page 57
- "Sun Directory Server With Full Schema (including Legacy)" on page 57
- "Access Manager Repository Plug-in" on page 57

Generic Lightweight Directory Access Protocol (LDAP) version 3

The Generic LDAPv3 identity repository plug-in can reside on an instance of any directory that complies with the LDAPv3 specifications. The underlying 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. Each data store has a name that is unique among a realm's data store names, but not necessarily unique across all realms. The `com.sun.identity.idm.plugins.ldapv3.LDAPv3Repo` class provides the default LDAPv3 identity repository implementation. There are also implementations for Active Directory and IBM Tivoli Directory

Note – This configuration is not compatible with previous versions of this product.

LDAPv3 Plug-in for Active Directory

The Generic LDAPv3 identity repository plug-in was used to develop a default plug-in to write identity data to an instance of Microsoft® Active Directory®. The administration console provides a way to configure the directory but the schema needs to be loaded manually.

LDAPv3 Plug-in for Tivoli Directory

The Generic LDAPv3 identity repository plug-in was used to develop a default plug-in to write identity data to an instance of IBM Tivoli Directory®. The administration console provides a way to configure the directory but the schema needs to be loaded manually.

Sun Directory Server With FAM Core Services

This repository resides in an instance of Sun Java System Directory Server and holds the configuration data. This option is available during the initial configuration of Federated Access Manager.

Sun Directory Server With Full Schema (including Legacy)

This repository resides in an instance of Sun Java System Directory Server and holds the configuration data when installing Federated Access Manager in Legacy and Realm mode. This option must be manually configured.

Access Manager Repository Plug-in

The Access Manager Repository can reside only in Sun Java System Directory Server and is used with the **Sun Directory Server With Access Manager Schema**. During installation, the repository is created in the same instance of Sun Java System Directory Server that holds the configuration data. 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. 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.

Authentication Data

Authentication data contains authentication credentials for Federated Access Manager users. An authentication data store is aligned with a particular authentication module, and might include:

- RADIUS servers
- SafeWord authentication servers
- RSA ACE/Server systems (supports SecurID authentication)
- LDAP directory servers

Note – Identity data may include authentication credentials although authentication data is generally stored in a separate authentication repository.

The `bootstrap` File

Federated Access Manager uses a file to bootstrap itself. Previously, `AMConfig.properties` held configuration information to bootstrap the server but now a file named `bootstrap` points to the configuration data store allowing the setup servlet to retrieve the bootstrapping data. After deploying the Federated Access Manager WAR and running the configuration wizard, configuration data is written to the configuration data store by the service management API contained in the Java package, `com.sun.identity.sm`. The setup servlet creates `bootstrap` in the top-level `/fam` directory. The content in bootstrap can be either of the following:

- A directory local to Federated Access Manager (for example, `/export/SUNWam`) indicating the server was configured with a previous release. The directory is where `AMConfig.properties` resides.

- An encoded URL that points to a directory service using the following format:

  `ldap://`*ds-host*`:`*ds-port*/*server-instance-name*`?pwd=`*encrypted-amadmin-password*`&`
  `embeddedds=`*path-to-directory-service-installation*`&basedn=`*base-dn*`&`
  `dsmgr=`*directory-admin*`&dspwd=`*encrypted-directory-admin-password*

  For example:

  ```
  ldap://ds.samples.com:389/http://owen2.red.sun.com:8080/fam?
  pwd=AQIC5wM2LY4Sfcxi1dVZEdtfwar2vhWNkmS8&embeddedds=/fam/opends&
  basedn=dc=fam,dc=java,dc=net&dsmgr=cn=Directory+Manager
  &dspwd=AQIC5wM2LY4Sfcxi1dVZEdtfwar2vhWNkmS8
  ```

  where

  - `ds.samples.com:389` is the host name and port of the machine on which the directory is installed.
  - `http://owen2.red.sun.com:8080/fam` is the instance name.
  - `AQIC5wM2LY4Sfcxi1dVZEdtfwar2vhWNkmS8` is the encrypted password of the OpenSSO administrator.
  - `/fam/opends` is the path to the directory installation.
  - `dc=fam,dc=java,dc=net` is the base DN.
  - `cn=Directory+Manager` is the directory administrator.
  - `BQIC5xM2LY4SfcximdVZEdtfwar4vhWNkmG7` is the encrypted password for the directory administrator.

If more than one URL is present in the file and Federated Access Manager is unable to connect or authenticate to the data store at the first URL, the bootstrapping servlet will try the second (and so on). Additionally, the number sign [#] can be used to exclude a URL as in:

```
# ldap://ds.samples.com:389/http://owen2.red.sun.com:8080/fam?
pwd=AQIC5wM2LY4Sfcxi1dVZEdtfwar2vhWNkmS8&embeddedds=/fam/opends&
basedn=dc=fam,dc=java,dc=net&dsmgr=cn=Directory+Manager
&dspwd=AQIC5wM2LY4Sfcxi1dVZEdtfwar2vhWNkmS8
```

Policy Agents

Policy agents are an integral part of SSO and CDSSO sessions. They are programs that police the web container on which resources are hosted. All policy agents interact with the Authentication Service in two ways:

- To authenticate itself in order to establish trust. This authentication happens using the Client SDK.

- To authenticate users having no valid session for access to a protected resource. This authentication happens as a browser redirect from the Distributed Authentication User Interface.

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 Federated Access Manager Authentication Service for authentication. Following this, the policy agent requests the authenticated user's assigned policy and evaluates it to allow or deny access. (A *policy* defines the rules that specify a user's access privileges to a protected resource.) Federated Access Manager supports two types of policy agents:

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

Both types of agents are available for you to install as programs separate from Federated Access Manager. Policy agents are basically clients written using the Client SDK and Identity Services.

Note – All HTTP requests are implicitly denied unless explicitly allowed by the presence of a valid session and a policy allowing access. If the resource is defined in the Not Enforced list for the policy agent, access is allowed even if there is no valid session.

For more information, see J2EE Agent Architecture and Web Agent and C-API Architecture on the OpenSSO web site. For an overview of the available policy agents and links to specific information on installation, see the *Sun Java System Federated Access Manager Policy Agent 3.0 User's Guide*.

Authentication Agents

Authentication agents plug into web containers to provide message level security for web services, and supports both Liberty Alliance Project token profiles as well as Web Services-Interoperability Basic Security Profiles (WS-I BSP). (A *profile* defines the HTTP exchanges required to transfer XML requests and responses between web service clients and providers.) Authentication agents use an instance of Federated Access Manager for all authentication decisions. Web services requests and responses are passed to the appropriate authentication modules using standard Java representations based on the transmission protocol. An HTTP Authentication Agent or a SOAP Authentication Agent can be used. For more information, see "Web Services Security" on page 45.

Federated Access Manager Tools

Contained within the Federated Access Manager ZIP are `famAdminTools.zip` and `famSessionTools.zip`. The following sections have some information about these tools.

- "`famadm` Command Line Interface" on page 61
- "Session Failover Tools" on page 61

`famadm` Command Line Interface

`famadm`, the command line interface (CLI), provides a second option to administer Federated Access Manager using the command line. For example, `famadm` can be used to create a policy or import and export Liberty ID-FF metadata. `famadm` is the recommended way for batch processing. It is in `famAdminTools.zip`. For more information, see Chapter 4, "Installing the Federated Access Manager Utilities and Scripts," in *Sun Federated Access Manager 8.0 Installation and Configuration Guide* and Part I, "Command Line Interface Reference," in *Sun Federated Access Manager Administration Reference*.

Session Failover Tools

`famSessionTools.zip` contains scripts and binaries for setting up session failover and databases. For more information, see Chapter 5, "Implementing Federated Access Manager Session Failover," in *Sun Federated Access Manager 8.0 Installation and Configuration Guide*.

Client SDK

Enterprise resources cannot be protected by Federated Access Manager until the Federated Access Manager Client SDK is installed on the machine that contains the resource that you want to protect. (The Client SDK is automatically installed with a policy agent.) The Client SDK allows you to customize an application by enabling communication with Federated Access Manager for retrieving user, session, and policy data. For more information, see Chapter 1, "Enhancing Remote Applications Using the Client Software Development Kit," in *Sun Federated Access Manager 8.0 Developer's Guide* and the *Federated Access Manager 8.0 Java API Reference*.

Service Provider Interfaces for Plug-ins

The Federated Access Manager service provider interfaces (SPI) can be implemented as plug-ins to provide customer data to the Federated Access Manager 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 Federated Access Manager plug-ins themselves. You can develop additional custom plug-ins to work with the SPI. For a complete list of the SPI, see the *Federated Access Manager 8.0 Java API Reference*. Additional information can be found in the *Sun Federated Access Manager 8.0 Developer's Guide*. The following sections contain brief descriptions.

- "Authentication Service SPI" on page 62
- "Federation Service SPI" on page 62
- "Identity Repository Service SPI" on page 62
- "Policy Service SPI" on page 62
- "Service Configuration Plug-in" on page 62

Authentication Service SPI

The `com.sun.identity.authentication.spi` package provides interfaces and classes for writing a supplemental authentication module to plug into Federated Access Manager. The `com.sun.identity.authentication` package provides interfaces and classes for writing a remote client application that can access user data in a specified identity repository to determine if a user's credentials are valid.

Federation Service SPI

The `com.sun.identity.federation.services` package provides plug—ins for customizing the Liberty ID-FF profiles implemented by Federated Access Manager. The `com.sun.identity.federation.plugins` package provides an interface that can be implemented to perform user specific processing on the service provider side during the federation process. The `com.sun.identity.saml2.plugins` package provides the SAML v2 service provider interfaces (SPI). The `com.sun.identity.wsfederation.plugins` package provides the WS-Federation based SPI.

Identity Repository Service SPI

The `com.sun.identity.idm` package contains the `IdRepo` interface that defines the abstract methods which need to be implemented or modified by Identity Repository Service plug-ins. `com.sun.identity.plugin.datastore` package contains interfaces that search for and return identity information such as user attributes and membership status for purposes of authentication.

Policy Service SPI

The `com.sun.identity.policy.interfaces` package provides interfaces for writing custom policy plug-ins for Conditions, Subjects, Referrals, Response Providers and Resources.

Service Configuration Plug-in

The `com.sun.identity.plugin.configuration` package provides interfaces to store and manage configuration data required by the core Federated Access Manager components and other plug-ins.

Note – In previous releases, the functionality provided by the Service Configuration plug-in was known as the Service Management Service (SMS).

◆ ◆ ◆ CHAPTER 3

Simplifying Federated Access Manager

This chapter contains information on the usability and manageability features of Federated Access Manager. It includes the following sections:

Installation and Configuration

Previous versions of Sun Microsystems' access management server were built for multiple hardware platforms, and different web containers. The complexity of this development process led to the release of separate platform and container patches. To alleviate this extraneous development, Federated Access Manager is now available as a single ZIP file which can be downloaded, unzipped, and quickly deployed; there will be no separate installations for each hardware platform. The ZIP file will contain the full Federated Access Manager web archive (WAR), layouts for the generation of other specific WARs, libraries, the Java API reference documentation, and samples. (Tools for use with Federated Access Manager, including the command line interfaces and policy and authentication agents, can be downloaded separately.) Figure 3–1 illustrates the process for deployment, installation and configuration of a new Federated Access Manager WAR and a patched WAR.

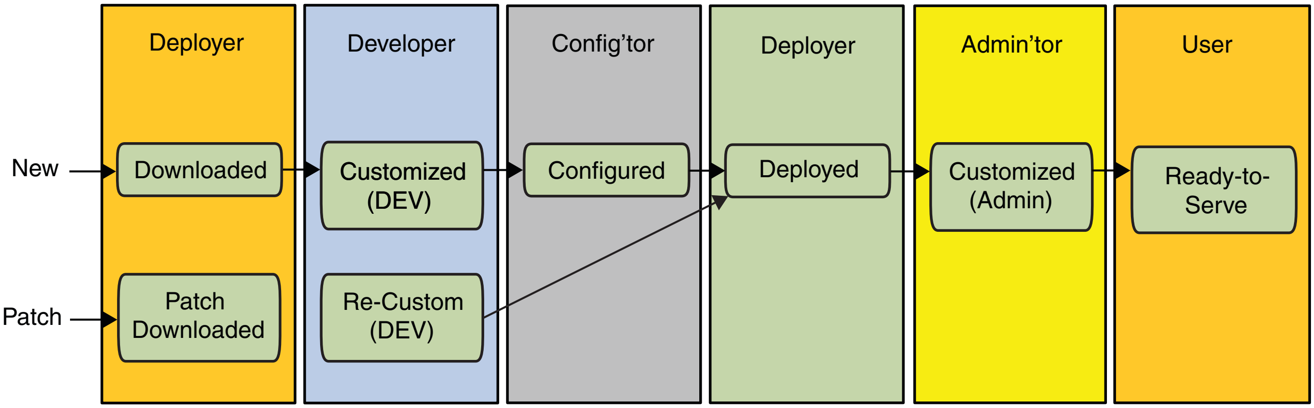


FIGURE 3–1 Customizing, Patching and Deploying the WAR

Note – When patched, a full patched version of the Federated Access Manager WAR will be included in the download, assuring that there is always a single download to get the latest bits.

The intent of this new process is to allow the administrator to download Federated Access Manager and deploy it on the container or platform of choice, using the web container's administration console or command line interfaces. After the initial launch of the deployed WAR, the user is directed to a JavaServer Page (JSP) called the `Configurator` that prompts for configuration parameters (including, but not limited to, the host name, port number, URI, and repositories), provides data validation for the parameter values to prevent errors, and eliminates post-installation configuration tasks. Once successfully configured, any further changes to the configuration data store must be made using the Federated Access Manager console or command line interfaces.

Note – When deploying Federated Access Manager against an existing legacy installation, the Directory Management tab will be enabled in the new console.

For more information including a directory layout of the ZIP, see the *Sun Federated Access Manager 8.0 Installation and Configuration Guide*.

Embedded Configuration Data

Federated Access Manager has implemented an embedded configuration data store to replace the `AMConfig.properties` and `serverconfig.xml` files which had been the storage files for server configuration data. Previously, each instance of the server installed had separate configuration files but now when deploying more than one instance of Federated Access Manager, all server configuration data is stored centrally, in one embedded configuration data store per instance. After the Federated Access Manager WAR is configured, a sub configuration is added under the Platform Service to store the data and a bootstrap file that contains the location of the configuration data store is created in the installation directory. Figure 3–2 illustrates how Federated Access Manager is bootstrapped.

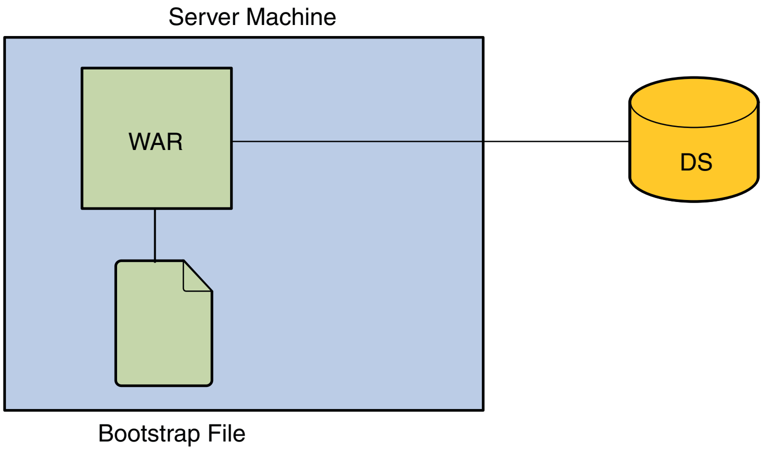


FIGURE 3–2 Bootstrapping Federated Access Manager

Post-installation, the configuration data can be reviewed and edited using the administration console or the `famadm` command line interface. For more information see the *Sun Federated Access Manager 8.0 Installation and Configuration Guide* and the *Sun Federated Access Manager 8.0 Administration Guide*.

Note – Federated Access Manager also supports an LDAPv3–based solution that uses an existing directory server for configuration data storage. This is configured during installation. Supported directories include Sun Java System Directory Server, Microsoft Active Directory, and IBM Tivoli Directory.

Centralized Agent Configuration

Policy agents function based on a set of configuration properties. Previously, these properties were stored in the `AMAgent.properties` file, residing on the same machine as the agent. With Centralized Agent Configuration, Federated Access Manager moves most of the agent configuration properties to the embedded configuration data store. Now agent profiles can be configured to store properties locally (on the machine to which the agent was deployed) or centrally (in the embedded configuration data store), making this new function compatible with both older 2.x agents and newer 3.0 agents. Following is an explanation of the local and central agent configuration repositories.

- Local agent configuration is supported for backward compatibility. Agent configuration data is stored in a property file named `FAMAgentConfiguration.properties` that is stored on the agent machine. It is only used by agent profiles configured locally.
- Centralized Agent Configuration stores agent configuration data in a centralized data store managed by Federated Access Manager. When an agent starts up, it reads its bootstrapping file to initialize itself. `FAMAgentBootstrap.properties` is stored on the agent machine and indicates the location from where the configuration properties need to be retrieved. It is used by agent profiles configured locally or centrally. Based on the repository setting in `FAMAgentBootstrap.properties`, it retrieves the rest of its configuration properties. If the repository is local, it reads the agent configuration from a local file; if the repository is remote, it fetches its configuration from Federated Access Manager.

Thus, Centralized Agent Configuration separates the agent configuration properties into two places: a bootstrapping file stored local to the agent and either a local (to the agent) or central (local to Federated Access Manager) agent configuration data store. `FAMAgentBootstrap.properties` is the bootstrapping file used by agent profiles configured locally or centrally. It is stored on the agent machine and indicates the local or central location from where the agent's configuration properties are retrieved. If the repository is local to the agent, it reads the configuration data from a local file; if the repository is remote, it fetches its configuration from Federated Access Manager. Choosing Centralized Agent Configuration provides an agent administrator with the means to manage multiple agent configurations from a central place using either the Federated Access Manager console or command line interface. Figure 3–3 illustrates how an agent retrieves bootstrapping and local configuration data, and configuration data from the configuration data store.

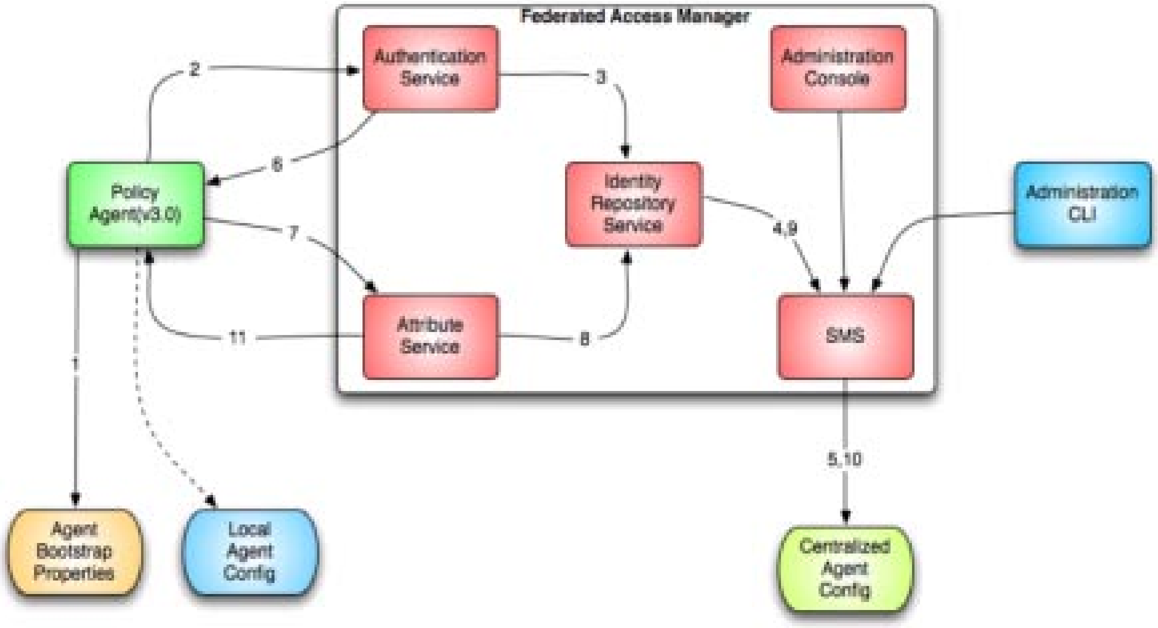


FIGURE 3–3 Retrieving Agent Configuration Data

An agent fetches its configuration properties periodically to determine if there have been any configuration changes. Any agent configuration changes made centrally are conveyed to the affected agents which will react accordingly based on the nature of the updated properties. If the properties affected are *hot swappable*, the agent can start using the new values without a restart of the underlying agent web container. Notification of the agent when configuration data changes and polling by the agent for configuration changes can be enabled. Agents can also receive notifications of session and policy changes.

Note – A agent configuration data change notification does not contain the actual data; it is just a ping that, when received, tells the agent to make a call to Federated Access Manager and reload the latest. Session and policy notifications, on the other hand, contain the actual data changes. Also, when using a load balancer, the notification is sent directly to the agent whose configuration has been changed. It does not go through the load balancer.

For more information see the *Sun Federated Access Manager 8.0 Administration Guide*.

Common Tasks

Federated Access Manager has implemented a Common Tasks tab that allows an administrators to create federation-based objects using console wizards. The wizards offer simplified entity provider configuration with metadata input using the URL `http://fam.sun.com:8080/fam/saml2/jsp/exportmetadata.jsp` or a metadata file. The following things can be done using a Common Task wizard:

- **Create SAML v2 Providers** They can be hosted or remote provider; and identity or service provider. To create them, you just need to provide some basic information about the providers.
- **Create Fedlet** A *Fedlet* is a small ZIP file that can be given to a service provider to allow for immediate federation with an identity provider configured with Federated Access Manager. It is ideal for an identity provider that needs to enable a service provider with no federation solution in place. The service provider simply adds the Fedlet to their application, deploys their application, and they are federation enabled.
- **Test Federation Connectivity** This task validates your federation configuration. It will show if federation connections are being made successfully by identifying where the troubles, if any, are located.
- **Access Documentation** This link opens the OpenSSO documentation page. View frequently asked questions, tips, product documentation, engineering documentation as well as links to the community blogs.
- **Register Your Product** This link allows you to register your product with Sun Connection. You must have a Sun Online Account in order to complete the registration. If you do not already have one, you may request one as part of this process.

For more information see the *Sun Federated Access Manager 8.0 Administration Guide*.

Multi-Federation Protocol Hub

Federated Access Manager allows a configured circle of trust to contain entities speaking different federation protocols thus supporting cross protocol single sign-on and logout among hosted identity providers in the same circle-of-trust. For example, you can create a circle of trust containing one identity provider instance that communicates with multiple federation protocol and three service provider instances that speak, respectively, Liberty ID-FF, SAML v2 and WS-Federation. Figure 3–4 illustrates the process of multi-federation protocol single sign-on and single logout.

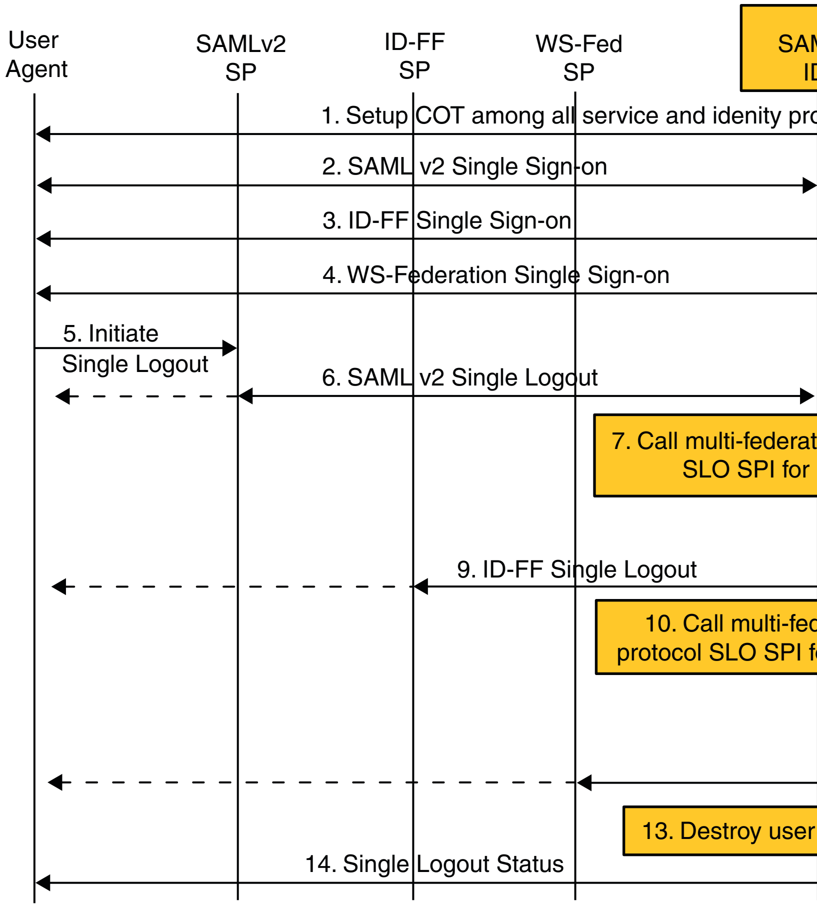

User Agent
SAMLv2 SP
ID-FF SP
WS-Fed SP
SAM
ID
1. Setup COT among all service and idenity pro
2. SAML v2 Single Sign-on
3. ID-FF Single Sign-on
4. WS-Federation Single Sign-on
5. Initiate Single Logout
6. SAML v2 Single Logout
7. Call multi-federat SLO SPI for
9. ID-FF Single Logout
10. Call multi-fec protocol SLO SPI fo
13. Destroy user
14. Single Logout Status

FIGURE 3–4 Multi-Federation Protocol Single Sign-on and Single Logout

Third Party Integration

Federated Access Manager makes it easy to integrate with third-party software. Plug-ins and other tools have been developed to ease the integration of Federated Access Manager and the following products.

- “Sun Java System Identity Manager” on page 72
- “Computer Associates SiteMinder” on page 72
- “Oracle Access Manager” on page 72

For more information, see *Sun Java System Federated Access Manager Integration Guide*.

Sun Java System Identity Manager

Sun Java System Identity Manager enables an enterprise to manage and audit access to accounts and resources as well as distribute the access management overhead. A Federated Access Manager policy agent is deployed on the Identity Manager machine to regulate access to the Identity Manager server. By mapping Identity Manager objects to Federated Access Manager users and resources, you may significantly increase operational efficiency. For use cases, a technical overview, installation and configuration procedures, architecture diagrams and process flows, see Chapter 1, “Integrating Identity Manager with Federated Access Manager,” in *Sun Java System Federated Access Manager Integration Guide*.

Computer Associates SiteMinder

Computer Associates SiteMinder (originally developed by Netegrity) is one of the industry's first SSO products — used in a majority of legacy web SSO deployments to protect their intranet and external applications. Federated Access Manager provides the tools for SSO integration with SiteMinder in both intranet and federated environments. They include a SiteMinder Agent and a Federated Access Manager Authentication Module for SiteMinder. They can be found in the `integrations/siteminder` directory of the exploded `fam.war`. For use cases, a technical overview, installation and configuration procedures, architecture diagrams and process flows, see Chapter 2, “Integrating CA SiteMinder,” in *Sun Java System Federated Access Manager Integration Guide*.

Oracle Access Manager

Oracle Access Manager (originally developed by Oblix) is an SSO product with many of the same features as Sun Federated Access Manager and Computer Associates SiteMinder. Oracle Access Manager can be deployed to protect both internal and external applications. Federated Access Manager provides an Oracle Agent and a custom Federated Access Manager Authentication Module for Oracle Access Manager. They can be found in the `integrations/oracle` directory of the exploded `fam.war`. For use cases, a technical overview, installation and configuration procedures, architecture diagrams and process flows, see Chapter 3, “Integrating Oracle Access Manager,” in *Sun Java System Federated Access Manager Integration Guide*.

PART II

Access Control Using Federated Access Manager

User authentication, authorization for access to protected resources, and programmatically defining user sessions are all aspects of access management, one of the core functions of Sun Federated Access Manager. Federated Access Manager offers access management features programmatically using the Client SDK, over the wire using HTTP and the Federated Access Manager console, and using an integrated development environment (IDE) application to incorporate Representational State Transfer (REST) calls and Web Services Definition Language (WSDL) files. The chapters in this part contain information on these aspects of access management.

CHAPTER 4

User Sessions and the Session Service

The Session Service in Sun Federated Access Manager tracks a user's interaction with web applications through the use of session data structures, session tokens, cookies, and other objects. This chapter explains these concepts and other components of a user's session and contains the following sections:

- "About the Session Service" on page 75
- "User Sessions and Single Sign-on" on page 76
- "Session Data Structures and Session Token Identifiers" on page 77

About the Session Service

The Session Service in Sun Federated Access Manager tracks a user's interaction with protected web applications. For example, the Session Service maintains information about how long a user has been logged in to a protected application, and enforces timeout limits when necessary. Additionally, the Session Service:

- 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 and cross-domain single sign-on among applications external to Federated Access Manager.
- Offers remote access to the Session Service through the Client SDK with which user sessions can be validated, updated, and destroyed.

The state of a particular session can be changed by user action or timeout. Figure 4–1 illustrates how a session is created as *invalid* before authentication, how it is activated following a successful authentication, and how it can be invalidated (and destroyed) based on timeout values.

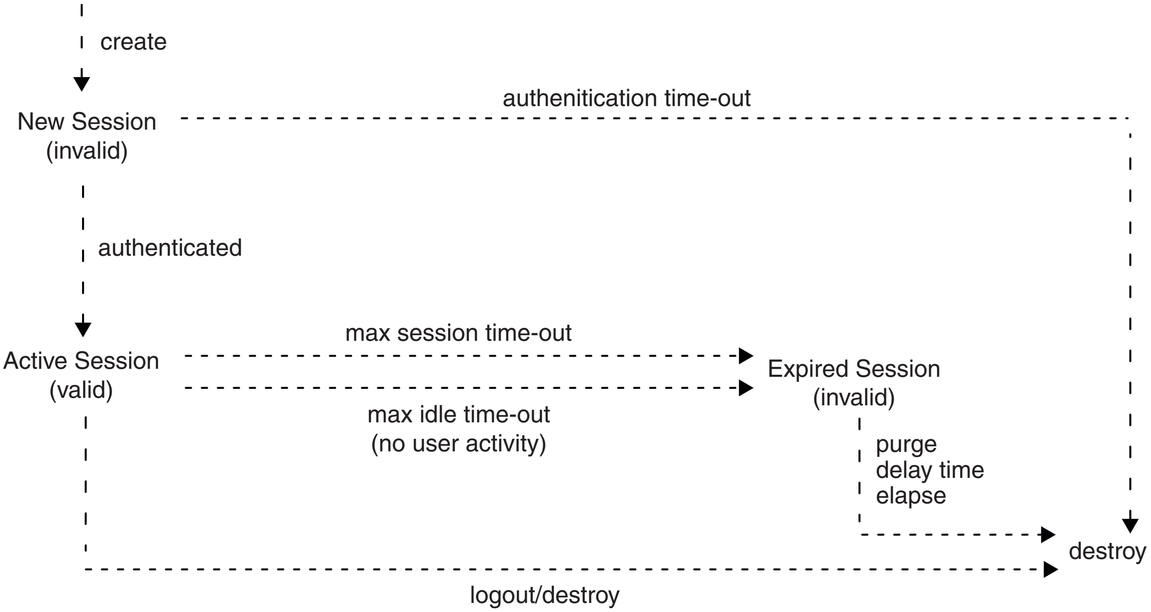


FIGURE 4–1 Life Cycle of a Session

User Sessions and Single Sign-on

A *user session* is the interval between the moment a user attempts to log in to a resource protected by Federated Access Manager, and the moment the user logs out of Federated Access Manager. As an example of a user session, an employee attempts to access the corporate benefits administration application protected by Federated Access Manager. A new invalid session is created, and the Authentication Service prompts the user for a username and password to verify the user's identity. Following a successful authentication, the Policy Service and policy agent work together to check that the user has the appropriate permissions to access the protected application and allows or denies access based on the outcome.

Oftentimes, in the same user session (without logging out of the corporate benefits application), the same employee might attempt to access a corporate expense reporting application. Because the expense reporting application is also protected by Federated Access Manager, the Session Service provides proof of the user’s authentication, and the employee is allowed to access the expense reporting application (based on the outcome of a second authorization check with the

Policy Service). If access is granted, the employee has accessed more than one application in a single user session without having to reauthenticate. This is called *single sign-on* (SSO). When SSO occurs among applications in more than one DNS domain, it is called *cross-domain single sign-on* (CDSSO). For a more detailed overview of a basic user session, an SSO session, and a CDSSO session, see Chapter 5, "Models of the User Session and Single Sign-On Processes."

Session Data Structures and Session Token Identifiers

The Session Service programmatically creates a session data structure to store information about a user session. The result of a successful authentication results in the validation of a *session data structure* for the user or entity and the creation of a *session token* identifier. The session data structure minimally stores the following information.

Identifier	A unique, universal identifier for the session data structure.
Host Name or IP Address	The location from which the client (browser) is making the request.
Principal	Set to the user's distinguished name (DN) or the application's principal name.
Type	USER or APPLICATION
Session State	Defines whether the session is valid or invalid.
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.

A session can also contain additional 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 Federated Access Manager to include the following types of data in a session:

- Protected properties are only modifiable by the server-side modules (primarily the Authentication Service).
- Custom properties are modifiable remotely by any application which possesses the session identifier.

For a detailed summary of information that can be included in a session, see Chapter 10, "Configuring Federated Access Manager Sessions," in *Sun Federated Access Manager 8.0 Installation and Configuration Guide*.

The session token, also referred to as a *sessionID* and programmatically as an SSOToken, is an encrypted, unique string that identifies the session data structure. As the user visits different protected resources using the browser, the session token is propagated to these resources and is used to retrieve the user's credentials. These credentials are then validated by sending a back-end request (using the Client SDK or a policy agent) to Federated Access Manager which then returns an error or the session's prior authentication data. Sessions (and hence the SSOToken) are invalidated when a user logs out, the session expires, or a user in an administrative role invalidates it. With Federated Access Manager, a session token is carried in a *cookie*, an information packet generated by a web server and passed to a web browser. (The generation of a cookie for a user by a web server does not guarantee that the user is allowed access to protected resources. The cookie simply points to information in a data store from which an access decision can be made.)

Note – Access to some Federated Access Manager services, such as the Policy Service and the Logging Service, require presentation of both the SSOToken of the application as well as the SSOToken of the user, allowing only designated applications to access these services.

CHAPTER 5

Models of the User Session and Single Sign-On Processes

This chapter traces events in a basic user session, a single sign-on session (SSO), a cross-domain single sign-on session (CDSSO), and session termination to give you an overview of the features and processes being invoked. It contains the following sections:

- "Basic User Session" on page 79
- "Single Sign-On Session" on page 89
- "Cross-Domain Single Sign-On Session" on page 91
- "Session Termination" on page 93

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 Federated Access Manager. In these examples, the server which hosts an application is protected by a policy agent. The Basic User Session includes the following phases:

- "Initial HTTP Request" on page 79
- "User Authentication" on page 81
- "Session Validation" on page 84
- "Policy Evaluation and Enforcement" on page 85
- "Logging the Results" on page 87

Initial HTTP Request

When a user initiates a user session by using a browser to access and log in to a protected web-based application, the events illustrated in Figure 5–1 occur. The accompanying text describes the model.

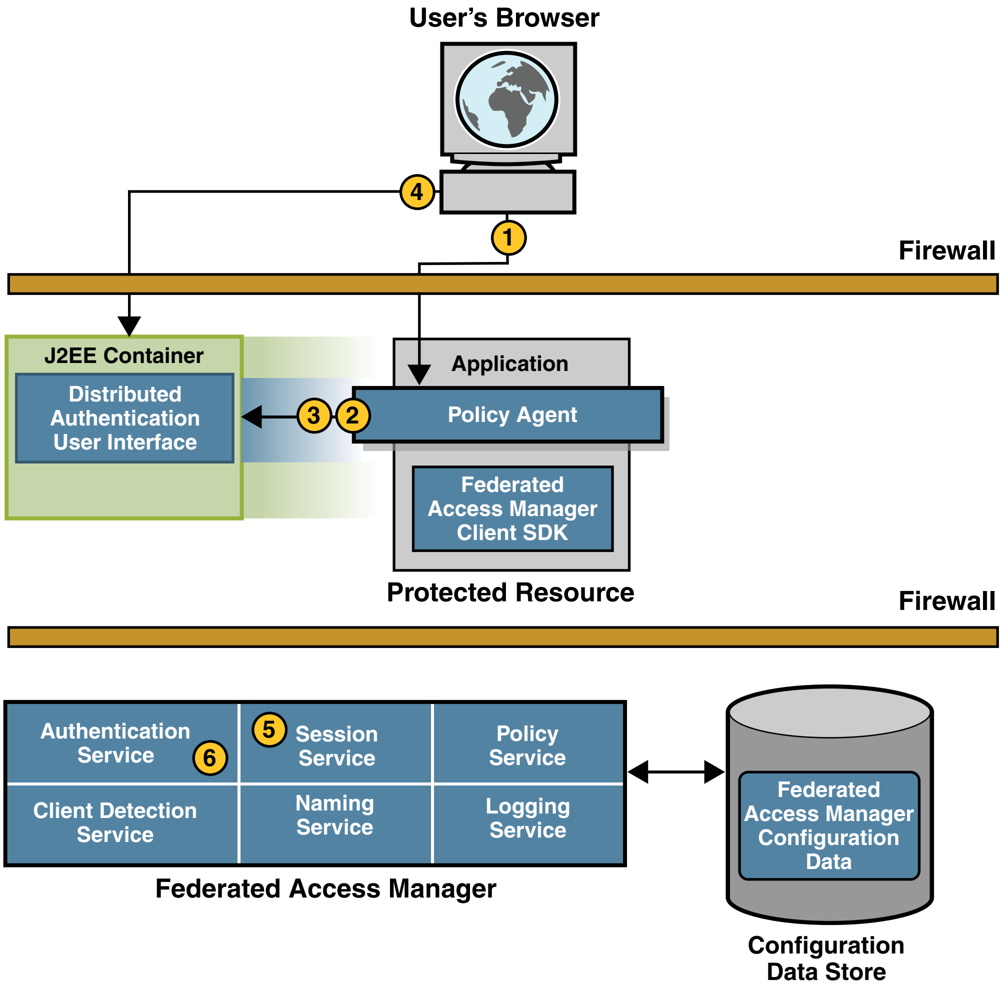


FIGURE 5–1 Initial HTTP Request

1. The user's browser sends an HTTP request to the protected resource.
2. The policy agent that protects the resource intercepts and inspects the user's request and finds no session token.
3. The policy agent issues a redirect to it's configured authentication URL to begin the authentication process.

   In this example, the authentication URL it is set to the URL of the Distributed Authentication User Interface.
4. The browser, following the redirect, sends a GET request for authentication credentials to the Distributed Authentication User Interface.

5. The Session Service creates a new session (session data structure) and generates a session token (a randomly-generated string that identifies the session).
6. The Authentication Service sets the session token in a cookie.

The next part of the user session is "User Authentication" on page 81.

User Authentication

When the browser sends the GET request to the Distributed Authentication User Interface, the events illustrated in Figure 5–2 occur.

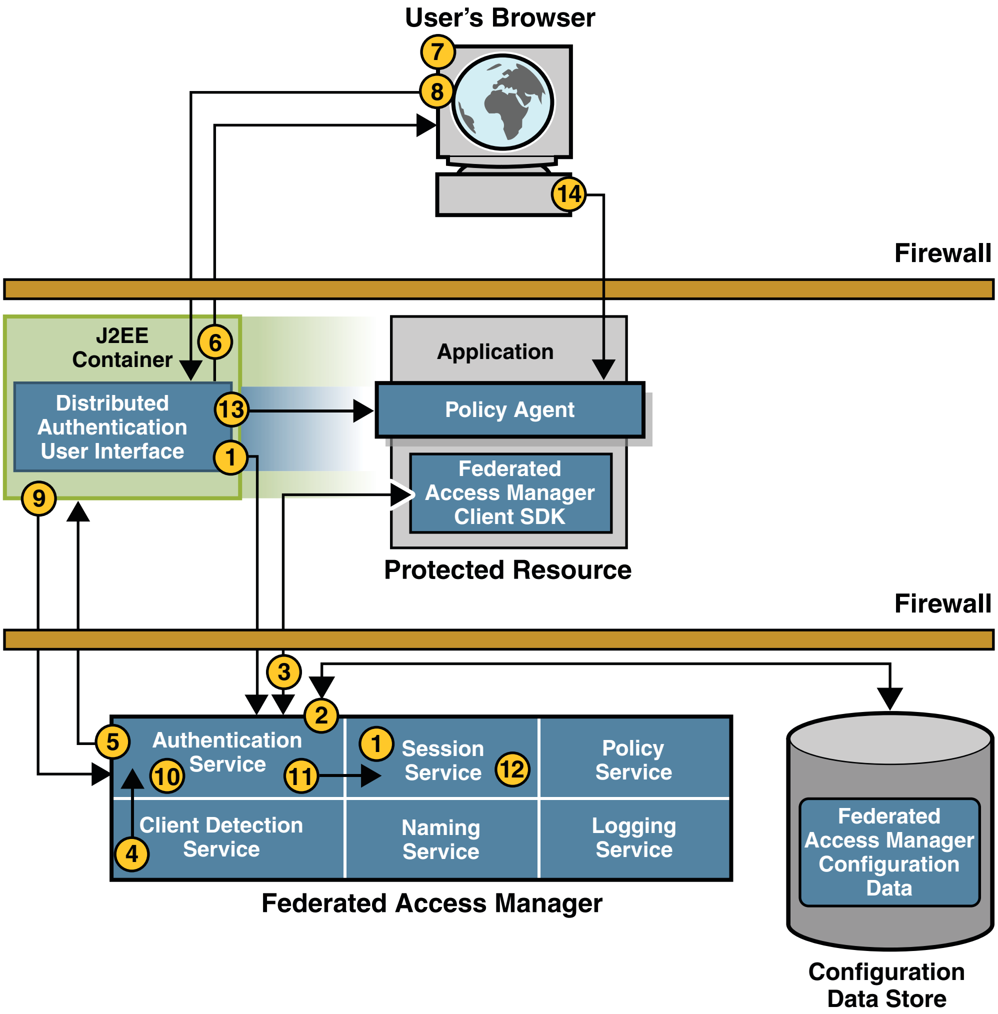


FIGURE 5–2 User Authentication

, and sends back to the Distributed Authentication User Interface all necessary credentials, requirements, and callbacks for use by the presentation framework layer.

1. Using the parameters in the GET request, the Distributed Authentication User Interface contacts the Federated Access Manager Authentication Service (which, in turn, communicates with the Session Service).
2. The Authentication Service determines what should be presented to the user based upon configuration data and retrieves the appropriate authentication module(s) and callback(s) information.

For example, if configured to use LDAP Authentication, the Authentication Service determines that the LDAP Authentication login page should be displayed.

3. The collected information is passed to the Distributed Authentication User Interface using the Client SDK.
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 generates a dynamic presentation extraction page that contains the appropriate credentials request and callbacks information obtained from Federated Access Manager.

   The session cookie will be included in this communication.
6. The user's browser displays the login page.
7. The user enters information in the fields of the login page.
8. The browser sends the credentials in an HTTP POST to the Distributed Authentication User Interface.
9. The Distributed Authentication User Interface uses the Client SDK to pass the credentials to the Authentication Service.
10. The Authentication Service uses the appropriate authentication module to validate the user's credentials.

    For example, if LDAP authentication is used, the LDAP authentication module verifies that the username and password provided exist in the LDAP directory.
11. 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.
12. Once the session is activated, the Session Service changes the state of the session token to valid.
13. The Distributed Authentication User Interface replies to the protected resource with the validated `SSOToken` in a set-cookie header.
14. Now, the browser makes a second 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" on page 84.

Session Validation

After successful authentication, the user's browser redirects the initial HTTP request to the server a second time for validation. The request now contains a session token in the same DNS domain as Federated Access Manager. The events in Figure 5–3 illustrate this process.

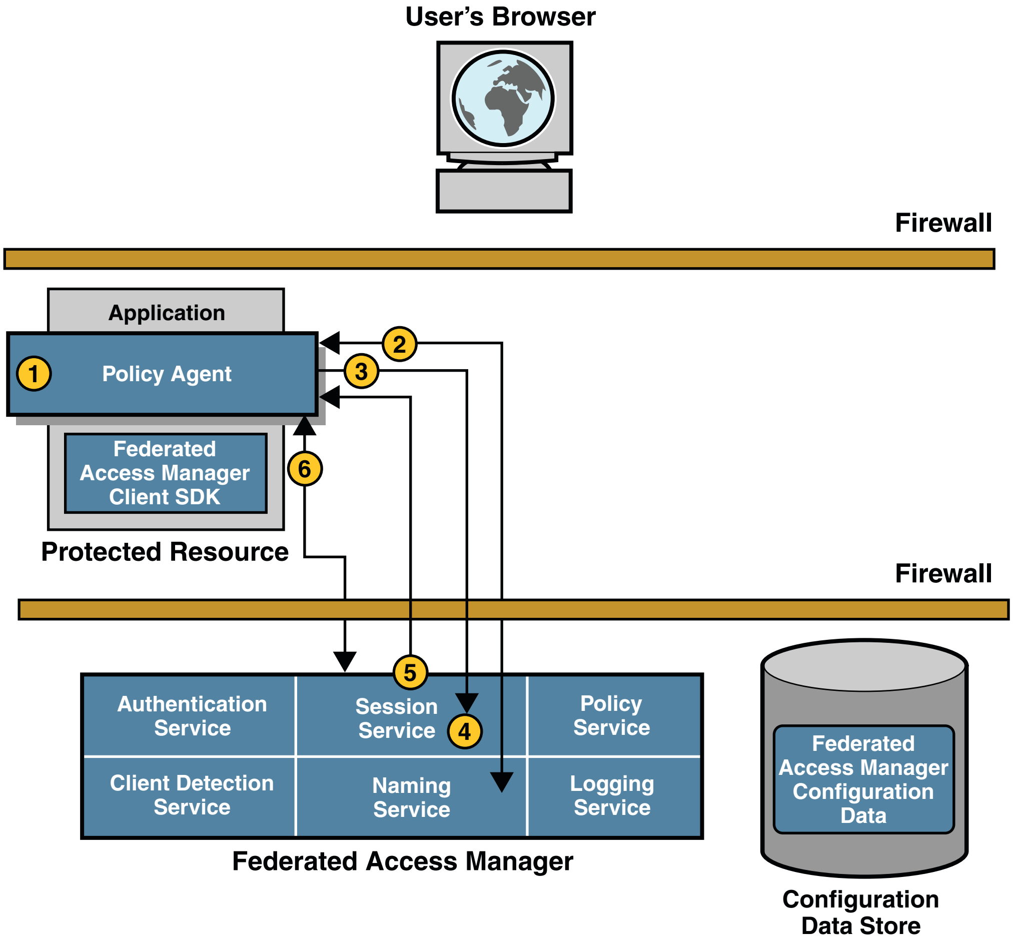


FIGURE 5–3 Session Validation

1. The policy agent intercepts the second access request.
2. To determine the validity of the session token, the policy agent contacts the Naming Service to learn where the session token originated.

The Naming Service allows clients to find the URL for internal Federated Access Manager services. When contacted, the Naming Service decrypts the session token and returns the corresponding URL which can 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 it with the Session Service, enabling notification to be sent 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 and Enforcement” on page 85.

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. Figure 5–4 illustrates this process.

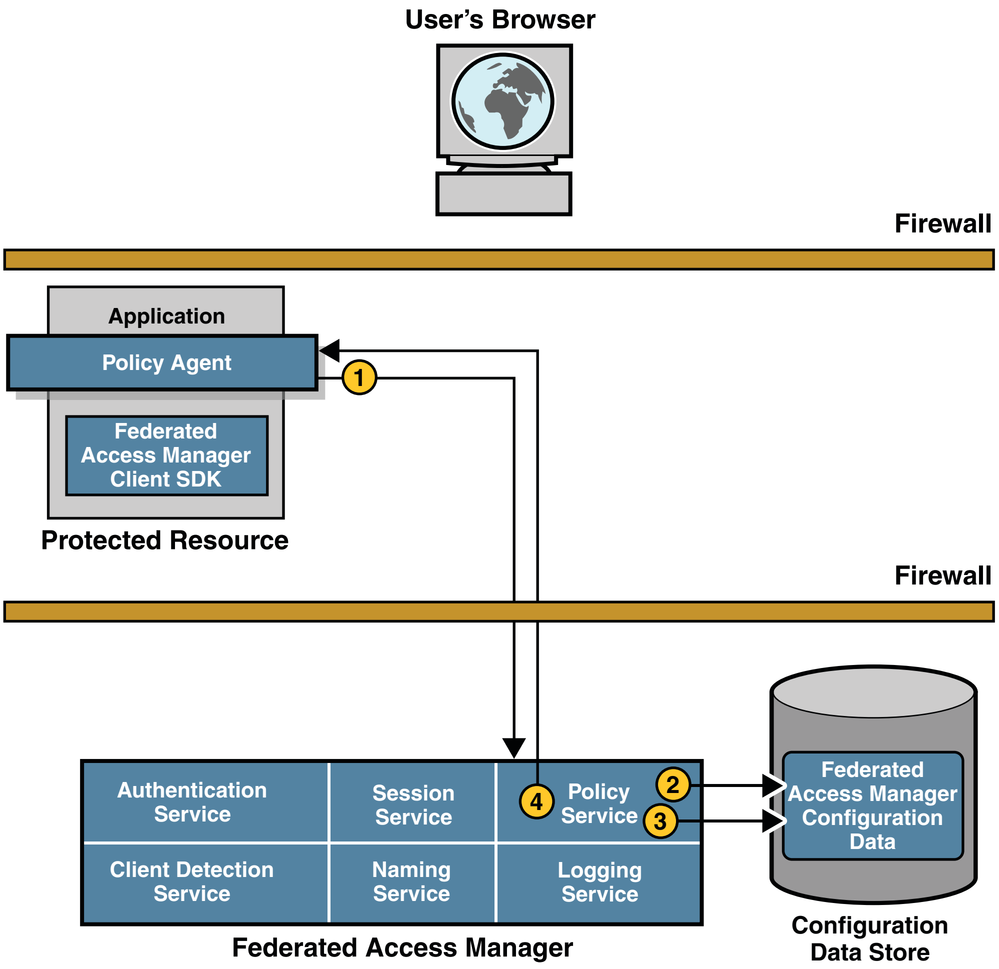


FIGURE 5–4 Policy Evaluation

1. The policy agent sends a request to the Policy Service, asking for decisions regarding resources in it'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 Federated Access Manager. If the policies have not been cached already, they are loaded from Federated Access Manager.
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 will be denied access.
   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.

The next part of the basic user session is "Logging the Results" on page 87.

Logging the Results

When the policy agent receives a decision from the Policy Service, the events illustrated in Figure 5–5 occur.

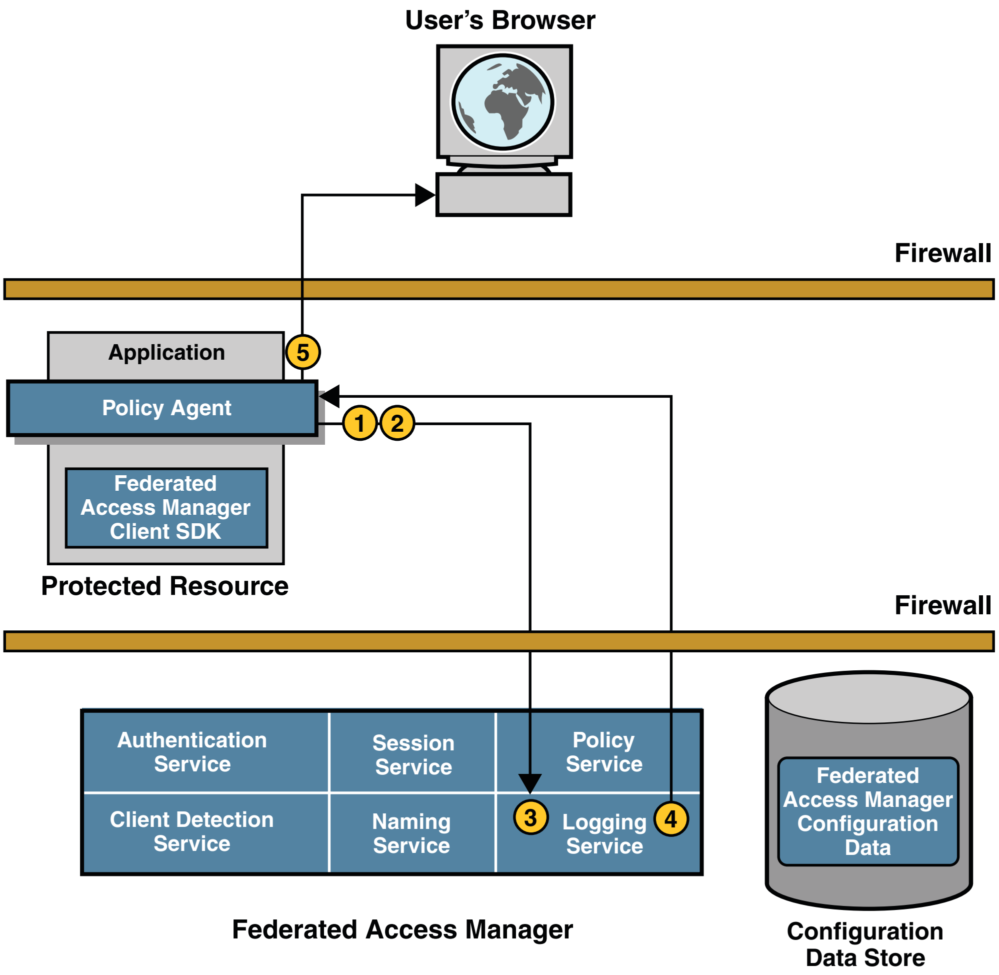


FIGURE 5–5 Logging the Policy Evaluation Results

1. The decision and session token are cached by the policy agent so subsequent requests can be checked using the cache (without contacting Federated Access Manager).

   The cache will expire after a (configurable) interval has passed or upon explicit notification of a change in policy or session status.

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 93 for more information. While logged in, if the user attempts to log into another protected resource, the “Single Sign-On Session” on page 89 begins.

Single Sign-On Session

Remark 5–1 Reviewer **There are 13 bullets in graphic and nine steps? Huh????? What should be changed?**

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 process describes an SSO session by tracking 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 79.

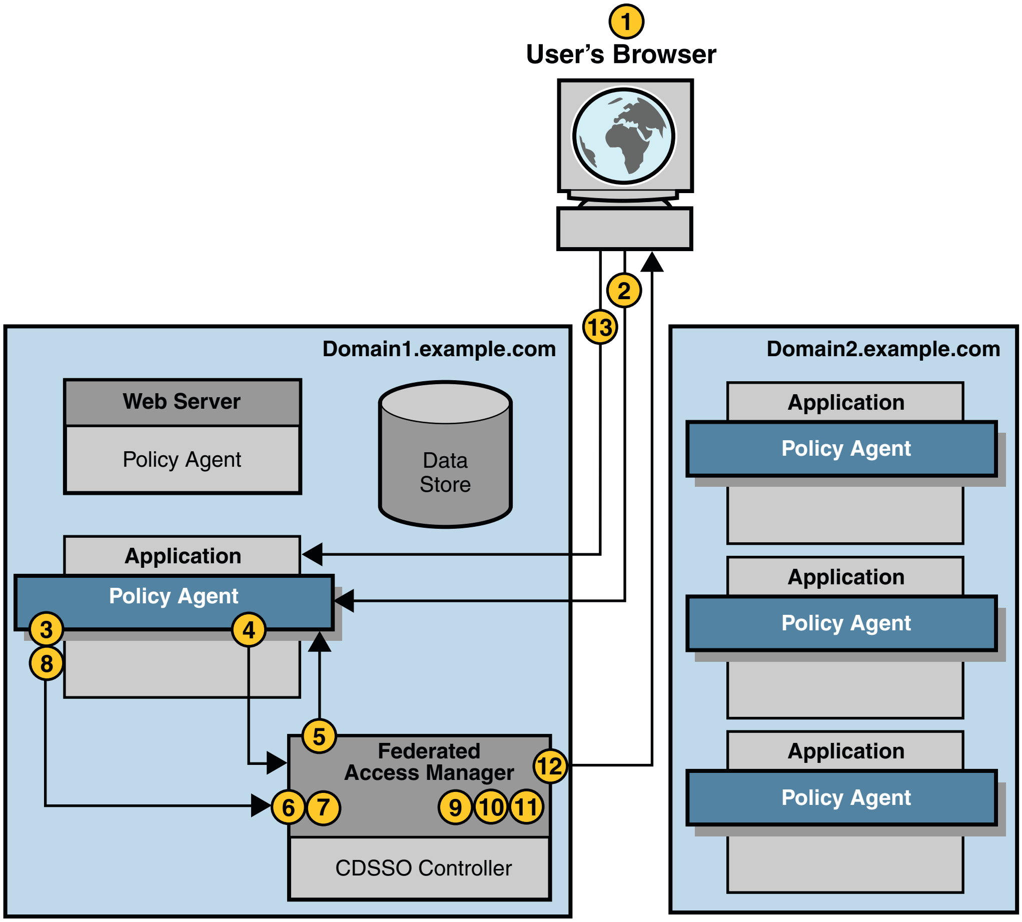


FIGURE 5–6 Single Sign-On Session

1. The user attempts to access a second application hosted on a server in the same domain as the first application to which authentication was successful.
2. The user's browser sends an HTTP request to the second application that includes the user's session token.
3. The policy agent intercepts the request and inspects it to determine whether a session token exists.

   A session token indicates the user is already authenticated. Since the user was authenticated, the Authentication Service is not required at this time. The Session Service API retrieve the session data using the session token identifier imbedded in the cookie.
4. The policy agent sends the session token identifier to the Federated Access Manager Session Service to determine whether the session is valid or not.

For detailed steps, see “Session Validation” on page 84.

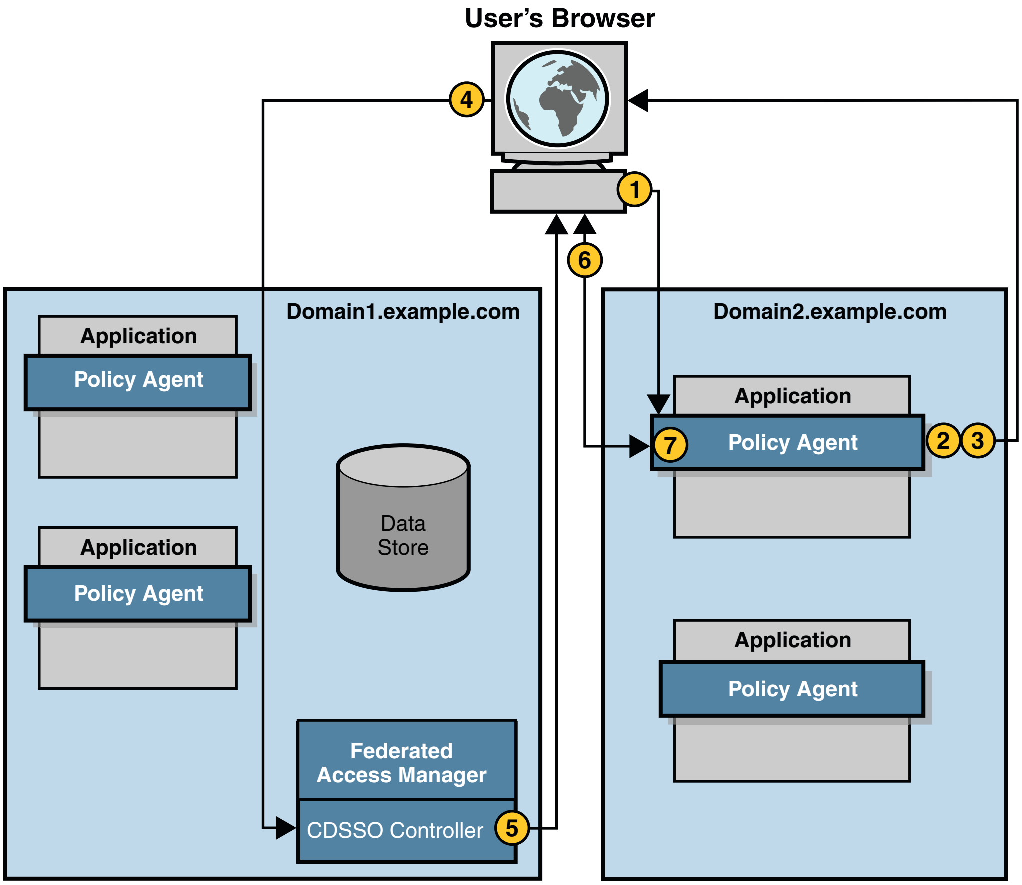
5. The Session Service sends a reply to the policy agent indicating whether the session is valid.
   - If the session is not valid, the user is redirected to the Authentication page.
   - If the session is valid, the Session Service creates a Session Listener.
6. As the session is valid, the Session Service creates a Session Listener.

   A Session Listener sends notification to the policy agent when a change in the session state occurs.
7. The policy agent sends a request for a decision regarding resources in it’s portion of the HTTP namespace to the Policy Service.
8. 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 Logging Service logs a denial of access and the policy agent issues a *Forbidden* message to the user. The user can then be redirected to an administrator-specified page indicating that access was denied.
   - If the Policy Service finds policy allowing access to the protected resource, the Logging Service logs the access, the user is granted access, and the session is valid until it is terminated.
9. If Policy Service does not find policy allowing access to the protected resource, the user is denied access, the Logging Service logs a denial of access. and 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.
10. 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 93.

While still logged in, if the user attempts to log in to another protected resource located in a different DNS domain, the “Cross-Domain Single Sign-On Session” on page 91 begins.

Cross-Domain Single Sign-On Session

CDSSO occurs when an authenticated user requests a protected resource on a server in a **different** DNS domain. The user in the previous sections, “Basic User Session” on page 79 and “Single Sign-On Session” on page 89, for example, accessed applications in one DNS domain. In this scenario, the CDSSO Controller within Federated Access Manager transfers the user’s session information from the initial domain, making it available to applications in a second domain.



1. The authenticated user's browser sends an HTTP request to the application in a different DNS domain.
2. The policy agent intercepts the request and inspects it to determine if a session token exists for the domain *in which the requested 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) will redirect the HTTP request to the CDSSO Controller.

     The CDSSO Controller 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.
4. The user's browser allows the redirect to the CDSSO Controller.

Recall that earlier in the user session the session token was set in a cookie in the first domain which is now part of the redirect.

5. The CDC Servlet (in the CDSSO Controller) receives the session token from the first domain, extracts the user's session information, formulates a Liberty POST profile response containing the information, and returns a response to the browser.
6. The user's browser automatically submits the response to the policy agent in the second domain.

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

The process continues with "Policy Evaluation and Enforcement" on page 85 and "Logging the Results" on page 87. Based on the policy outcome, the user is granted or denied access to the application.

1. If the user is denied access, the policy agent displays an "access denied" page.
2. If the user is granted access, the protected resource displays its access page. The new cookie can now be used by all agents in the new domain, and the session is valid until it is terminated.

Session Termination

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

- "User Ends Session" on page 93
- "Administrator Ends Session" on page 94
- "Federated Access Manager Enforces Timeout Rules" on page 94
- "Session Quota Constraints" on page 94

User Ends Session

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

1. The Logout Service receives the Logout request, and:
   a. Marks the user's session as destroyed.
   b. Destroys the 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 the cache and the user session ends.

Administrator Ends Session

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

1. The Logout Service receives the Logout request, and:
   a. Marks the user's session as destroyed.
   b. Destroys the 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.

Federated Access Manager Enforces Timeout Rules

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

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

Session Quota Constraints

Federated 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 6

Authentication and the Authentication Service

The Sun Federated 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 components to prove that the user's identity is genuine. Topics covered in this chapter include:

Authentication Service Overview

The function of the Authentication Service is to request information from an authenticating party, and validate it against the configured identity repository using the specified authentication module. After successful authentication, the user session is activated and can be validated across all web applications participating in an SSO environment. For example, when a user or application attempts to access a protected resource, credentials are requested by one (or more) authentication modules. Gaining access to the resource requires that the user or application be allowed based on the submitted credentials. From the user perspective, a company employee wants to look up a colleague's phone number. The employee 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. Federated Access Manager compares the user's input with data stored in the appropriate identity repository. If Federated Access Manager finds a match for the user name, and if the given password matches the stored password, the user's identity is authenticated.

Note – The "Basic User Session" on page 79 section in the previous chapter contains a detailed description of the authentication process itself.

The Authentication Service can be accessed by a user with a web browser, by an application using the Client SDK, or by any other client that correctly implements the Authentication Service messaging interfaces. The Authentication Service framework has a pluggable architecture for authentication modules that have different user credential requirements. Together with the Session Service, the Authentication Service establishes the fundamental infrastructure for SSO. Generally speaking, the Authentication Service:

- Identifies a requester's credential requirements.
- Generates a dynamic user interface based on the requirements of the authentication module being called.
- Supports custom, pluggable authentication modules.
- Provides pre- and post-processing SPI.
- Populates and manages system domain cookies.
- Generates time dependent alerts and session termination notifications.
- Provides a remote user interface application for distributed deployments.
- Implements a clean logout interface which destroys the session.

Every time a request is used to access the Authentication Service, the session token identifier is retrieved and used to get the associated session data structure from the Session Service. Additionally, the Authentication Service interfaces with the Session Service to:

- Initiate or create user sessions.
- Maintain session state information.
- Activate sessions after successful authentication.
- Populate the valid session data structure with all user-authenticated identity data and properties.
- Destroy sessions after logout.

The following diagram illustrates how the two services work together.

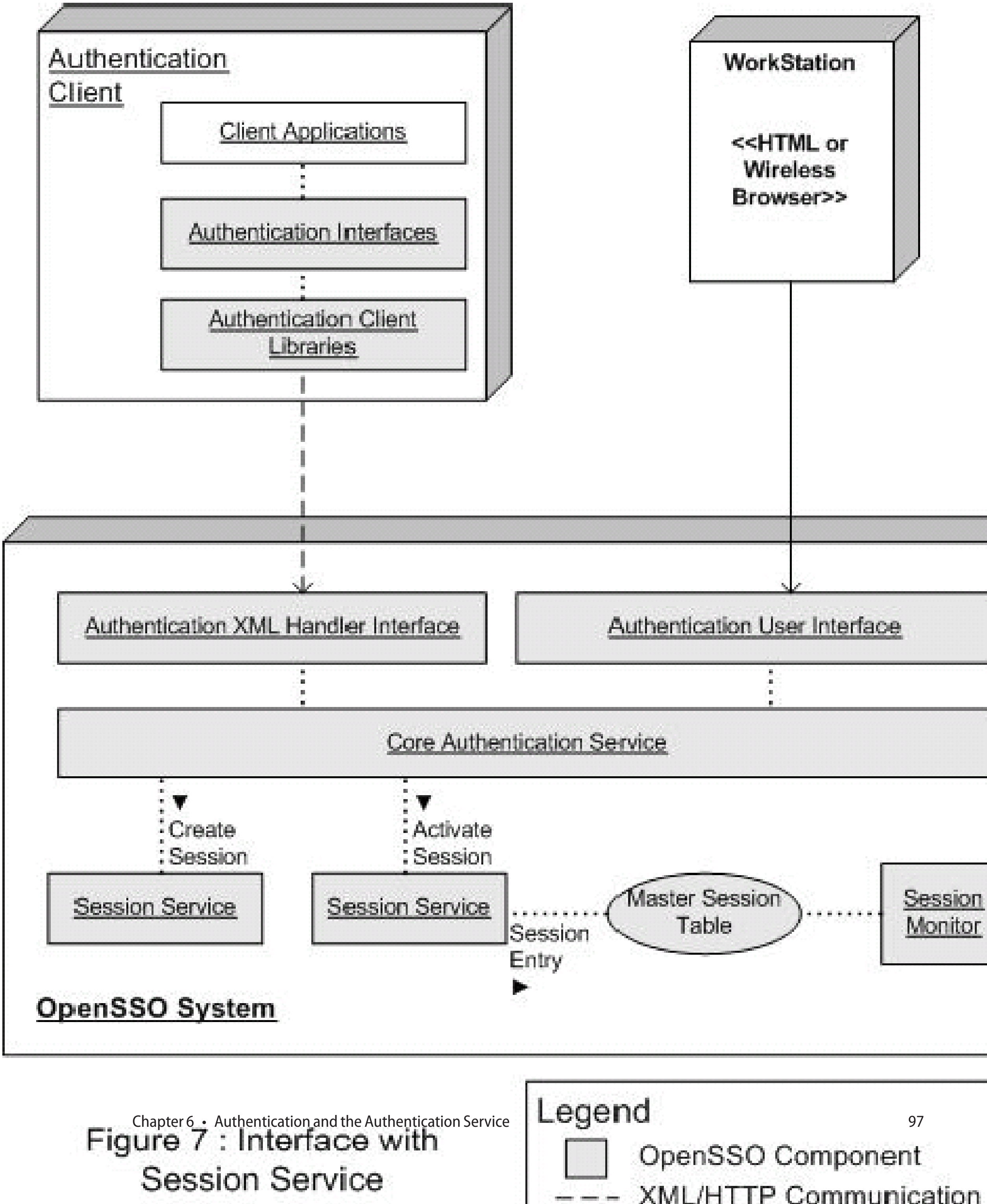


Figure 7 : Interface with Session Service

The Authentication Service also interfaces with other Federated Access Manager services including the Naming Service, the Identity Repository Service, the Logging Service, and the Monitoring Service. It also interfaces with the configuration data store and policy agents protecting system resources. (A policy agent must authenticate itself using the Client SDK authentication interfaces, and users with no valid session must be authenticated.)

Authentication Service Features

The following sections explain some of the features of the Authentication Service.

- "Client Detection" on page 98
- "Account Locking" on page 98
- "Authentication Chaining" on page 99
- "Fully Qualified Domain Name Mapping" on page 100
- "Persistent Cookies" on page 100
- "Session Upgrade" on page 100
- "JAAS Shared State" on page 101
- "Security" on page 101

Client Detection

Because the Authentication Service is client-type aware, the initial step in the authentication process is to identify the type of client making the HTTP(s) request. This 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 web browser is used for requesting access, an HTML login page will be displayed. Once the user is authenticated, the client type is added to the session token for future use. For more information, see Chapter 11, "Identifying the Client Type," in *Sun Federated Access Manager 8.0 Developer's Guide*.

Note – Federated Access Manager supports all configured client types including cookie-less and cookie-enabled.

Account Locking

The Authentication Service provides account locking to prevent a user from completing the authentication process after a specified number of failures. Federated Access Manager sends email notifications to administrators when account lockouts occur. Federated Access Manager supports:

Physical Locking. By default, user accounts are `active` or physically unlocked. You can initiate physical locking by changing the value of the Lockout Attribute

Name 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 authentication attempts.

The account locking feature is disabled by default. Account locking activities are also logged. For information on how to enable it, see "Account Locking" in *Sun Federated Access Manager 8.0 Administration Guide*.

Note – Only authentication modules that throw an `Invalid Password Exception` can leverage the Account Locking feature.

Authentication Chaining

Federated Access Manager allows the configuration of an authentication process in which a user must pass credentials to one or more authentication modules before session validation is accomplished. This is called *authentication chaining*. Federated Access Manager uses the Java Authentication and Authorization Service (JAAS) framework (integrated with the Authentication Service) to implement authentication chaining. The JAAS framework validates all user identifiers used during the authentication process, and maps them to one principal. (The mapping is based on the configuration of the User Alias List attribute in the user's profile.) If all the maps are correct, the session token is validated. 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 configure an authentication chain by realm, user, role, or service. Determining validation is based upon the control flag configured for each authentication module instance defined in the chain. The flags are:

Requisite Authentication to this module instance is required to succeed. If it succeeds, authentication continues down the module instance list. If it fails, control immediately returns to the application.

Required Authentication to this module instance is required to succeed. If any of the required module instances defined in the chain fails, the whole authentication chain will fail.

Sufficient — The module instance is not required to succeed. If it does succeed, control immediately returns to the application (authentication does not proceed down the module instance list). If it fails, authentication continues down the list.

Optional — The module instance is not required to succeed. Whether it succeeds or fails, authentication still continues to proceed down the module instance list.

For more information, see “Authentication Modules and Chains” in *Sun Federated Access Manager 8.0 Administration Guide*. For an overview of the authentication module instances, see “Authentication Modules” on page 101.

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 can also be used to allow access to one instance of Federated Access Manager using many different aliases. For example, you might configure one instance of Federated 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 Federated Access Manager 8.0 Administration Guide*.

Persistent Cookies

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. The Authentication Service can be enabled to write persistent cookies rather than cookies that are written to a web browser's memory. For more information, see “Persistent Cookie” in *Sun Federated Access Manager 8.0 Administration Guide*.

Session Upgrade

The Authentication Service allows for the upgrade of a valid session based on a second, successful authentication performed by the same user. If a user with a valid session 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 in 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 Federated Access Manager 8.0 Administration Guide*.

JAAS Shared State

The JAAS shared state enables sharing of both a user identifier and a 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 Federated Access Manager 8.0 Administration Guide*.

Security

From a security point of view, here are some general practices implemented in the Authentication Service.

- SSL is strongly recommended to prevent the user credentials from being stolen through passive network snooping.
- The signing and encryption of some user data is to prevent other software applications, sharing the same system resources, from subverting it.
- The main user entry points of the Authentication Service (Distributed Authentication User Interface, Authentication XML Handler Interface for remote clients, the Authentication Service User Interface) are protected by entry level validation of the size of the requested data.
- Creation and modification of authentication configuration information is only allowed by privileged Federated Access Manager administrators.

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 validated and, assuming the appropriate policy configuration, granted access to the requested resource. If the user provides information that does not meet the authentication criteria, the user is not validated and denied access to the requested resource. Federated Access Manager is deployed with a number of authentication modules. Table 6–1 provides a brief description of each.

TABLE 6–1 Authentication Service Modules

Authentication Module Name	Description
Active Directory	Uses an Active Directory operation to associate a user identifier 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 Federated 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.
Federation	Used by the service provider during federation (using SAML v1.x, SAML v2, WS-Federation, Liberty ID-FF) to create a session after validating the assertion. This authentication module can not be invoked like the other modules as it is invoked directly by the `SAMLAwareServlet`.
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.
LDAP	Enables authentication using LDAP bind, a directory server operation which associates a user identifier and password with a particular LDAP entry. You can define multiple LDAP authentication configurations for a realm.

TABLE 6–1 Authentication Service Modules *(Continued)*

Authentication Module Name	Description
Membership	Enables user to self-register a user entry. The user creates 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 by Federated 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 Federated Access Manager console to enable and configure the authentication modules. 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 Federated Access Manager authentication framework. See Chapter 4, "Managing Authentication," in *Sun Federated Access Manager 8.0 Administration Guide* for detailed information about enabling and configuring default authentication modules and authentication module instances. See Chapter 2, "Using the

Authentication Interfaces," in *Sun Federated Access Manager 8.0 Developer's Guide* for more information about writing custom authentication modules.

Authentication Types

After granting or denying access to a resource, Federated Access Manager checks for information about where to redirect the user. A specific order of precedence is used when checking for 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 Federated Access Manager, a number of authentication types are automatically configured.

Realm-based Authentication.	User authenticates to a configured realm or sub-realm.
Role-based Authentication.	User authenticates to a configured role within a realm or sub-realm. The user must possess the role. A *static role* is possessed when an attribute is assigned to a specific user or container. A *filtered role* is dynamically generated based on an attribute contained in the user's or container's entry. For example, all users that contain a value for the `employee` attribute can be included in a role named *employees* when the filtered role is created.
Service-based Authentication.	User authenticates to a specific service or application registered to a realm or sub-realm.
User-based Authentication.	User authenticates using an authentication process configured specifically for him or her.
Authentication Level-based Authentication	An administrator specifies the security level of the authentication modules by defining each with an *authentication level*. Successful authentication to a higher authentication level defines a higher level of trust for the user. If a user attempts to access a service, the service can determine if the user is allowed access by checking the authentication level in the user's session data. If the authentication level is not high enough, the service redirects the user to go through an authentication process with a set authentication level.
Module-based Authentication.	Allows a user to specify the module to which they will authenticate.

Organization-based Authentication.	User authenticates to an organization or sub-organization. **Note –** This authentication type only applies to Federated Access Manager when installed in Legacy mode.

For more information, see "Authentication Types" in *Sun Federated Access Manager 8.0 Administration Guide*.

Configuring for Authentication

The authentication framework includes the following places where you can configure for authentication:

- "Core Authentication Module and Realm Configuration" on page 105
- "Authentication Configuration Service" on page 106
- "Login URLs and Redirection URLs" on page 106

Explanations of the authentication attributes can be found in the Online Help and the Part II, "Configuration Attribute Reference," in *Sun Federated Access Manager Administration Reference*.

Core Authentication Module and Realm Configuration

The Core Authentication Module contains general authentication properties that can be defined globally using the Federated Access Manager console (under the Configuration tab) or more specifically for each configured realm (under the Access Control tab). Core authentication properties are added and enabled for the top-level realm during installation. As new realms are configured under the top-level realm, these properties (and the values defined globally for them) are dynamically added to each new realm when it is created. Once added, new values can be defined and configured values can be modified by the realm's administrator. The values are then used if no overriding value is defined in the specified authentication module instance or authentication chain. The default values for the Core Authentication Module are defined in the `amAuth.xml` file and stored in the configuration data store. For more information, see "General Authentication Properties" in *Sun Federated Access Manager 8.0 Administration Guide* and the Part II, "Configuration Attribute Reference," in *Sun Federated Access Manager Administration Reference*.

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 Federated Access Manager 8.0 Administration Guide*.

Login URLs and Redirection URLs

In the last phase of the authentication process, Federated Access Manager either grants or denies access to the user. If access is granted, Federated Access Manager uses a login URL to display a page in the browser. If access is denied, Federated 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 (as discussed in "Authentication Types" on page 104) 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 Federated Access Manager proceeds through the order of precedence, see "Authentication Types" in *Sun Federated Access Manager 8.0 Administration Guide*.

Authentication Graphical User Interfaces

The Federated Access Manager Authentication Service has two separate graphical user interfaces that can be used. The following sections contain information on them.

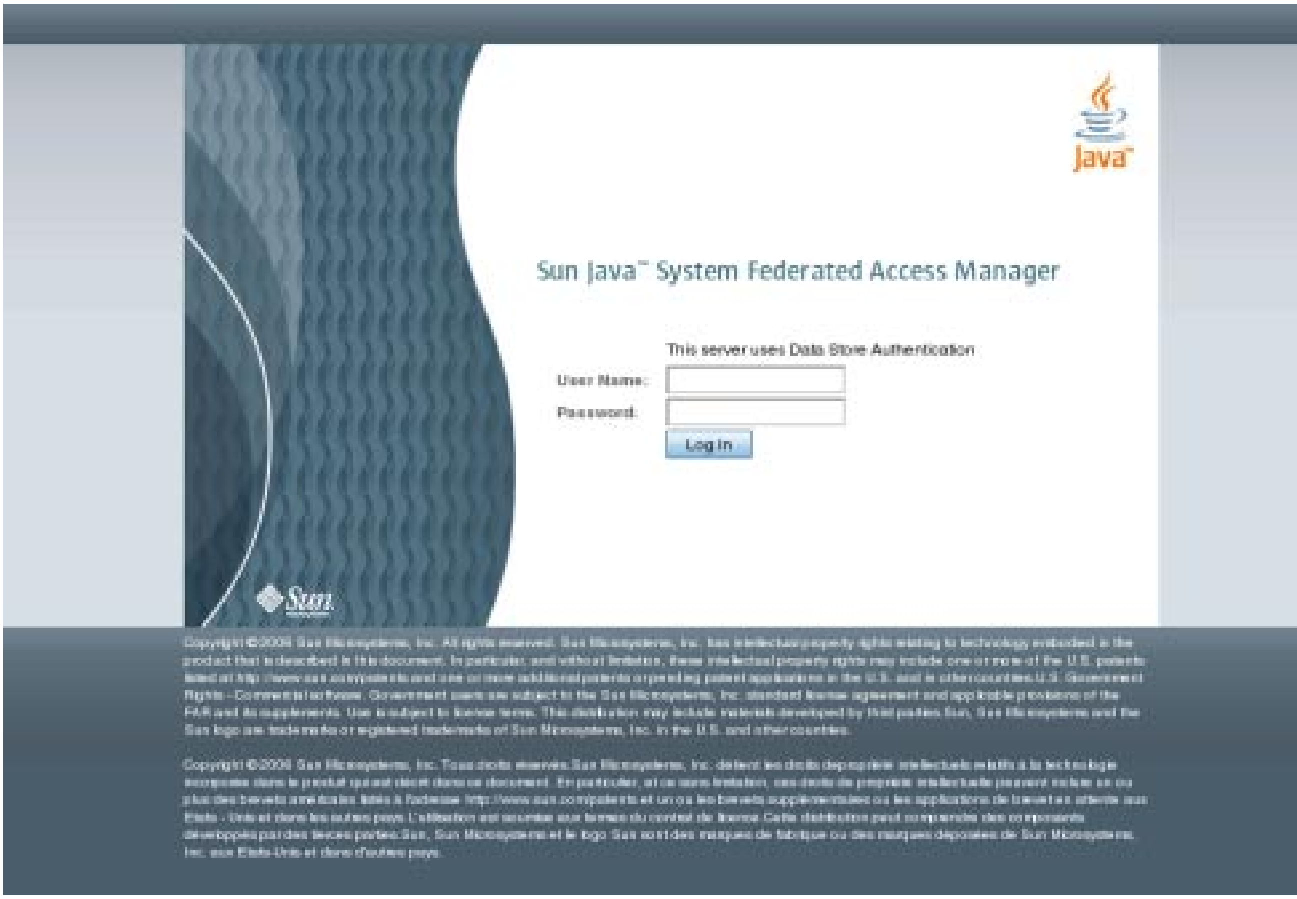
- "Authentication Service User Interface" on page 106
- "Distributed Authentication User Interface" on page 108

Authentication Service User Interface

The Authentication Service implements a user interface that is separate from the Federated 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 the appropriate credentials based on the configured authentication module or chain. Once the credentials have been passed back to Federated Access Manager and authentication is deemed successful, the user may gain access to the protected resource if authorized to do so. The Authentication Service user interface can be used for the following:

- Administrators can access the administration portion of the Federated 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.

Below is a screen capture of the default Authentication Service user interface.



Sun Java™ System Federated Access Manager

This server uses Data Store Authentication

User Name:

Password:

Copyright © 2008 Sun Microsystems, Inc. All rights reserved. Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at http://www.sun.com/patents and one or more additional patents or pending patent applications in the U.S. and in other countries.U.S. Government Rights - Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements. Use is subject to license terms. This distribution may include materials developed by third parties.Sun, Sun Microsystems and the Sun logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Copyright © 2008 Sun Microsystems, Inc. Tous droits réservés.Sun Microsystems, Inc. détient les droits de propriété intellectuels relatifs à la technologie incorporée dans le produit qui est décrit dans ce document. En particulier, et ce sans limitation, ces droits de propriété intellectuelle peuvent inclure un ou plus des brevets américains listés à l'adresse http://www.sun.com/patents et un ou les brevets supplémentaires ou les applications de brevet en attente aux Etats - Unis et dans les autres pays. L'utilisation est soumise aux termes du contrat de licence.Cette distribution peut comprendre des composants développés par des tierces parties.Sun, Sun Microsystems et le logo Sun sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays.

FIGURE 6–2 Authentication Service User Interface

Federated Access Manager provides customization support for the Authentication Service user interface. You can customize JavaServer Pages™ (JSP™) and the file directory level by organization, service, locale, or client type. See Chapter 16, “Customizing the Authentication User Interface,” in *Sun Federated Access Manager 8.0 Developer’s Guide* for more information.

Distributed Authentication User Interface

Federated Access Manager also 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 the appropriate module 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 with Federated Access Manager. The Distributed Authentication User Interface enables a policy agent or an application that is deployed in a non-secured area to communicate with the Federated Access Manager Authentication Service installed in a secured area of the deployment. Figure 6–3 illustrates this scenario.

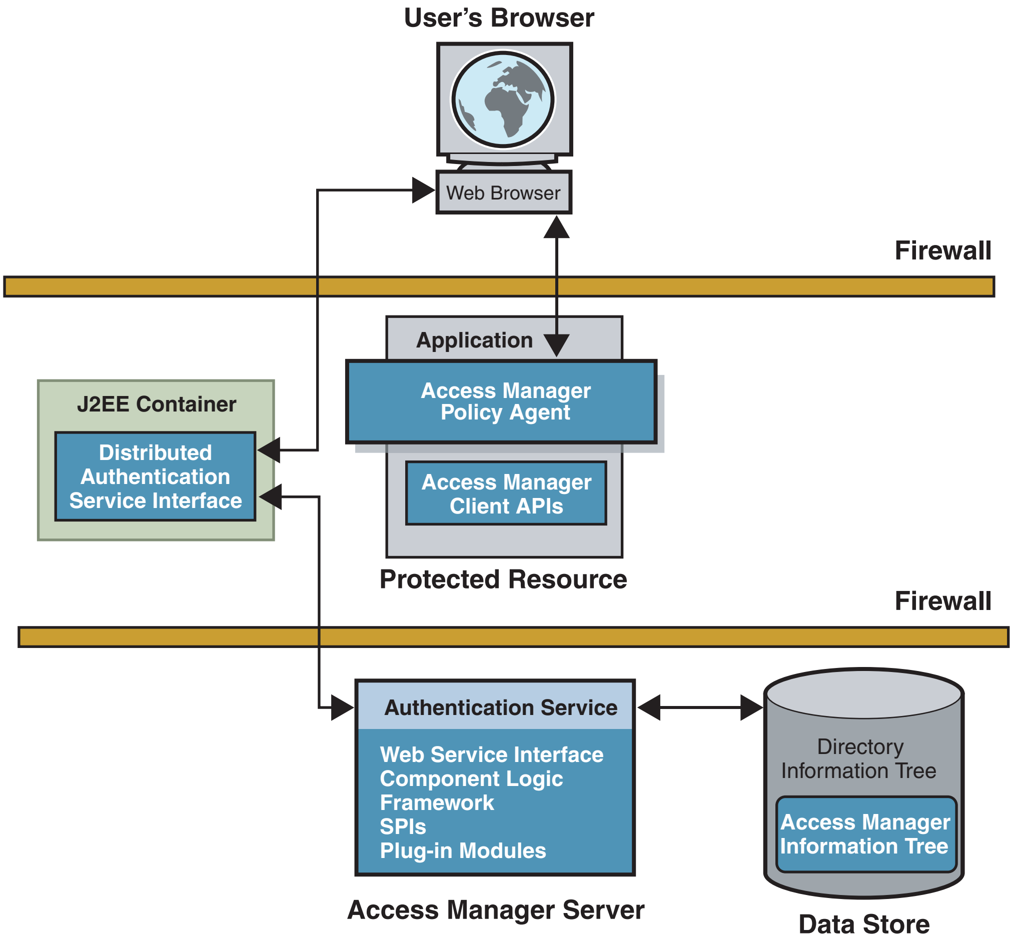


FIGURE 6–3 Distributed Authentication Process

The Distributed Authentication User Interface uses a JATO presentation framework and is customizable. (See screen capture in "Authentication Service User Interface" on page 106.) You can install the Distributed Authentication User Interface on any servlet-compliant web container within the non-secure layer of a Federated 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, see "User Authentication" on page 81. For detailed installation and configuration instructions, see Chapter 6, "Deploying a Distributed Authentication UI Server," in *Sun Federated Access Manager 8.0 Installation and Configuration Guide*.

Authentication Service Programming Interfaces

Federated 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 use the XML/HTTP interface directly to initiate an authentication request. Additionally, you can add custom authentication modules to Federated Access Manager by using the service provider interface (SPI) package, `com.iplanet.authentication.spi`. 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 the Authentication Interfaces,” in *Sun Federated Access Manager 8.0 Developer’s Guide* and *Federated Access Manager 8.0 Java API Reference*.

Federated Access Manager also provides a Client SDK that can implement authentication logic on a remote web server or application server. For information, see Chapter 1, “Enhancing Remote Applications Using the Client Software Development Kit,” in *Sun Federated Access Manager 8.0 Developer’s Guide*.

CHAPTER 7

Authorization and the Policy Service

The Sun Federated 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 *authorization*. This chapter describes how the various parts of the Policy Service work together to perform authorization. Topics covered include:

Authorization and Policy Service Overview

A *policy* is a rule that defines who is authorized to access a resource. A single policy can define authorization with 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, the Policy Service allows administrators to configure, modify, and delete policies. The configured policies are then added to a *realm* and applied against the subjects in the realm. The Policy Service can be accessed using the Policy Service API: a privileged user can define access control policies using the administration API while a protected application or policy agent can obtain policy decisions using the evaluation API. The Policy Service relies on:

- A Policy Administration Point (PAP) implements the functionality to define policies. The Policy Service is the PAP.
- A Policy Enforcement Point (PEP) to protect an enterprise's resources by enforcing access control. The PEP uses the policy component of the Client SDK to retrieve policy decisions. The policy agent is the PEP.

- A Policy Decision Point (PDP) to evaluate policy and make an access determination. The Policy Service is the PDP.
- A data store in which configured policies are stored and from which they are retrieved. The Configuration Data Store is the data store.

Access to a resource is always preceded by a basic user session in which the requestor is authenticated, a session is created by the Authentication Service, and the session token identifier is validated. (See Chapter 5, "Models of the User Session and Single Sign-On Processes.") The policy agent protecting the resource then provides the session token identifier, resource name, desired action, and additional context parameters to the Policy Service which uses configured policies to determine if the user has been given permission to access the protected resource by a recognized authority. When the policy agent gets the decision from the Policy Service, it allows or denies access to the user, enforcing the policy decision provided by Policy Service. This whole process is referred to as *authorization*. The Policy Service is defined by the `amPolicy.xml` and, generally speaking:

- 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.
- Supports the delegation of policy management.
- Provides an SPI for extensibility.
- Provides access from remote clients using the Client SDK.
- Caches and reuses policy decisions, where applicable, to improve performance.
- Allows periodic polling of the Policy Service by a client to update locally cached policy decisions.
- Dynamically recognizes changes to policies and provides policy decisions that reflect them.

Note – The Policy Configuration Service provides a means to specify how policies are defined and evaluated. It enables you to specify, for example, which directory to use for subject lookup, the directory password, which search filters to use, and which subjects, conditions, and response providers to use. This configuration can be done within a realm or a subrealm and is accessible using the Federated Access Manager console.

See Chapter 5, "Managing Policies," in *Sun Federated Access Manager 8.0 Administration Guide* and Chapter 3, "Enforcing Authorization with the Policy Service," in *Sun Federated Access Manager 8.0 Developer's Guide* for more information.

Policy Types

The Policy Service authorizes access to a user based on policies created and stored in the Federated Access Manager configuration data store. The following sections contain information on the two types of policies you can create.

- "Normal Policy" on page 113
- "Referral Policy" on page 116

For more information, see Chapter 5, "Managing Policies," in *Sun Federated Access Manager 8.0 Administration Guide*.

Normal Policy

A *normal policy* specifies a protected resource and who is allowed to access it. The protected resource can be anything hosted on a protected server. Examples of protected resources are applications, document files, images, or the server itself. A normal policy consists of *rules*, *subjects*, *conditions*, and *response providers*. The following sections contain information on these elements.

- "Rules" on page 113
- "Subjects" on page 113
- "Conditions" on page 114
- "Response Providers" on page 115

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 the user or collection of users that the policy affects. The following list of subjects can be assigned to policies.

Access Manager Identity Subjects
The identities you create and manage under the Subjects tab in a configured realm can be added as a value of the subject.

Authenticated Users

Any user with a valid session (even if they have authenticated to a realm that is different from the realm in which the policy is defined) is a member of this subject. This is useful if the resource owner would like to allow access to users from other organizations. To restrict a resource's access to members of a specific organization, use the Organization subject.

Web Services Clients

This implies that a web service client (WSC) identified by a session token identifier is a member of this subject — as long as the distinguished name (DN) of any principal contained in the session token identifier matches any selected value of this subject.

The following list of subjects can only be specified after they are selected using the Policy Configuration Service of the appropriate realm.

Federated Access Manager Roles

Any member of a Federated Access Manager role is a member of this subject. A Federated Access Manager role is created using Federated Access Manager running in legacy mode. These roles have object classes mandated by Federated Access Manager and can only be accessed through the hosting Federated Access Manager Policy Service.

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 member of a realm is a member of this subject.

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. Federated Access Manager allows for the following list of 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).

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. Federated Access Manager includes one implementation of the `com.sun.identity.policy.interfaces.ResponseProvider` class, the `IDResponseProvider`.

See Chapter 3, "Enforcing Authorization with the Policy Service," in *Sun Federated Access Manager 8.0 Developer's Guide* for more information.

Referral Policy

A user with the Top—level Realm Administrator or Policy Administrator roles can create policy. (A Realm Administrator or Policy Administrator configured for a specific realm have permission to create policies only for resources delegated to that realm.) A *referral policy* enables either 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 of which policy creation or evaluation is being referred.
- A *referral* defines the identity 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, assume a top-level realm exists named `ISP`. It contains two subrealms: `company1` and `company2`. The Top-Level Realm 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 Realm Administrator creates two referral policies, defining the appropriate realm in the rule and the appropriate administrator in the referral. See "Creating Policies" in *Sun Federated Access Manager 8.0 Administration Guide* for more information.

Realms and Access Control

When a user logs into an application, Federated Access Manager plug-ins retrieve all user information, authentication properties, and authorization policies that the Federated 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 within a *realm*. You create a realm when you want to apply policies to a group of related subjects, services or servers. The Policy Configuration Service within the realm provides a means to specify how policies are defined and evaluated. It enables you to specify, for example, which directory to use for subject lookup, the directory password, which search filters to use, and which subjects, conditions, and response providers to use. 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. These configurations can be done within a realm or a sub realm and is accessible using the Federated Access Manager console.

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

Policy Service Programming Interfaces

Federated Access Manager provides both Java API and C API for writing clients that remote applications can use to administer policies and evaluate policy decisions. They are used to add, lookup, modify or replace policies, and to evaluate policy decisions when a principal attempts an action on a protected resource. Communication between the API and the Policy Service occurs by sending XML messages over HTTP(S). Additionally, you can extend and customize the Policy Service using the SPI. The classes are used by service developers and policy administrators who need to provide additional policy features as well as support for legacy policies. For example, you can develop customized plug-ins for creating custom policy subjects, referrals, conditions, and response providers. Lastly, the Client SDK is provided to implement policy evaluation logic on a remote web server or application server. For information, see Chapter 1, “Enhancing Remote Applications Using the Client Software Development Kit,” in *Sun Federated Access Manager 8.0 Developer's Guide*, Chapter 3, “Enforcing Authorization with the Policy Service,” in *Sun Federated Access Manager 8.0 Developer's Guide*, the *Sun Federated Access Manager 8.0 C API Reference*, and the *Federated Access Manager 8.0 Java API Reference*.

XACML Service

XACML is an access control language that provides an XML syntax for defining policies (who can do what, where can it be done, and when), for querying whether access to a protected resource can be allowed (requests), and for receiving responses to those queries (decisions). XACML is built around the standard access control separation of the Policy Enforcement Point (PEP) and the Policy Decision Point (PDP) as discussed in “Authorization and Policy Service Overview” on page 111 except using XACML queries and responses. The XACML PEP is responsible for intercepting all access requests, collecting the appropriate information (such as who is making the request, which resource is being accessed, and what action is to be taken), and sending a request for a decision to the XACML PDP. The XACML PDP (Federated Access Manager) evaluates configured policies against the information in the decision request. It uses a Context Handler to request the appropriate policies and attributes in order to render one of the following decisions.

- Permit
- Deny
- Not Applicable (no policy created by this PDP applies to the access request)
- Indeterminate (an error occurred that prevents the PDP from knowing the correct response)

The following sections contain more information.

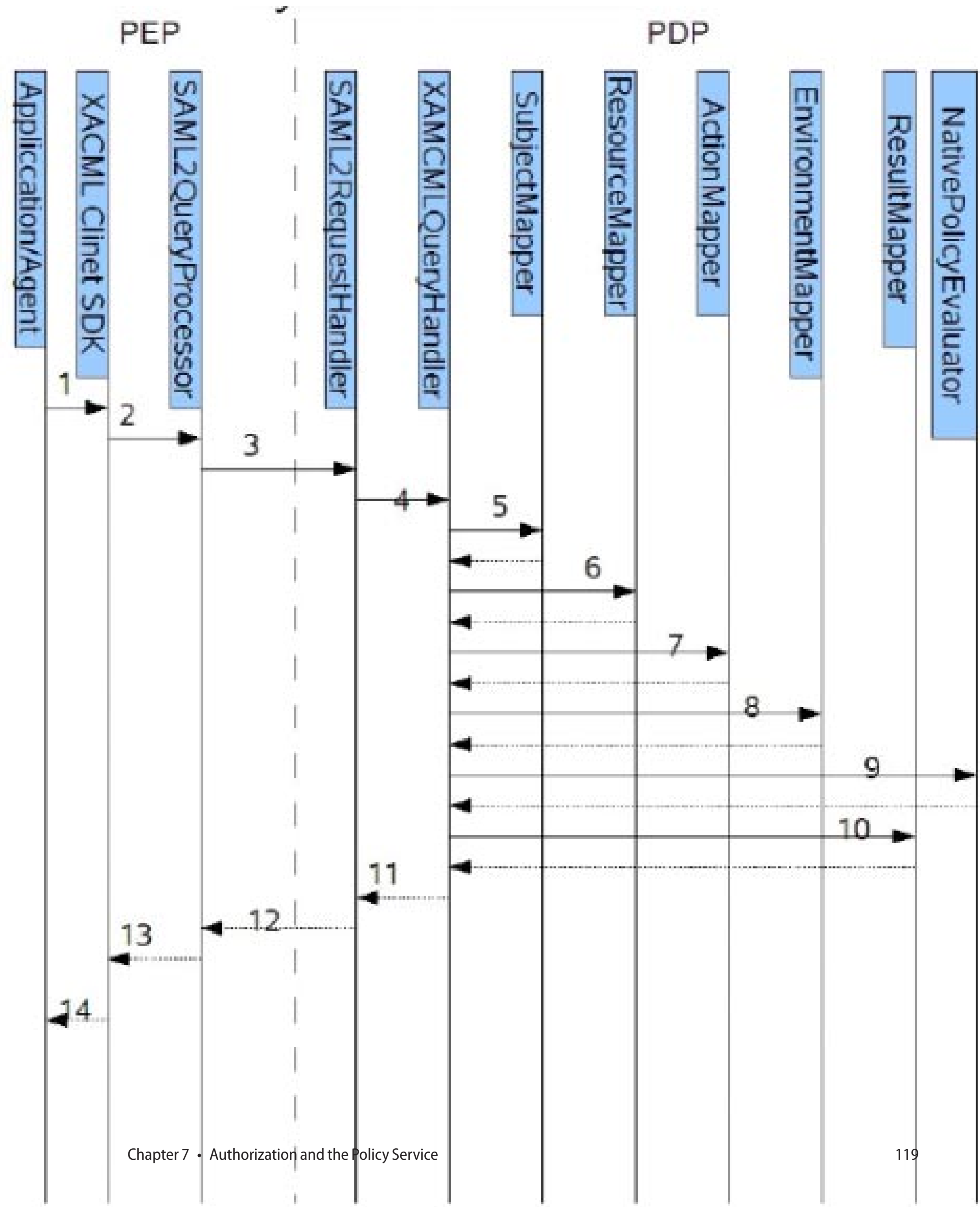
- "XACML in Federated Access Manager" on page 118
- "XACML Programming Interfaces" on page 121

XACML in Federated Access Manager

Federated Access Manager implements the SAML v2 Profile of the eXtensible Access Control Markup Language (XACML) version 2.0 - supporting `XACMLAuthzDecisionQuery` and `XACMLAuthzDecisionStatement`. In a Federated Access Manager XACML interaction, after receiving a request for access, the XACML PEP makes a `XACMLAuthzDecisionQuery` request and receives a `XACMLAuthzDecisionStatement` response that contains the decision. (The policies themselves are not returned.) The XACML components on the client side include Client SDK interfaces for passing XACML requests and receiving XACML responses as well as an interface to construct the communications.

Note – The framework relies internally on the Client SDK SAML v2 interfaces for communication between the PEP and PDP, and includes an implementation of the SAML v2 request handler called the `XACML2AuthzDecisionQueryHandler` that plugs into the SAML v2 Service framework.

The XACML components on the Federated Access Manager side include out-of-the-box implementations of XACML mappers for subjects, resources, actions and environment. These implementations use the Policy Service to compute authorization decisions. Figure 7–1 illustrates how XACML and Federated Access Manager interact with each other. The communications are explained more fully following the image.


PEP
PDP
Appliccation/Agent
XACML Clinet SDK
SAML2QueryProcessor
SAML2RequestHandler
XAMCMLQueryHandler
SubjectMapper
ResourceMapper
ActionMapper
EnvironmentMapper
ResultMapper
NativePolicyEvaluator
1
2
3
4
5
6
7
8
9
10
11
12
13
14

1. The policy agent protecting a resource constructs a XACML access request using the Client SDK.
2. The Client SDK wraps the request in a `XACMLAuthzDecisionQuery` element and sends it to the SAML v2 query processor on the local machine (also part of the Client SDK).
3. The SAML v2 query processor consults the metadata for the PEP and the PDP, sets additional elements or attributes in the query, signs it (if necessary) and sends a SOAP request containing the query to the PDP.
4. The SAML v2 request handler on the PDP side receives the request, consults the metadata for the PEP and the PDP, verifies the trust relationships, enforces any signing or encryption requirements, verifies the signature and forwards the query to the `XACMLAuthzDecisionQueryHandler`.
5. The `XACMLAuthzDecisionQueryHandler` consults the appropriate metadata using the `entityID` values of the PEP and PDP (included in the request) to find the correct mapper implementations to use.
6. `XACMLAuthzDecisionQueryHandler` uses the Resource mapper to map the given Resource to a resource and service configured with Federated Access Manager.
7. `XACMLAuthzDecisionQueryHandler` uses the Action mapper to map the given Action to an action name configured with Federated Access Manager.
8. `XACMLAuthzDecisionQueryHandler` uses the Environment mapper to map the given Environment to conditions configured with Federated Access Manager.
9. `XACMLAuthzDecisionQueryHandler` uses the Federated Access Manager policy evaluator to get the policy decision.
10. `XACMLAuthzDecisionQueryHandler` uses the Result mapper to map the decision to an XACML Result element.

Note – Federated Access Manager is not an XACML policy engine. It has no support for XACML policies themselves and thus no support for retrieving the policies, only the decision.

11. `XACMLAuthzDecisionQueryHandler` wraps the XACML Result in an XACML Response, the XACML Response in an `XACMLAuthzDecisionStatement`, the `XACMLAuthzDecisionStatement` in a SAML Assertion, the Assertion in a SAML Response, and hands over the SAML Response to the SAML v2 request handler.
12. The SAML v2 request handler sets additional attributes and elements (based on the SAML v2 protocol), signs it as required and returns it in a SOAP message to the PEP side.
13. The SAML v2 query processor verifies the trust relationships, the signing requirements, and the signature as necessary. It then extracts the SAML Response from the SOAP message and returns it to the XACML portion of the Client SDK.

14. The Client SDK extracts the XACML Response from the SAML v2 Response and returns it (and the decision) to the client application.

XACML Programming Interfaces

Federated Access Manager provides Java API for using, and interacting with, the XACML Service. For information, see Chapter 1, “Enhancing Remote Applications Using the Client Software Development Kit,” in *Sun Federated Access Manager 8.0 Developer’s Guide*, Chapter 3, “Enforcing Authorization with the Policy Service,” in *Sun Federated Access Manager 8.0 Developer’s Guide*, and the *Federated Access Manager 8.0 Java API Reference*.

PART III

Federation Management Using Federated Access Manager

This third part of the Sun Federated Access Manager Technical Overview contains information on the Federation features of Federated Access Manager. It contains the following chapters:

PART IV

Delivering Identity Services, Web Services and Web Services Security

This fourth part of the Sun Federated Access Manager Technical Overview contains information on implementing identity services, web services and web services security. It contains the following chapters:

PART V

Additional Features

This final section of the Sun Federated Access Manager Technical Overview contains information on the Logging Service and third-party product information. It contains the following chapters:

Index