Sun Java System 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
June 2008

080422@19860

List of Remarks

Contents

CHAPTER 1

Introducing Federated Access Manager

Sun Java^{TM} System Federated Access Manager integrates authentication and authorization services, single sign-on, and open, standards-based federation protocol 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 7
- "What Does Federated Access Manager Do?" on page 8
- "What Are the Core Functions of Federated Access Manager?" on page 9
- "What Else Does Federated Access Manager Offer?" on page 11

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 as well as new features and 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. It is delivered as a simple web archive (WAR) that can be easily deployed as a self-contained Java Platform, Enterprise Edition (Java EE) application in a supported web container.

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

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.

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* 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.

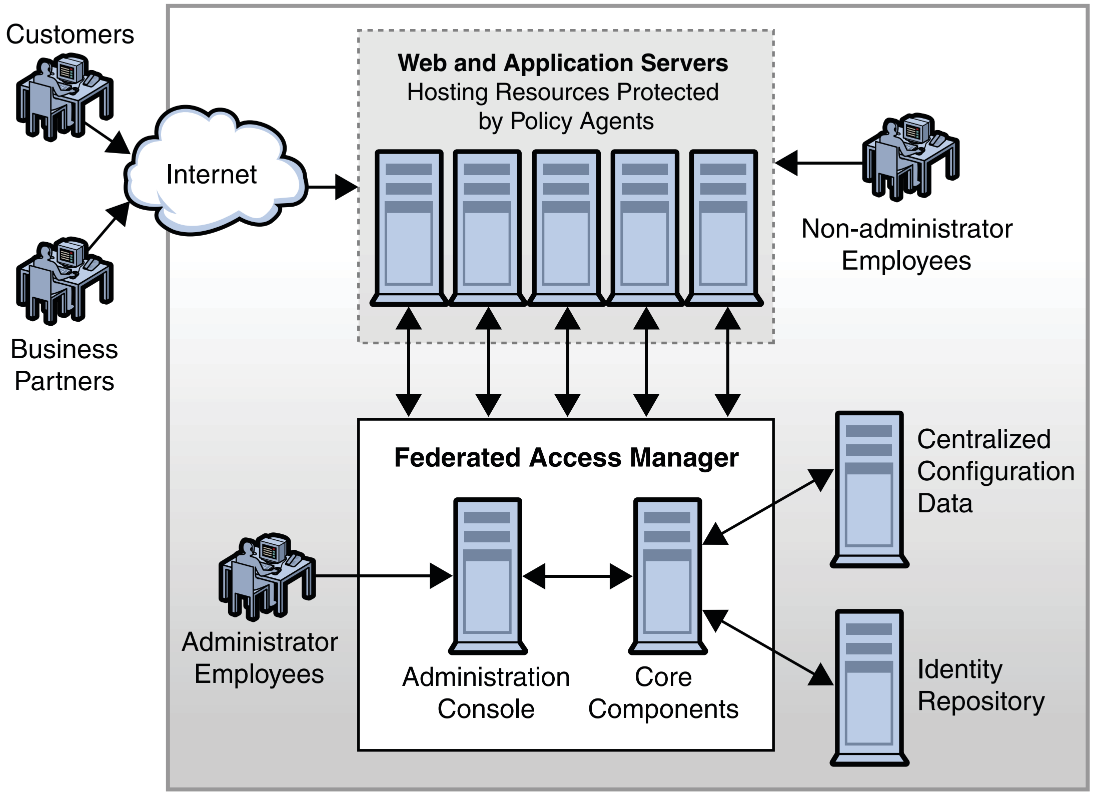


FIGURE 1–1 High-level Deployment Architecture of Federated Access Manager

What Are the Core Functions of Federated Access Manager?

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

- "Access Management" on page 9
- "Federation Management" on page 10
- "Web Services and Web Services Security" on page 10
- "Identity Services" on page 11

Access Management

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 is 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.

Federation Management

With the introduction of federation protocol 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 re-authenticate; 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.

Web Services and Web Services Security

A *web service* is an application that exposes some type of business or infrastructure functionality through a language-neutral and platform-independent, callable interface. In particular, the service defines its interface using the Web Services Description Language (WSDL), and communicates using SOAP and eXtensible Markup Language (XML) messages. (The message being exchanged is defined by the WSDL as published by the WSP.) The web service client (WSC) communicates with the web service provider (WSP) through an intermediary — usually a firewall or load balancer. In Federated Access Manager, the following web service standards are implemented:

- Liberty Alliance Project Identity Web Services Framework (Liberty ID-WSF)
- WS-I Basic Security Profile

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 can 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.

Identity Services

Federated Access Manager exposes its back-end identity functions as simple web services that are accessible using SOAP, WSDL and REST, allowing developers to easily invoke them when developing an application using one of the supported integrated development environments (IDE). This access management solution does not require deployment of an agent or a proxy. Capabilities available include:

- Authentication
- Authorization
- Collection of the profiles of authenticated users
- Ability to audit and record operations

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.

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)

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 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. **Embedded Configuration Data Store:** Federated Access Manager embeds the open-source OpenDS directory server as a configuration data store. You can also point to instances of Sun Java System Directory Server 5.1, 5.2 & 6.2 during configuration of the product for use as a configuration data store. See “Data and Data Stores” on page 37 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, Tivoli Directory Server 6.1, and Microsoft Active Directory 2003. See “Data and Data Stores” on page 37 for more information.

Note – The open-source OpenDS data store embedded with Federated Access Manager as a configuration data store should only be used as a user data store for proof of concepts and developers.

---

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:** Every Federated Access Manager service exposes a service provider interface (SPI) that allows 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

Dissecting 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" on page 15
- "How Federated Access Manager Works" on page 17
- "Core Services" on page 19
- "Infrastructure" on page 36

Federated Access Manager Architecture

Federated Access Manager is written entirely in Java, and leverages 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.

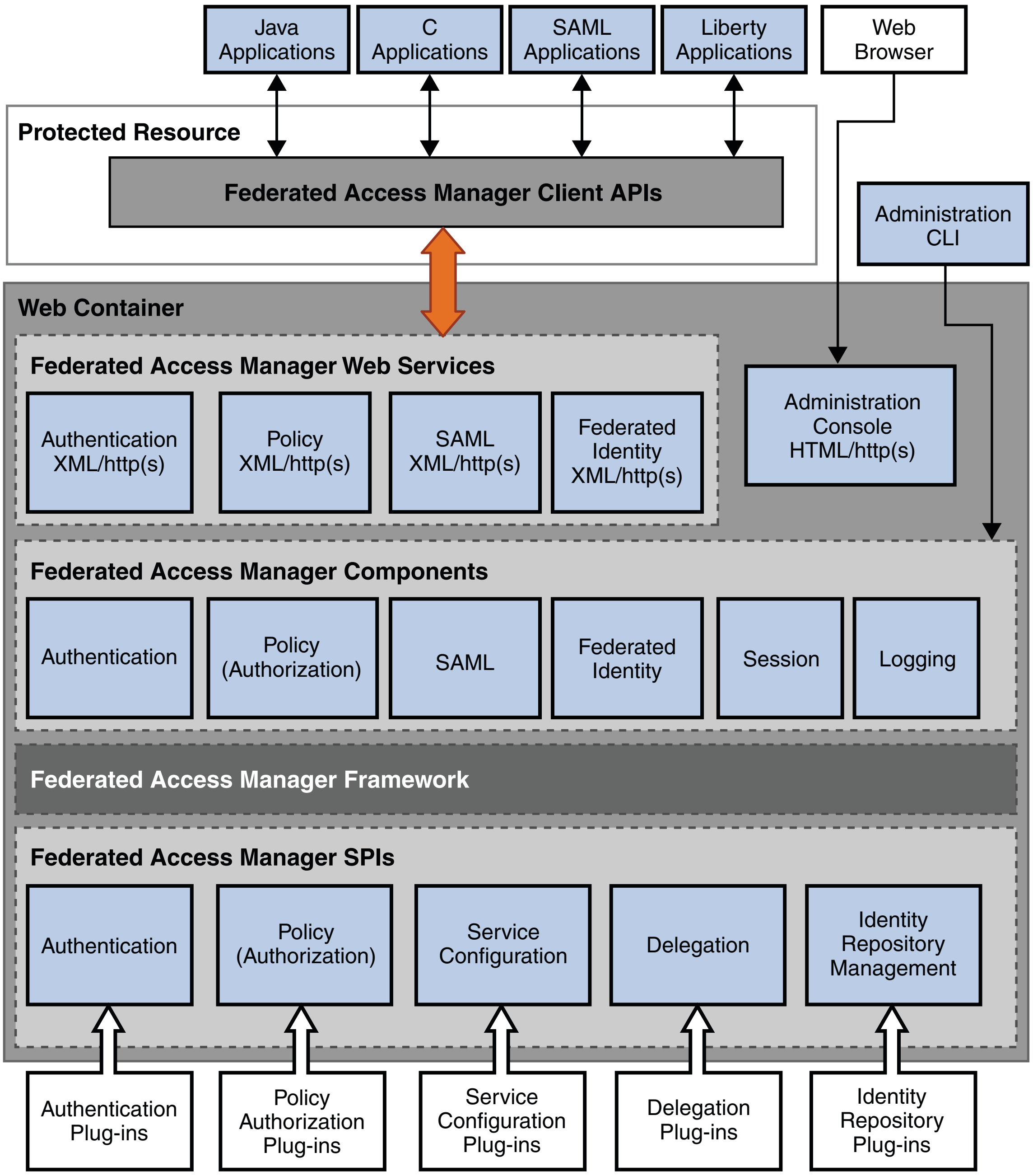

Java Applications
C Applications
SAML Applications
Liberty Applications
Web Browser
Protected Resource
Federated Access Manager Client APIs
Administration CLI
Web Container
Federated Access Manager Web Services
Authentication XML/http(s)
Policy XML/http(s)
SAML XML/http(s)
Federated Identity XML/http(s)
Administration Console HTML/http(s)
Federated Access Manager Components
Authentication
Policy (Authorization)
SAML
Federated Identity
Session
Logging
Federated Access Manager Framework
Federated Access Manager SPIs
Authentication
Policy (Authorization)
Service Configuration
Delegation
Identity Repository Management
Authentication Plug-ins
Policy Authorization Plug-ins
Service Configuration Plug-ins
Delegation Plug-ins
Identity Repository Plug-ins

FIGURE 2–1 Internal Architecture of Federated Access Manager

The Federated Access Manager framework is where business logic is implemented. Each core component uses its own framework to retrieve customer data from the plug-in layer and to provide data to other core components. The Federated Access Manager framework integrates all of the business logic into one layer that is accessible to all core components and plug-ins.

The Client SDK is installed on a machine remote to the Federated Access Manager server that holds the resource to be protected. 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

When Federated Access Manager starts up, it is initialized with configuration data from various service plug-ins including those for the Policy Service, the Identity Repository Service, as well as the embedded service configuration data store. When someone (using a browser) sends an HTTP request for access to a protected resource, Federated Access Manager immediately binds to the appropriate Identity Repository to obtain user information (which may include definitions for roles, realms, user IDs, and so forth). At the same time, a policy agent (separately downloaded and installed on the protected resource) intercepts the request and examines it. If no session token is found, the policy agent contacts Federated Access Manager which will 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.

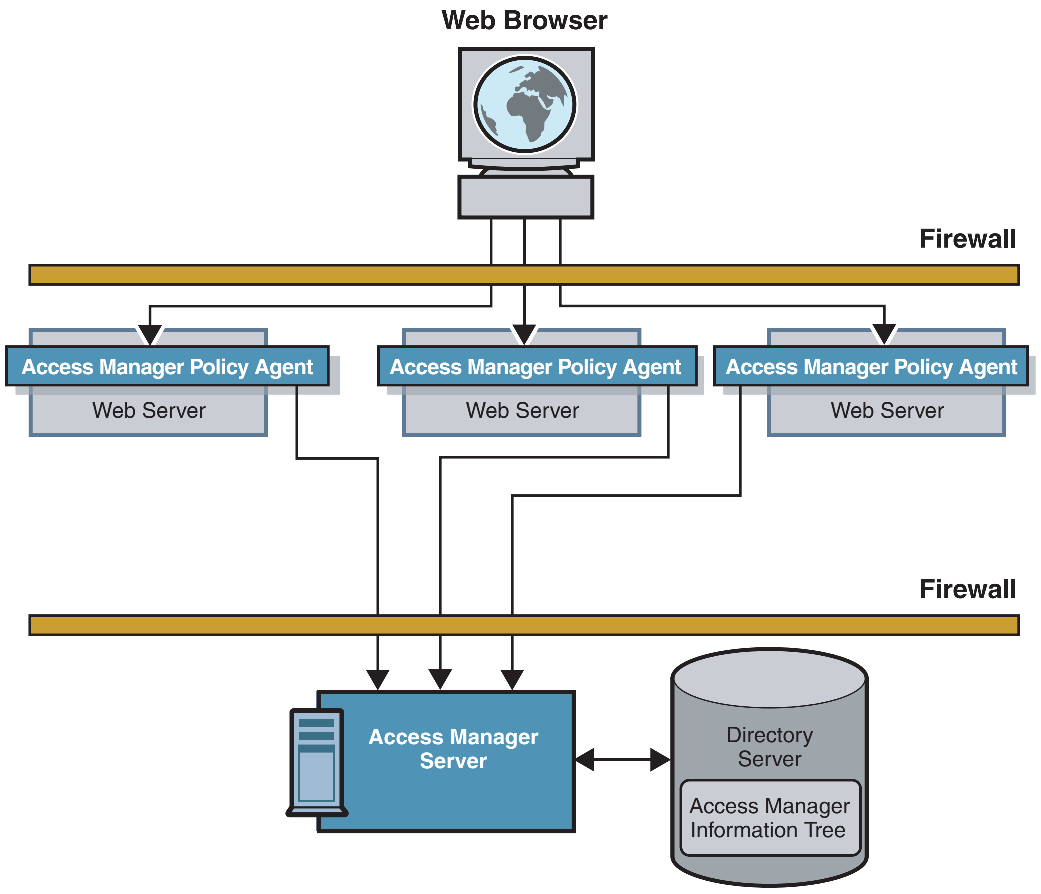

Web Browser
Firewall
Access Manager Policy Agent
Web Server
Access Manager Policy Agent
Web Server
Access Manager Policy Agent
Web Server
Firewall
Access Manager Server
Directory Server
Access Manager Information Tree

FIGURE 2–2 Basic Federated Access Manager Deployment

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

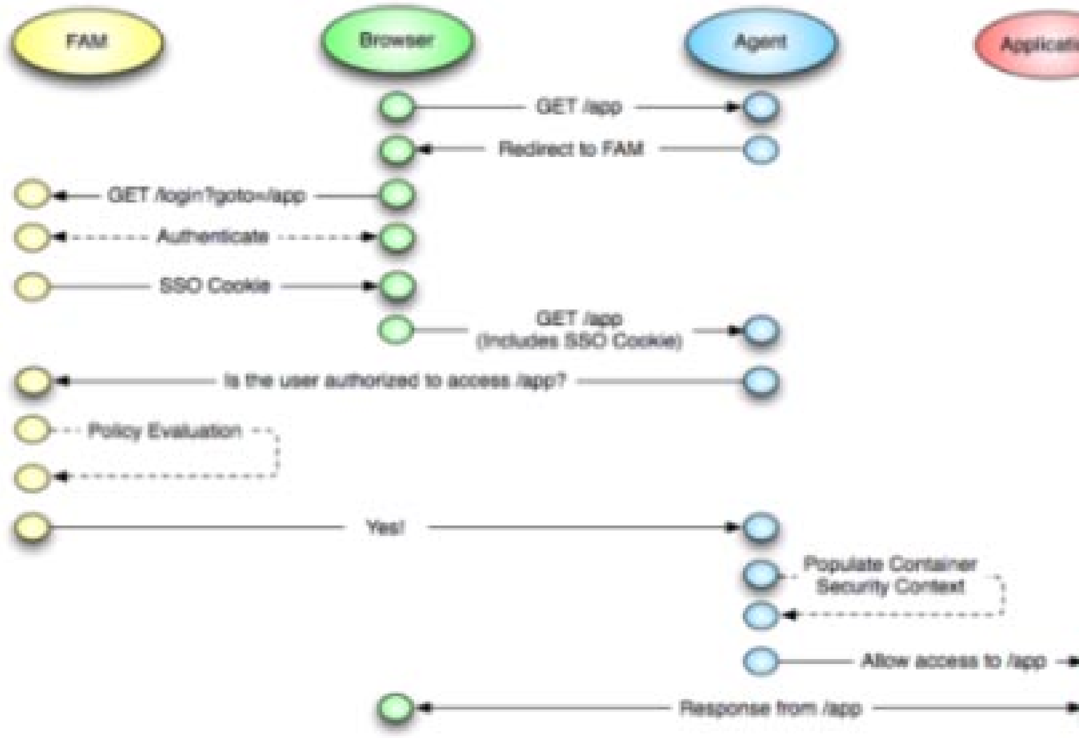


FIGURE 2–3 Federated Access Manager Authentication and Authorization Interactions

For more information on the core functions of Federated Access Manager, see "Core Services" on page 19.

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.

As illustrated in "Federated Access Manager Architecture" on page 15, the Federated Access Manager framework is where business logic is implemented. 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 20
- "Policy Service" on page 22
- "Session Service" on page 25
- "Logging Service" on page 30
- "Identity Repository Service" on page 31
- "Federation Services" on page 32
- "Web Services Security" on page 33

Note – Some services also provide a public SPI that allows the service to be extended. See the *Sun Java System Federated Access Manager 8.0 Developer's Guide*, the *Sun Java System 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. This is typically done to protect the identity repository by deploying an authentication proxy in the DMZ and using the authentication interfaces provided in the Client SDK to pass user credentials back and forth. 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 data 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*. The following figure illustrates how the authentication subcomponents interact within the infrastructure.

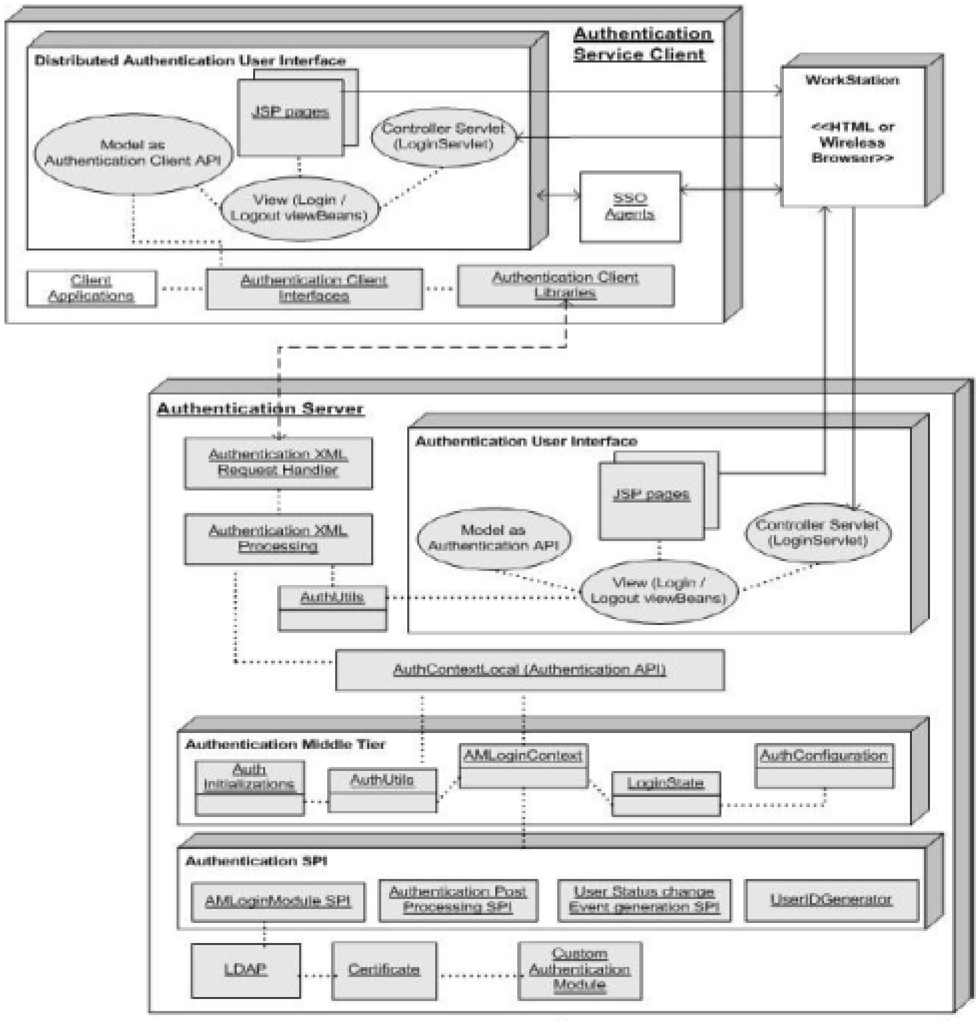

Authentication Service Client
Distributed Authentication User Interface
JSP pages
Model as Authentication Client API
Controller Servlet (LoginServlet)
View (Login / Logout viewBeans)
SSO Agents
WorkStation
<<HTML or Wireless Browser>>
Client Applications
Authentication Client Interfaces
Authentication Client Libraries
Authentication Server
Authentication XML Request Handler
Authentication XML Processing
AuthUtils
Authentication User Interface
JSP pages
Model as Authentication API
Controller Servlet (LoginServlet)
View (Login / Logout viewBeans)
AuthContextLocal (Authentication API)
Authentication Middle Tier
Auth Initializations
AuthUtils
AMLoginContext
LoginState
AuthConfiguration
Authentication SPI
AMLoginModule SPI
Authentication Post Processing SPI
User Status change Event generation SPI
UserIDGenerator
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 is configured using the Administration Console, and comprises the following:

- A *Schema* for the policy type (normal or referral) that describes the syntax of policy. This is defined in `amPolicy.xml`.
- 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.

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

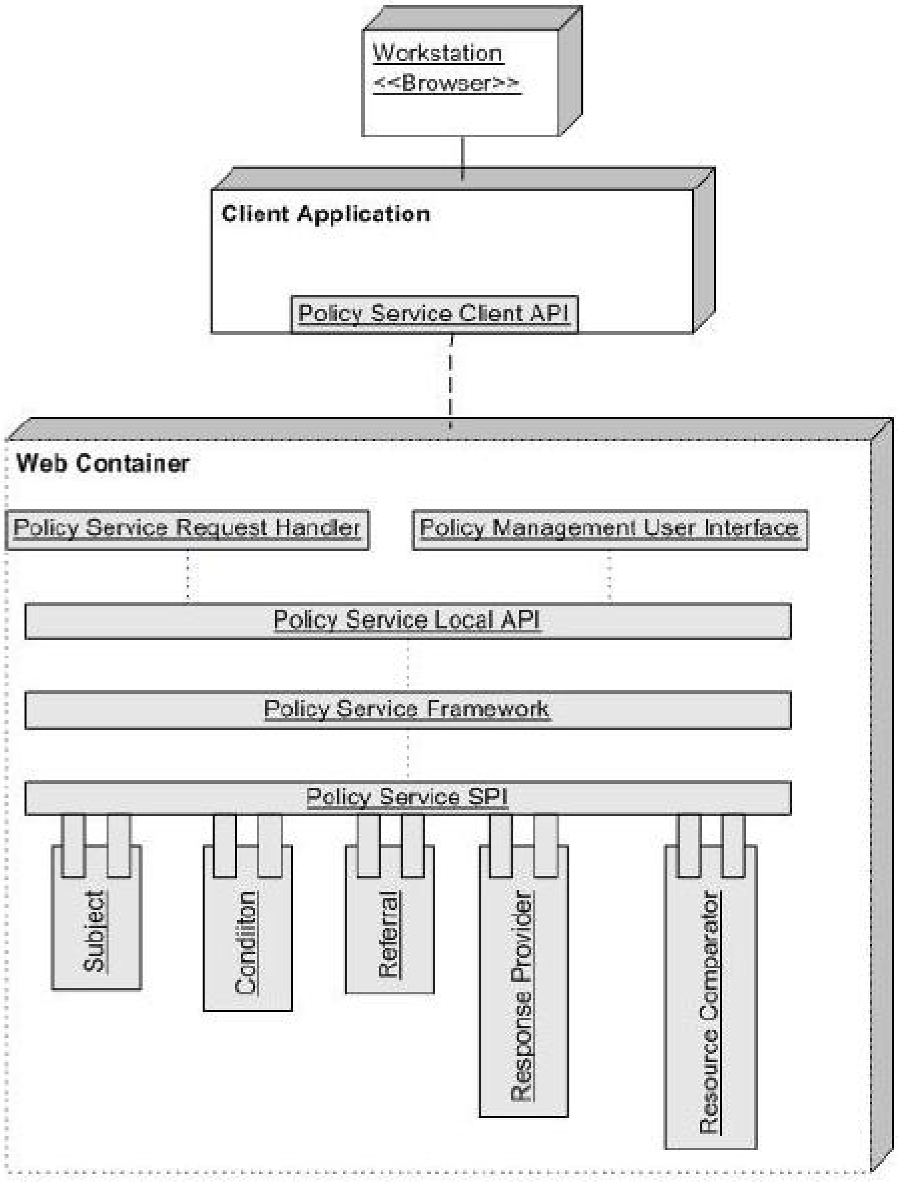


FIGURE 2–5 Policy Components within the Policy Service Framework

---

Note – remoteinterface.dtd defines the XML interfaces that can be used to evaluate policies and to get policy decisions for users.

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 J2EE 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

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

For an overview of the available policy agents and links to specific information on installation, see the *Sun Java System Access Manager Policy Agent 2.2 User's Guide*.

Session Service

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, protected 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 the user is authorized). Successful authentication through Federated Access Manager results in the creation of a session data structure for the user or entity by the 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.

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.

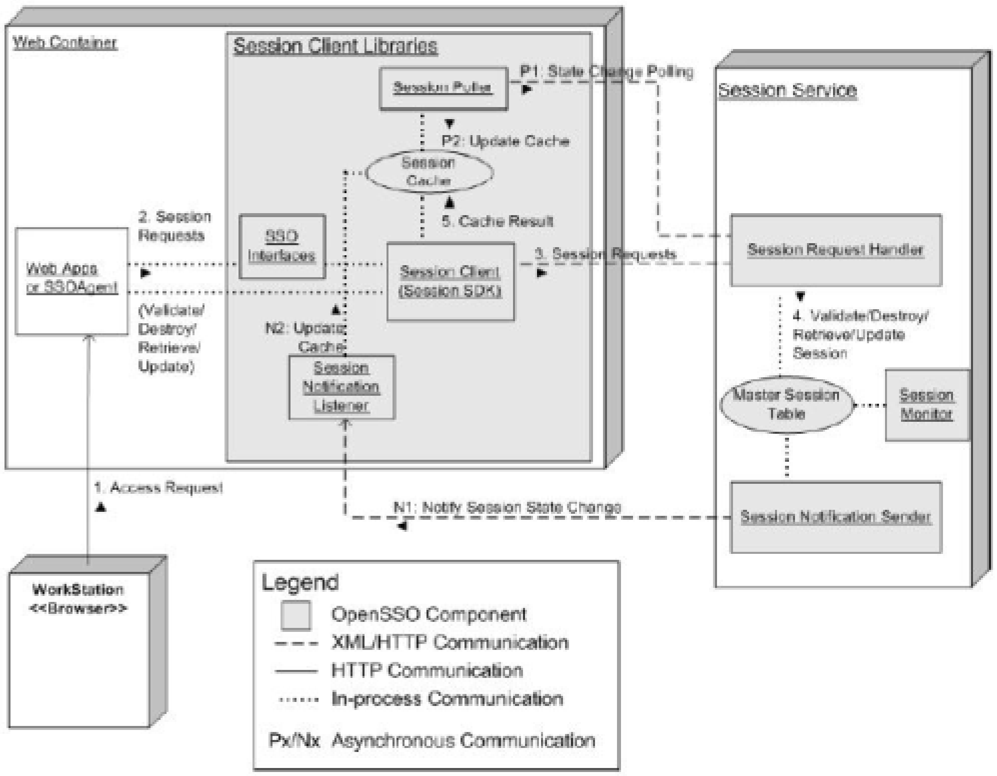


FIGURE 2–6 Components within the Session Service Framework

The following DTD files define the protocol for requests sent to, and responses received from, the Session Service:

- SessionNotification.dtd
- SessionRequest.dtd
- SessionResponse.dtd

Figure 2–7 illustrates how the efficient messaging capabilities of a Message Queue are used to push session information to a persistent store based on the Berkeley DataBase (DB). Two key features are enabled with this:

- 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.

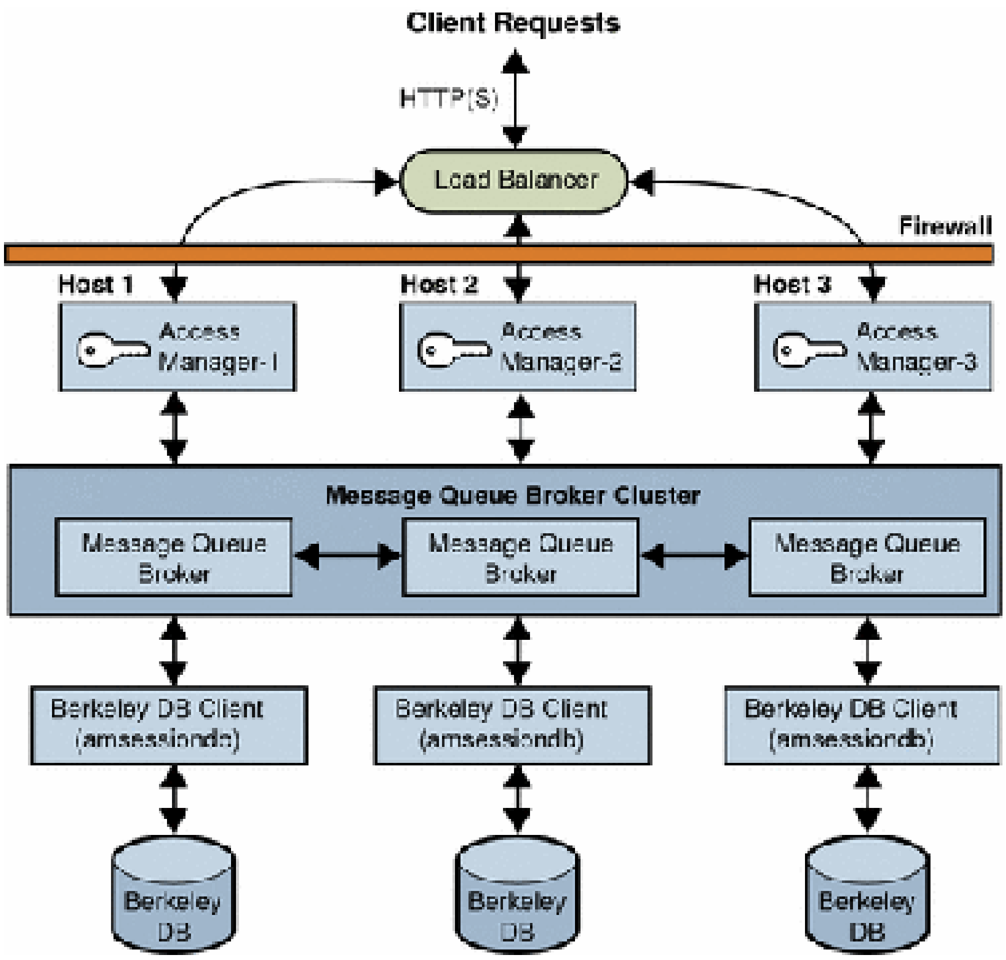


FIGURE 2–7 Session Persistence

The following sections have more information regarding sessions and the Session Service.

- "Session Data Structure" on page 28
- "Session Data Structure Identifiers" on page 29

Additional information can be found in Chapter 6, "Authorization and the Policy Service."

Session Data Structure

The session data structure contains session information exchanged by all Federated Access Manager entities regarding proof of authentication. It is a programmatic object that maintains up-to-date information about a session including:

- Principal
- Universal identifier
- Time user authenticated
- Authentication modules list (list of modules to which the user has authenticated)
- Authentication level
- Time of last access (idle time out processing)
- Maximum time (hard limit time at which the session is destroyed)

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

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

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

- Fixed session attributes
- Protected properties
- Custom properties

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

Session Data Structure Identifiers

The result of a successful authentication results in the creation of a session data structure for the user or entity and a session token. 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. If the session token is known to a protected resource such as an application, the application can access all user and session information identified by 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.

Note – Cookies are domain-specific. For example, a cookie generated by a web server within Domain A cannot be used by a web server in Domain B. Cookies can be passed only between servers in the same domain in which the cookie was set. Similarly, servers can set cookies only on servers within in their own domain. The fact that a web server generates a cookie for a user does not guarantee that the user is allowed access to protected resources. The cookie simply points to user information in a data store from which an access decision can be derived.

The web-based authentication of a user to Federated Access Manager results in the creation of an `SSOToken` stored as a cookie in the user's browser. As the user visits different protected resources using the browser, the `SSOToken` is propagated to these resources. The resource uses the `SSOToken` to retrieve the user's credentials and validate them by sending a back-end request (via an embedded client API or agent) to Federated Access Manager. The server then returns an error or the session's prior authentication information (including the authentication level, authentication module, time of authentication, and idle time). 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. Sessions (and hence the `SSOToken`) are invalidated when a user logs out, the session expires, or a user in an administrative role invalidates it.

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.

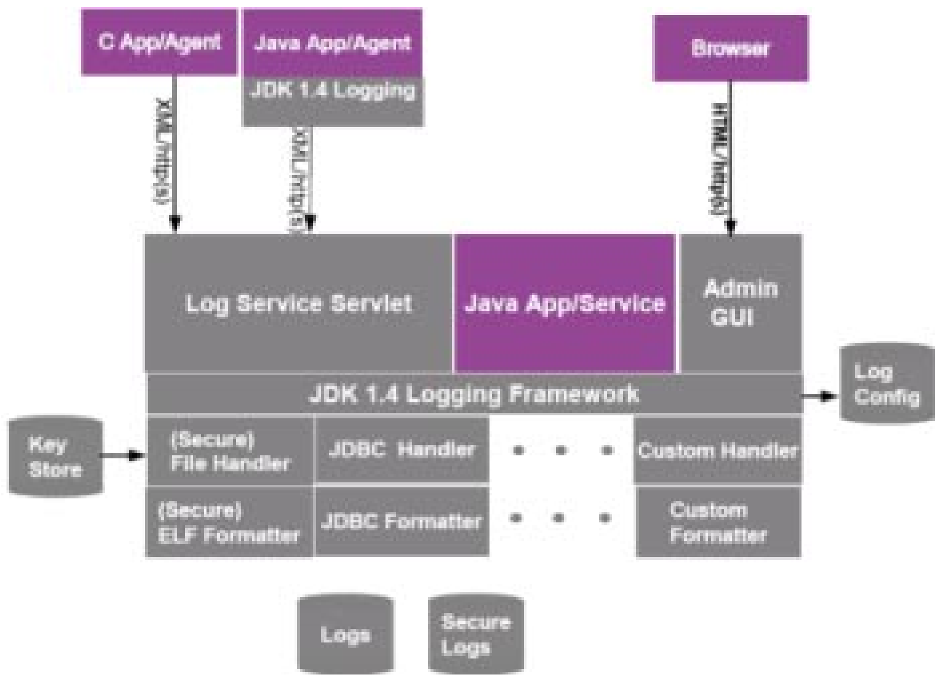


FIGURE 2–8 Components within the Logging Service Framework

See Chapter 10, "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, and is capable of spanning multiple repositories — even of different types. The current implementation supports any LDAPv3 compliant repository and is certified for Sun Java System Directory Server and 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 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 – Access control for the Identity Repository Service is controlled by the Delegation Service using the Policy Service framework. The Delegation Service defines permissions that can be assigned to groups or roles. Those configured permissions become realm and policy privileges, and can be further assigned to administrator roles.

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 is built on the following standards:

- Liberty Alliance Project Identity Federation Framework (Liberty ID-FF) 1.1 and 1.2
- Liberty Alliance Project Identity Web Services Framework (Liberty ID-WSF) 1.0, 1.1, and 2.0
- OASIS Security Assertion Markup Language (SAML) 1.0 and 1.1
- OASIS Security Assertion Markup Language (SAML) 2.0
- WS-Federation (Passive Requestor Profile)
- Web Services-Interoperability (WS-I) Basic Security 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.

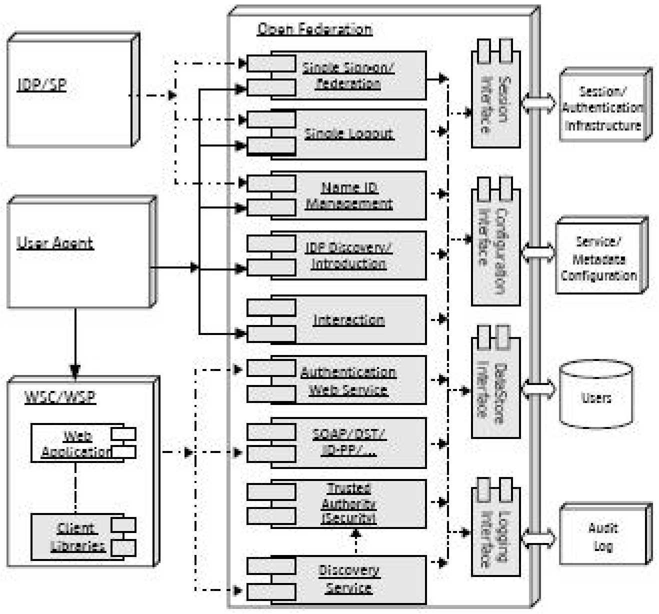


FIGURE 2–9 Federation Components Interaction within the Federated Access Manager Infrastructure

See the Federation Use Case documentation for more information.

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.

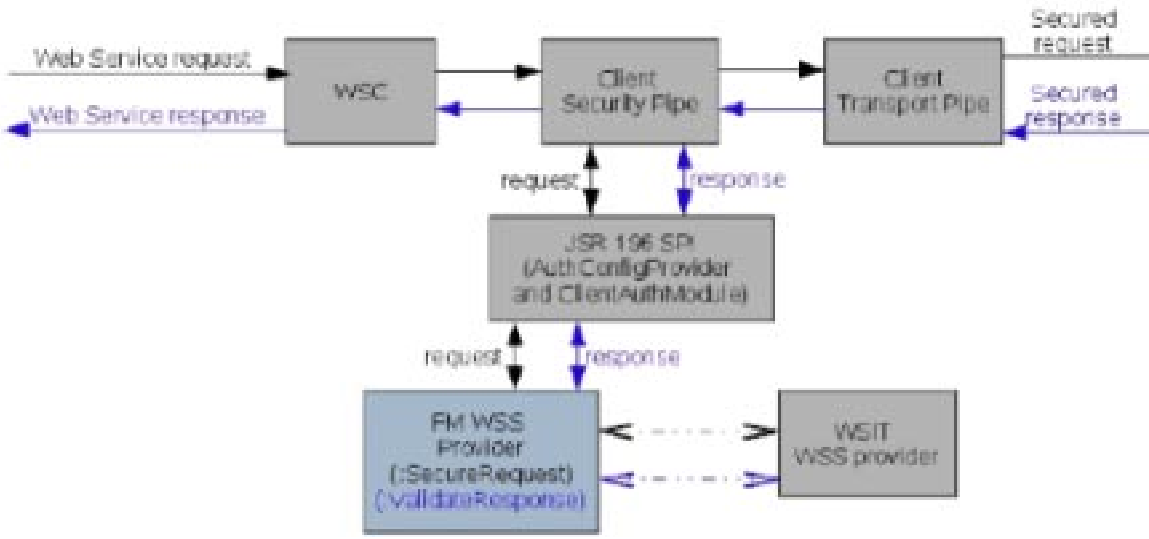


FIGURE 2–10 Components within the Web Services Security Interactions

---

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.

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 the following security-related capabilities:

- Authentication for verification of user credentials.
- Authorization for authorized access of secured resources.
- Attributes for collecting the profile attributes of authenticated users.
- Logging for auditing and recording operations.

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.

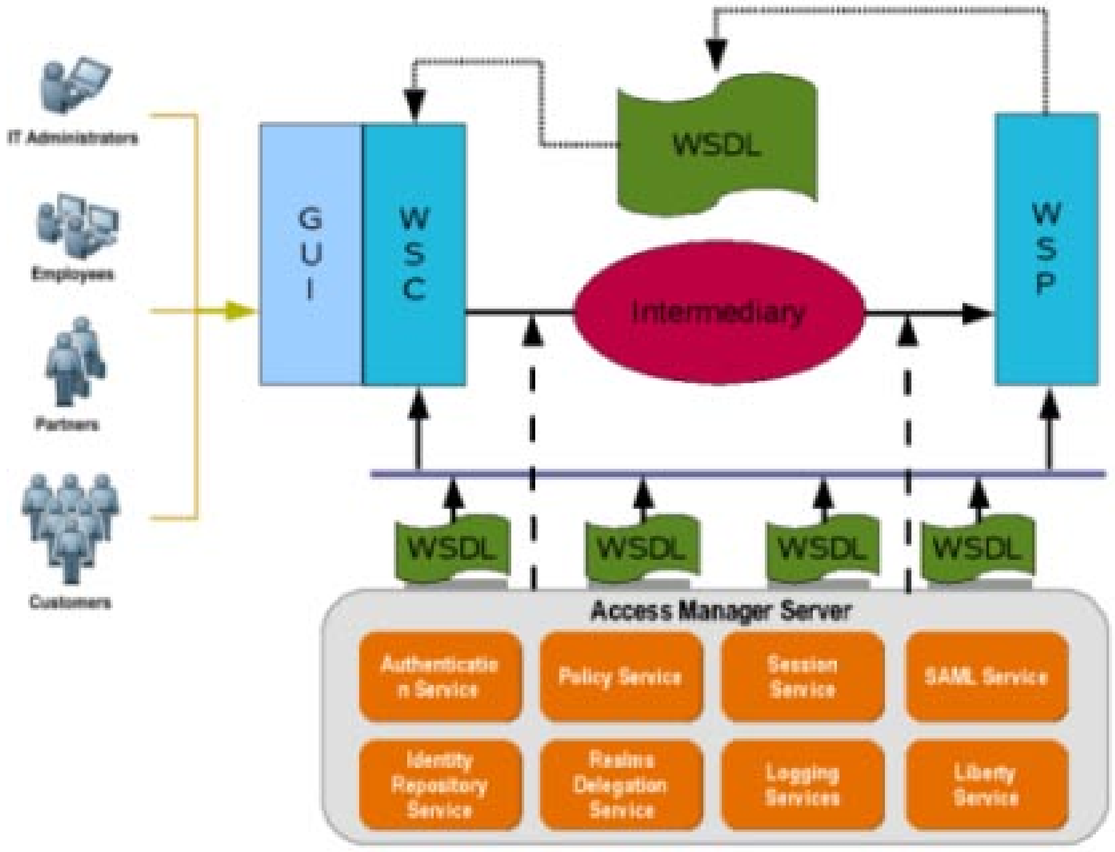


FIGURE 2–11 Components within the Identity Services Interactions

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

Infrastructure

The following sections provide information on the components that define the infrastructure of a deployed Federated Access Manager server.

- "Data and Data Stores" on page 37
- "Realms" on page 43
- "The `bootstrap` File" on page 45
- "Web Services" on page 46

- "Global Services" on page 47
- "Policy Agents" on page 48
- "Authentication Agents" on page 48
- "Client SDK" on page 48
- "Service Provider Interfaces and Plug-ins" on page 49

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.

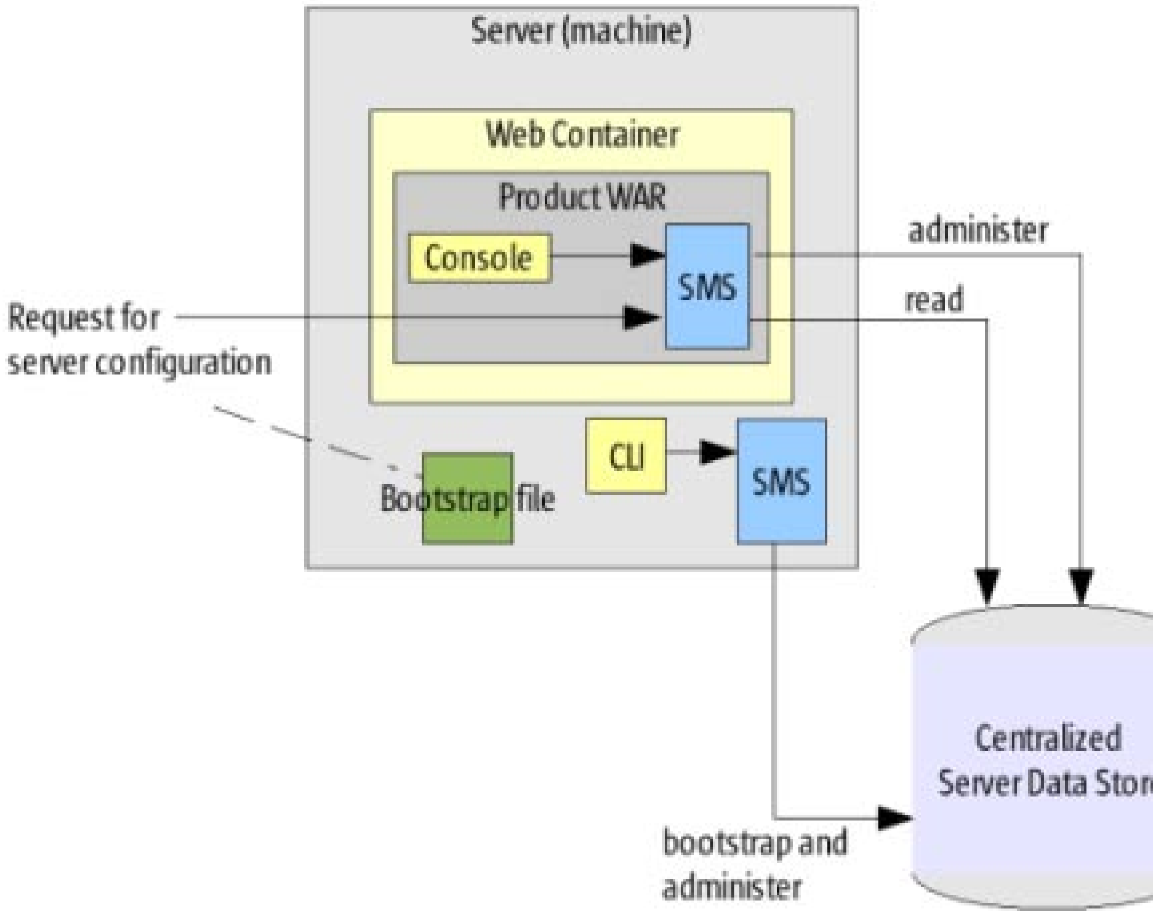
- "Configuration Data" on page 37
- "Identity Data" on page 40
- "Authentication Data" on page 42

Configuration Data

The default configuration of Federated Access Manager creates a branch in an embedded 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.

The following image illustrates how configuration data in this embedded data store is accessed.



Note – OpenDS is the embedded open source directory service installed by default. Clients can retrieve and update the configuration data using standard network protocols. LDAP and Directory Services Markup Language (DSML) are currently supported.

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 Java System 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,dc=opensso,dc=java,dc=net` 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.

- `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.

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, or 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. Any LDAPv3 compliant repository is supported although Federated Access Manager is certified for Sun Java System Directory Server and Microsoft Active Directory. Plug-ins can be developed to integrate other types of repositories (for example, a relational database). The following figure illustrates a Federated Access Manager deployment where the identity data and the configuration data are kept in separate data stores.

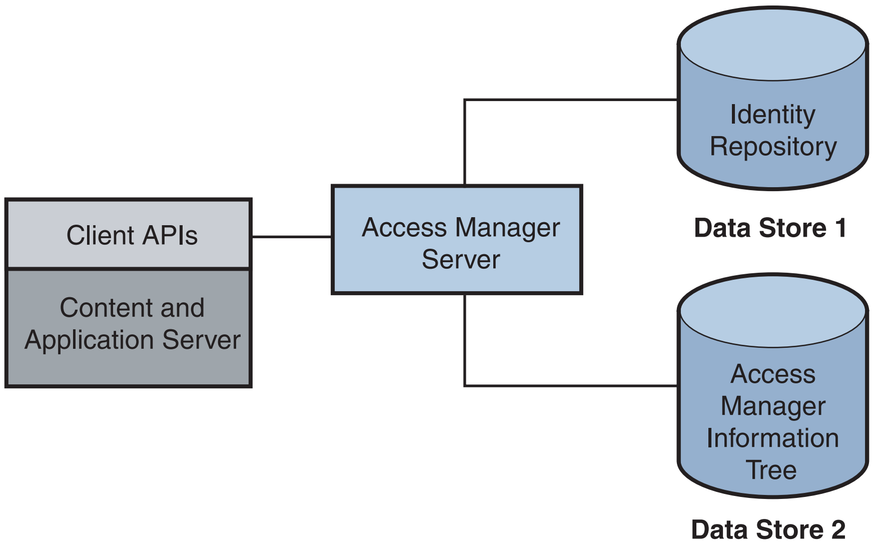


FIGURE 2–12 Federated Access Manager Deployment with Two Data Stores

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 different types of 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 will instantiate it and execute the operations on the identity repository it supports. Each new plug-in developed must have a corresponding service management schema defining its configuration attributes. This schema would be integrated into `idRepoService.xml` (the service management file for the Identity Repository Service which controls the identity data stores under a realm's Data Stores tab) as a sub schema. The following sections contain information on the out-of-the-box plug-ins.

- "Access Manager Repository Plug-in" on page 41
- "Active Directory" on page 42
- "Generic Lightweight Directory Access Protocol (LDAP) version 3" on page 42
- "Sun Directory Server With Access Manager Schema" on page 42

Note – For more information see the "Identity Repository Service" on page 31.

Access Manager Repository Plug-in

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

Note – Previously, the functionality of this plug-in was provided by the AMSDK component. In Access Manager 7.1, the AMSDK functionality still exists, but as a plug-in only. (See "AM SDK Plug-in" on page 50.) Thus, the Access Manager Repository is compatible with previous versions of Access Manager.

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

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

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

Active Directory

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

Generic Lightweight Directory Access Protocol (LDAP) version 3

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

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

Sun Directory Server With Access Manager Schema

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

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.

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. The following figure 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.

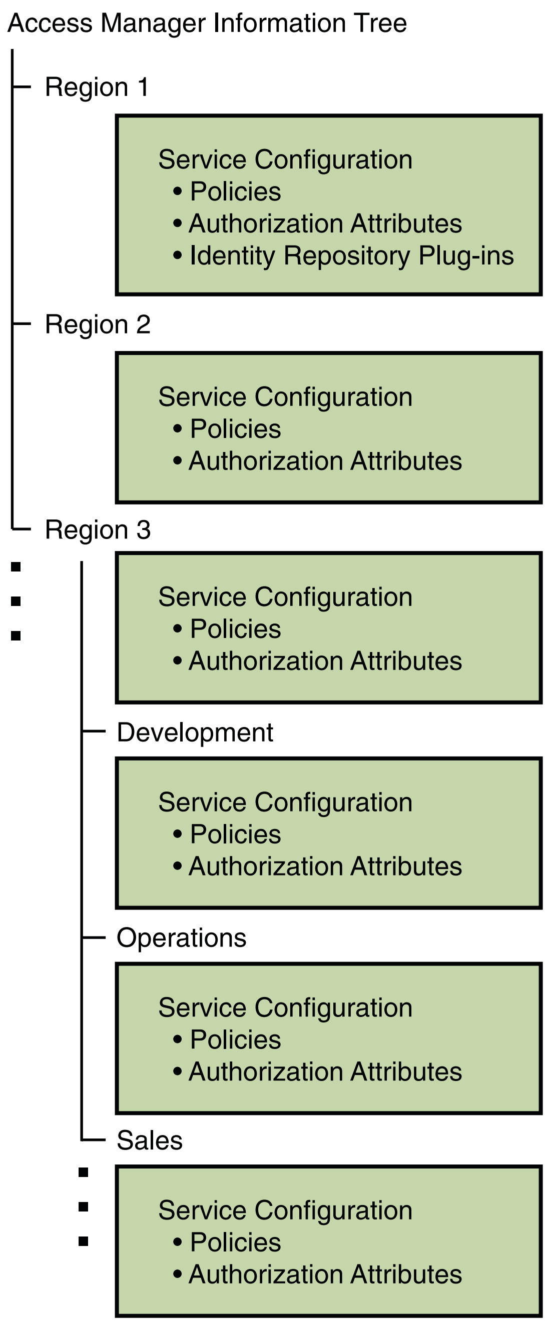


FIGURE 2–13 Realm Hierarchy for Configuration Data

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

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.
- A 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*`&b`

For example:

```
ldap://ds.samples.com:389/http%3A%2F%2Fowen2.red.sun.com%3A8080%2Fopensso?
pwd=AQIC5wM2LY4Sfcxi1dVZEdtfwar2vhWNkmS8&embeddedds=%2Fopensso%2Fopends&
basedn=dc%3Dopensso%2Cdc%3Djava%2Cdc%3Dnet&dsmgr=cn%3DDirectory+Manager
&dspwd=AQIC5wM2LY4Sfcxi1dVZEdtfwar2vhWNkmS8
```

where

- `ds.samples.com:389` is the host name and port of the machine on which the directory is installed.
- `http%3A%2F%2Fowen2.red.sun.com%3A8080%2Fopensso` is the instance name.
- `AQIC5wM2LY4Sfcxi1dVZEdtfwar2vhWNkmS8` is the encrypted password of the OpenSSO administrator.
- `%2Fopensso%2Fopends` is the path to the directory installation.
- `dc%3Dopensso%2Cdc%3Djava%2Cdc%3Dnet` is the base DN.
- `cn%3DDirectory+Manager` is the directory administrator.
- `BQIC5xM2LY4SfcximdVZEdtfwar4vhWNkmG7` is the encrypted password for the directory administrator.

Federated Access Manager supports Microsoft's 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 — using `bootstrap`. 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%3A%2F%2Fowen2.red.sun.com%3A8080%2Fopensso?
pwd=AQIC5wM2LY4Sfcxi1dVZEdtfwar2vhWNkmS8&embeddedds=%2Fopensso%2Fopends&
basedn=dc%3Dopensso%2Cdc%3Djava%2Cdc%3Dnet&dsmgr=cn%3DDirectory+Manager
&dspwd=AQIC5wM2LY4Sfcxi1dVZEdtfwar2vhWNkmS8
```

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 use XML and SOAP over HTTP. These web services are designed to be centrally provided in an enterprise's network for convenient access by client applications. The following table summarizes the Federated Access Manager web services.

Additionally, the following web service standards are implemented:

- ID-WSF 1.1
- WS-Security (WS-I BSP)

TABLE 2–1 Federated Access Manager Web Services

Web Service Name	Description
Authentication Web Service	Verifies that a user really is the person he claims to be.
Policy (Authorization)	Evaluates rules (policies) associated with a user’s identity, and determines whether an authenticated user has permission to access a protected resource.
SAML	Enables single sign-on sessions among different business domains. Allows business partners to securely exchange authentication and authorization information over the Internet.

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

Web Service Name	Description
Federation	Enables a user to log in at one service provider's site and move to an affiliated service provider site without having to re-authenticate or re-establish identity.
Session	Maintains information about the user's interaction with various applications the user accesses.

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.

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	XXXXX
Liberty ID-FF Service Configuration	XXXXX
Liberty ID-WSF Security Service	XXXXX
Liberty Interaction Service	XXXXX
Multi-Federation Protocol	XXXXX
Password Reset	XXXXX
Policy Configuration	XXXXX
SAML v2 Service Configuration	XXXXX

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

Service	What it Does
SAML v2 SOAP Binding	XXXXX
Security Token Service	XXXXX
Session	XXXXX
User	XXXXX

Policy Agents

Policy agents enforce policy decisions determined by the Policy Service. You install a policy agent on the machine that hosts a resource you'd like to protect. The policy agent intercepts requests from applications, and redirects the requests to Federated Access Manager for authentication. Once the user is authenticated, the policy agent communicates with the Policy Service for authorization. The policy agent allows or denies the user access depending upon the result of policy evaluation. Policy agents are downloaded and installed separately from Federated Access Manager. For more information, see *Sun Java System Access Manager Policy Agent 2.2 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 33.

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.

Service Provider Interfaces and Plug-ins

The Federated Access Manager service provider interfaces (SPI) work with 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*. The following sections contain brief descriptions.

- "Authentication Plug-in" on page 49
- "Delegation Service Plug-in" on page 49
- "Identity Repository Service Plug-in" on page 50
- "Policy Plug-in" on page 50
- "Service Configuration Plug-in" on page 50
- "AM SDK Plug-in" on page 50

Authentication Plug-in

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

Delegation Service Plug-in

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

TABLE 2–3 Administrator Roles and Scope of Privileges

Administrator Role	Delegation Rule
Realm Administrator	Can access all data in all configured realms.
Subrealm Administrator	Can access all data within the specified realm.
Policy Administrator	Can access all policies in all configured realms.
Policy Realm Administrator	Can access policies only within the specified realm.

Identity Repository Service Plug-in

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

Policy Plug-in

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

Service Configuration Plug-in

The Service Configuration plug-in stores and manages configuration data required by the core 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).

AM SDK Plug-in

The AM SDK plug-in creates and modifies users and stores information in the user branch of the identity repository.

Note – The AM SDK Plug-in implements the user management API used in previous releases.

Index