

For Review Purposes Only

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
May 2008

Copyright 2008 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. 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 U.S. 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.

This distribution may include materials developed by third parties.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, the Solaris logo, the Java Coffee Cup logo, docs.sun.com, Java, Java, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and SunTM Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

Products covered by and information contained in this publication are controlled by U.S. Export Control laws and may be subject to the export or import laws in other countries. Nuclear, missile, chemical or biological weapons or nuclear maritime end uses or end users, whether direct or indirect, are strictly prohibited. Export or reexport to countries subject to U.S. embargo or to entities identified on U.S. export exclusion lists, including, but not limited to, the denied persons and specially designated nationals lists is strictly prohibited.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 2008 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. Tous droits réservés.

Sun Microsystems, Inc. détient les droits de propriété intellectuelle 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 plusieurs brevets américains ou des applications de brevet en attente aux Etats-Unis et dans d'autres pays.

Cette distribution peut comprendre des composants développés par des tierces personnes.

Certains composants de ce produit peuvent être dérivées du logiciel Berkeley BSD, licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays; elle est licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, le logo Solaris, le logo Java Coffee Cup, docs.sun.com, Java, Java et Solaris sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

L'interface d'utilisation graphique OPEN LOOK et Sun a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui, en outre, se conforment aux licences écrites de Sun.

Les produits qui font l'objet de cette publication et les informations qu'il contient sont régis par la législation américaine en matière de contrôle des exportations et peuvent être soumis au droit d'autres pays dans le domaine des exportations et importations. Les utilisations finales, ou utilisateurs finaux, pour des armes nucléaires, des missiles, des armes chimiques ou biologiques ou pour le nucléaire maritime, directement ou indirectement, sont strictement interdites. Les exportations ou réexportations vers des pays sous embargo des Etats-Unis, ou vers des entités figurant sur les listes d'exclusion d'exportation américaines, y compris, mais de manière non exclusive, la liste de personnes qui font objet d'un ordre de ne pas participer, d'une façon directe ou indirecte, aux exportations des produits ou des services qui sont régis par la législation américaine en matière de contrôle des exportations et la liste de ressortissants spécifiquement désignés, sont rigoureusement interdites.

LA DOCUMENTATION EST FOURNIE "EN L'ETAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFACON.

List of Remarks

Contents

- 5 Implementing Federation7**
 - Federating Identities7
 - The Concept of Identity8
 - The Concept of Federation8
 - The Concept of Trust9
 - How Federation Works 10
 - Choosing a Federation Option 13
 - Using SAML 13
 - About SAML v2 15
 - About SAML v1.x 19
 - Using the Liberty ID-FF 22
 - About the Liberty ID-FF Process 23
 - Liberty ID-FF Features 26
 - Using WS-Federation 31
 - Creating a Common Domain for Identity Provider Discovery 33
 - The Common Domain 33
 - The Common Domain Cookie 34
 - The Writer Service and the Reader Service 34
- Index35**

Implementing Federation

Sun Java™ System Federated Access Manager provides a framework for implementing a federated identity infrastructure, enabling single sign-on, provisioning users dynamically, and sharing identity attributes across security domains. Forming trust relationships across security domains allows an organization to integrate applications offered by different departments or divisions within the enterprise as well as engage in relationships with cooperating business partners that offer complementary services. Towards this end, multiple industry standards, such as those developed by the Organization for the Advancement of Structured Information Standards (OASIS) and the Liberty Alliance Project, are supported. This chapter contains an overview of the federation framework and federation options.

- [“Federating Identities” on page 7](#)
- [“How Federation Works” on page 10](#)
- [“Choosing a Federation Option” on page 13](#)
- [“Using SAML” on page 13](#)
- [“Using the Liberty ID-FF” on page 22](#)
- [“Using WS-Federation” on page 31](#)
- [“Creating a Common Domain for Identity Provider Discovery” on page 33](#)

Federating Identities

Identity federation allows users to link the many personal online identities they have created with multiple service providers. With a *federated identity*, the individual can log in at one service provider site and move to an affiliated service provider site without having to re-establish identity. The following sections contain information about the underlying concepts regarding a framework for federating identities.

- [“The Concept of Identity” on page 8](#)
- [“The Concept of Federation” on page 8](#)
- [“The Concept of Trust” on page 9](#)

The Concept of Identity

In one dictionary, *identity* is defined as "a set of information by which one person is definitively distinguished." This information undoubtedly begins with the document that corroborates a person's name: a birth certificate. Over time, additional information further defines different aspects of an individual's identity:

- An address
- A telephone number
- One or more diplomas
- A driver's license
- A passport
- Financial institution accounts
- Medical records
- Insurance statements
- Employment records
- Magazine subscriptions
- Utility bills

Each of these represents data that designates a piece of a person's identity as it relates to the enterprise for which the data was defined. The composite of this data constitutes an overall identity with each specific piece providing a distinguishing characteristic.

Because the Internet is one of the primary vehicles for the types of interactions represented by this identity-defining information, people are now creating online identities specific to the businesses with which they are interacting. By creating a user account with an identifier and password, an email address, personal preferences (such as style of music, or opt-in/opt-out marketing decisions) and other information specific to the particular business (a bank account number or ship-to address), a user is able to distinguish their account from others who also use the enterprise's services. This distinguishing information is referred to as a *local identity* because it is specific to the service provider for which it has been defined.

Considering the number of service providers for which a user can define a local identity, accessing each one can be a time-consuming and frustrating experiencing. In addition, although most local identities are configured independently (and fragmented across the Internet), it might be useful to connect the information. For example, a user's local identity with a bank could be securely connected to the same user's local identity with a utility company for easy, online payments. Federation addresses this idea of linking disparate service provider user accounts.

The Concept of Federation

Federation establishes a standards-based method of sharing and managing identity data and establishing single sign-on across security domains and organizations. As a concept, federation encompasses both *identity federation* and *provider federation*.

- [“Identity Federation” on page 9](#)
- [“Provider Federation” on page 9](#)

Identity Federation

Identity federation, as it has evolved with regard to online activity, begins with the notion of local identity. (See [“The Concept of Identity” on page 8.](#)) Sending and receiving email, checking bank balances, finalizing travel arrangements, accessing utility accounts, and shopping are just a few online services for which a user might define a local identity. If a user accesses all of these services, many different local identities have been configured. This virtual phenomenon offers an opportunity to fashion a system for users to *federate* these local identities.

Identity federation allows the user to link, connect, or bind the local identities that have been created for each *service provider* (a networked entity that provides one or more services to other entities). The linked local identities, referred to as a *federated identity*, allow the user to log in to one service provider site and click through to an affiliated service provider without having to reauthenticate or reestablish identity; in effect, single sign-on.

Provider Federation

Provider federation begins with a circle of trust. A *circle of trust* is a group of service providers who contractually agree to exchange authentication information. Each circle of trust must include at least one *identity provider*, a service provider that maintains and manages identity data, and provides authentication services. After the contracts and policies defining a circle of trust are in place, the specific protocols, profiles, endpoints, and security mechanisms being used in the deployment are collected into a metadata document that is exchanged among the members of the circle. Federated Access Manager provides the tools necessary to integrate the metadata and enable a circle of trust technologically. Authentication within this federation is honored by all membered providers.

Note – The establishment of contractual trust agreements between providers is beyond the scope of this guide. See [“The Concept of Trust” on page 9](#) for an overview.

The Concept of Trust

Federating identities assumes existing trust relationships between participants. This trust is usually defined through business arrangements or contracts that describe the technical, operational, and legal responsibilities of each party and the consequences for not completing them. When defined, a trust relationship allows one organization to trust the user authentication and authorization decisions of another organization. This trust then enables a user to log in to one site and, if desired, access a trusted site without reauthentication.

Ensure that trust agreements are in force before configuring circles of trust with Federated Access Manager and going live. The Liberty Alliance Project has created a support document for helping to establish these trust arrangements. The *Liberty Trust Model Guidelines* document is located on the [Support Documents and Utility Schema Files](#) page of the Liberty Alliance Project web site.

How Federation Works

The goal of federation is to enable individuals and service providers to easily conduct network transactions across secure domains while protecting identity data. When organizations form a trust agreement, they agree to exchange user authentication information using specific web technologies. The trust agreement would be among multiple service providers that offer web-based services to users and, at least, one *identity provider* (a service provider that maintains and manages identity information). Once metadata is exchanged and the trust is established, single sign-on can be enabled between all the included providers, and users (in some federation protocol) may opt to federate their multiple identities. In Federated Access Manager, the trust agreement is virtually configured as a *circle of trust* using the console or command line interface. A circle of trust contains *entity providers* (service providers or identity providers) that are grouped together for the purpose of offering identity federation. *Identity federation* occurs when a user chooses to unite distinct service provider and identity provider accounts while retaining the individual account information with each provider. The user establishes a link that allows the exchange of authentication information between provider accounts. Users can choose to federate any or all identities they might have. After identity federation, when a user successfully authenticates to one of the service providers, access to any of the federated accounts within the circle of trust is allowed *without having to reauthenticate*. The following figure shows the subjects involved in federation.

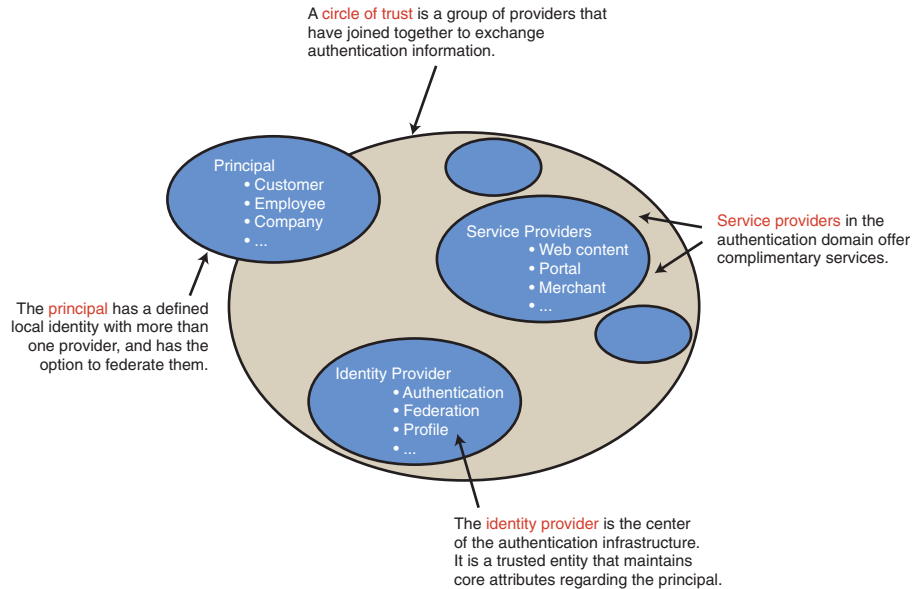


FIGURE 5-1 Subjects Involved in an Identity Federation

- A *principal* can have a defined local identity with more than one provider, and it has the option to federate the local identities. The principal might be an individual user, a group of individuals, a corporation, or a component of the Liberty architecture.
- A *service provider* is a commercial or not-for-profit organization that offers a web-based service such as a news portal, a financial repository, or retail outlet.
- An *identity provider* is a service provider that stores identity profiles and offers incentives to other service providers for the prerogative of federating their user identities. Identity providers might also offer services above and beyond those related to identity profile storage.
- To support identity federation, all service providers and identity providers must join together into a *circle of trust*. A circle of trust must contain at least one identity provider and at least two service providers. (One organization may be both an identity provider and a service provider.) Providers in a circle of trust must first write trust agreements to define their relationships. A *trust agreement* is a contract between organizations that defines how the circle will work. For more information, see “[The Concept of Trust](#)” on page 9.

A travel portal is a good example of a circle of trust. Typically, a travel portal is a web site designed to help you access various travel-related services from one location. The travel portal forms a partnership with each service provider displayed on its web site. (This might include hotels, airlines, and car rental agencies.) The user registers with the travel portal which, in effect, is the identity provider for the circle of trust. After logging in, the user might click through to an airline service provider to look for a flight. After booking a flight, the user might click through

to an accommodations service provider to look for a hotel. Because of the trust agreements previously established, the travel portal shares authentication information with the airline service provider, and the airline service provider with the accommodations service provider. The user moves from the hotel reservations web site to the airline reservations web site without having to reauthenticate. All of this is transparent to the user who must, depending on the underlying federation protocol, choose to federate any or all local identities. The following figure illustrates the travel portal example.

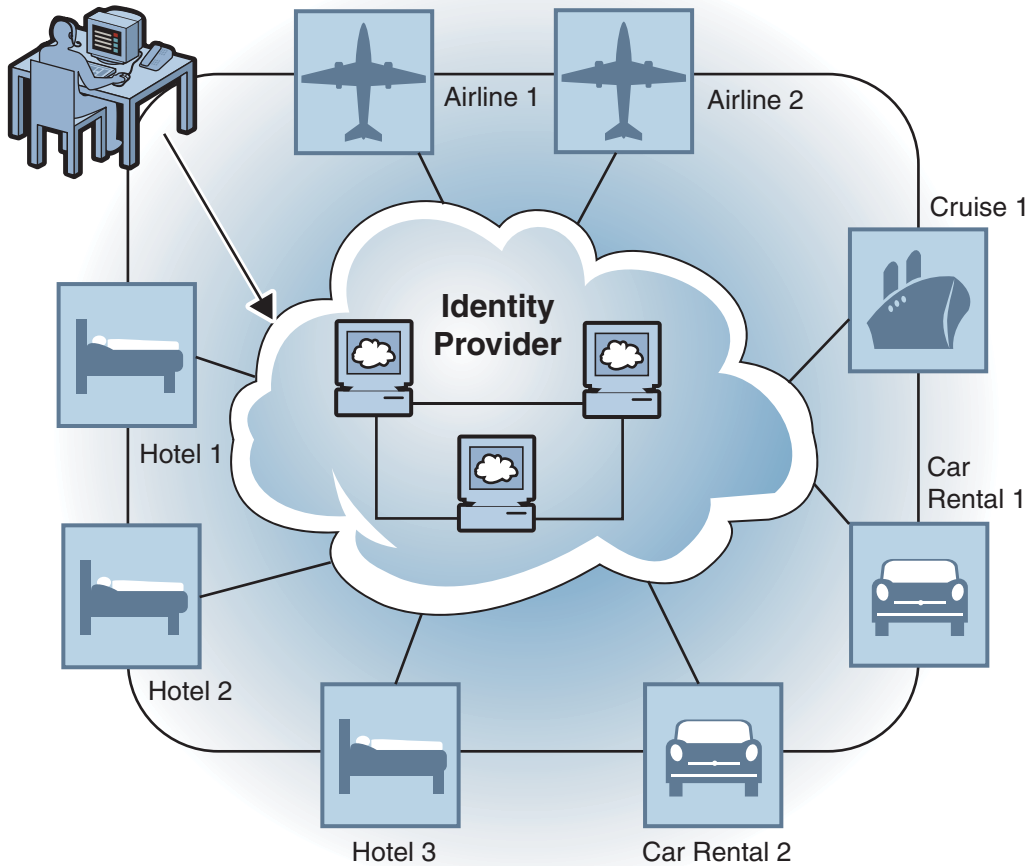


FIGURE 5-2 Federation Within a Travel Portal

Choosing a Federation Option

Federation is used to solve the problem of cooperation across heterogeneous, autonomous environments. In the beginning, federation meant using the Liberty Alliance Project Identity Federation Framework (Liberty ID-FF). Since then, other specifications have been developed with which federation can be accomplished including the Security Assertion Markup Language (SAML) and WS-Federation. To get started, use SAML v2 whenever possible as it supersedes both the Liberty ID-FF and SAML v1.x specifications. If you are integrating Federated Access Manager with Microsoft Active Directory with Federation Services, you **must** use WS-Federation. The Liberty ID-FF and SAML v1.x should only be used when integrating with a partner that is not able to use SAML v2.

Note – Federated Access Manager has appropriated the terms from the Liberty ID-FF for all federation protocol implementations in the Federated Access Manager console.

More information on these options can be found in the following sections:

- [“Using SAML” on page 13](#)
- [“Using the Liberty ID-FF” on page 22](#)
- [“Using WS-Federation” on page 31](#)

Using SAML

SAML defines an XML-based framework for exchanging identity information across security domains for purposes of authentication, authorization and single sign-on. It was designed to be used within other specifications (the Liberty Alliance Project, the Shibboleth project, and the Organization for the Advancement of Structured Information Standards have all adopted aspects of SAML) although the latest release (SAML v2) has incorporated elements from the specifications developed by those very same organizations. The SAML specifications consist of a number of components, illustrated by the following figure.

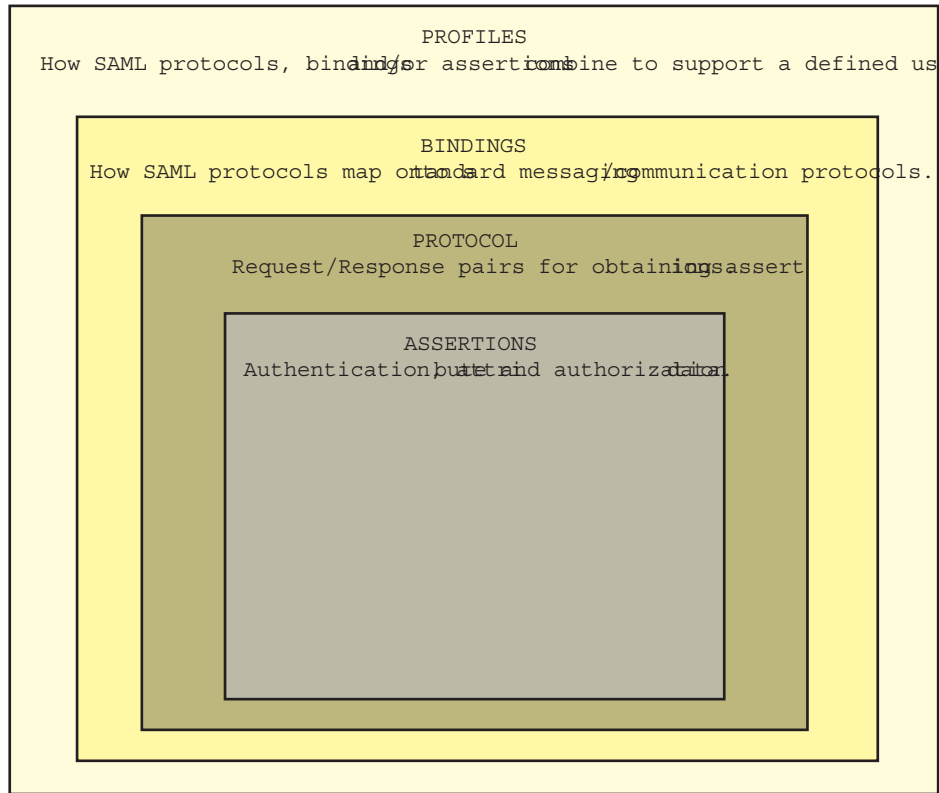


FIGURE 5-3 Components of the SAML Specifications

The SAML specification defines the *assertion* security token format as well as *profiles* that standardize the HTTP exchanges required to transfer XML requests and responses between an asserting authority and a trusted partner. An *assertion* is a package of verified security information that supplies one or more statements concerning a principal's authentication status, access authorization decisions, or identity attributes. (A person identified by an email address is a principal as might be a printer.) Assertions are issued by an asserting authority (a platform or application that declares whether a subject has been authenticated into its system), and received by relying parties (partner sites defined by the authority as *trusted*). Asserting authorities use different sources to configure assertion information, including external data stores or assertions that have already been received and verified.

The most recent SAML v2 specifications are defined more broadly than those developed for SAML v1.x — with particular attention paid to functionality dealing with federation. Before SAML v2 was introduced, SAML v1.x was simply a way to exchange identity data. In fact, up to version 1.1, the Liberty Alliance Project Identity Federation Framework (Liberty ID-FF) was developed using the SAML 1.0 specification. Liberty ID-FF version 1.2 development was continued using the SAML v1.1 specification. But, following the release of version 1.2, the

Liberty ID-FF was incorporated into the SAML v2 specification. Additionally, SAML v2 adds components of the Shibboleth initiative. Thus, SAML v2 is a major revision, providing significant additional functionality and making the previous versions of SAML incompatible with it. Going forward, SAML v2 will be the basis on which Federated Access Manager implements federation. The following diagram illustrates the convergence.

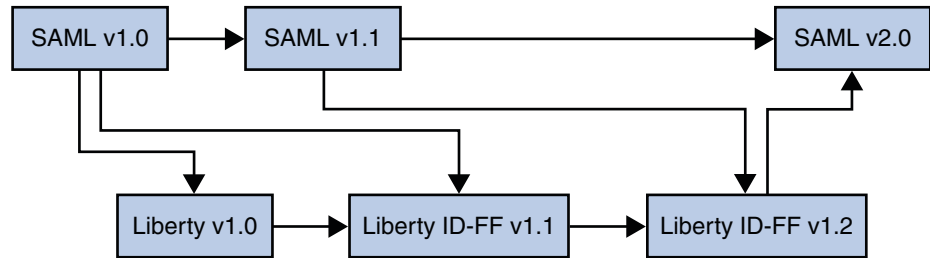


FIGURE 5-4 Liberty ID-FF and SAML Convergence

Note – For more information on this convergence (including how the Shibboleth Project was also integrated), see the [Federation section of Strategic Initiatives](#) on the Liberty Alliance Project web site.

In Federated Access Manager, you can federate identities using the Liberty ID-FF (as discussed in [“Using the Liberty ID-FF” on page 22](#)) or SAML v1.x or SAML v2 (as discussed in the following sections).

- [“About SAML v2” on page 15](#)
- [“About SAML v1.x” on page 19](#)



Caution – SAML v1.x and SAML v2 assertions and protocol messages are incompatible.

About SAML v2

Federated Access Manager delivers a solution that allows businesses to establish a framework for sharing trusted information across a distributed network of partners using the standards-based SAML v2. Towards this end, HTTP(S)-based service endpoints and SOAP service endpoints are supplied as well as assertion and protocol object manipulating classes. A web browser can access all HTTP(S)-based service endpoints and an application can make use of the SOAP endpoints and API as long as metadata for each participating business on BOTH sides of the SAML v2 interaction is exchanged beforehand. The key features of SAML v2 in Federated Access Manager include:

- Single sign-on using the POST profile, the Artifact binding (also referred to as HTTP redirect), and unsolicited responses (initiated by the identity provider)
- Single logout using HTTP redirect and SOAP binding
- Federation termination using HTTP redirect and SOAP binding
- Auto-federation (automatic linking of service provider and identity provider user accounts based on a common attribute)
- Bulk federation
- Dynamic creation of user accounts
- One time federation (transient NameID format for SSO)
- Basic Authentication, SSL and SSL with client authentication for SOAP binding security
- SAML v2 authentication
- Identity provider discovery
- XML verification, signing, encryption and decryption
- Profile initiation and processing using included JavaServer Pages™ (JSP™)
- Load balancing support
- Protocol coexistence with the SAML v1.x and the Liberty ID-FF

The following figure illustrates the SAML v2 framework which consists of web-based services [using SOAP, XML over HTTP(S) or HTML over HTTP(S)], and Java™-based application provider interfaces (API) and service provider interfaces (SPI). Additionally, the figure shows an agent embedded into a web container in which a service provider application is deployed. This agent enables the service provider to participate in the SAML v1.x or Liberty ID-FF protocols.

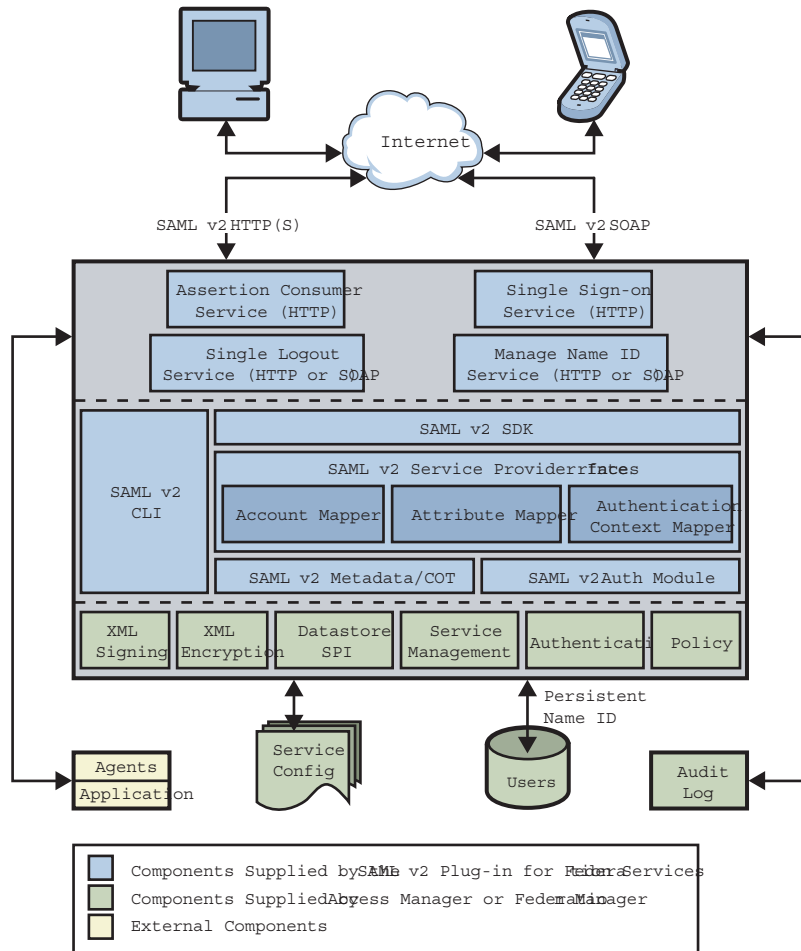


FIGURE 5-5 SAML v2 Architecture

More information about the SAML v2 framework can be found in the following sections:

- “Administration” on page 18
- “Application Programming Interfaces” on page 18
- “Service Provider Interfaces” on page 18
- “JavaServer Pages” on page 19

Administration

In order to communicate using the SAML v2 profiles you need, at least, two instances of Federated Access Manager. One instance will act for the identity provider and the other will act for the service provider. Name identifiers are used to communicate with each other regarding a user.

Note – SAML v2 single sign-on interactions support both *persistent* and *transient* identifiers. A persistent identifier is saved to a particular user entry as the value of two attributes. A transient identifier is temporary and no data will be written to the user's data store entry.

To prepare your instances for SAML v2 interactions, you need to exchange configuration information or *metadata* between all participating identity and service providers, and assemble the providers into a *circle of trust*. Utility APIs can then be used to communicate with the data store, reading, writing, and managing the relevant properties and property values. For more information see the XXXXXX.

Application Programming Interfaces

The SAML v2 framework contains API that can be used to construct and process assertions, requests, and responses. The SAML v2 Java API packages include:

- The `com.sun.identity.saml2.assertion` package provides interfaces to construct and process SAML v2 assertions. It also contains the `AssertionFactory`, a factory class used to obtain instances of the objects defined in the assertion schema.
- The `com.sun.identity.saml2.common` package provides interfaces and classes used to define common SAML v2 utilities and constants.
- The `com.sun.identity.saml2.protocol` package provides interfaces used to construct and process the SAML v2 requests and responses. It also contains the `ProtocolFactory`, a factory class used to obtain object instances for concrete elements in the protocol schema.

More information can be found in “Using the SAML v2 SDK” in *Sun Java System Federated Access Manager 8.0 Developer's Guide* and the *Federated Access Manager 8.0 Java API Reference*.

Service Provider Interfaces

The `com.sun.identity.saml2.plugins` package provides pluggable interfaces to implement SAML v2 functionality into your application. Default implementations are provided, but a customized implementation can be plugged in by modifying the corresponding attribute in the entity provider's extended metadata configuration file. The interfaces include mappers for:

- Account mapping (map between the account referred to in the incoming request and the local user account)

- Attribute mapping (specifies which set of user attributes in an identity provider user account needs to be included in an assertion, and maps the included attributes to attributes in the user account defined by the service provider)
- Authentication context mapping (map between Authentication Contexts defined in the SAML v2 specifications and authentication framework schemes defined in Federated Access Manager (user/module/service/role/level based authentication))

More information can be found in “Service Provider Interfaces” in *Sun Java System Federated Access Manager 8.0 Developer’s Guide* and the *Federated Access Manager 8.0 Java API Reference*.

JavaServer Pages

The SAML v2 framework provides JSP that can be used to initiate single sign-on, single logout and termination requests from either the identity provider or the service provider using a web browser. The JSP accept query parameters to allow flexibility in constructing SAML v2 requests; they can be modified for your deployment. More information can be found in “JavaServer Pages” in *Sun Java System Federated Access Manager 8.0 Developer’s Guide*.

About SAML v1.x

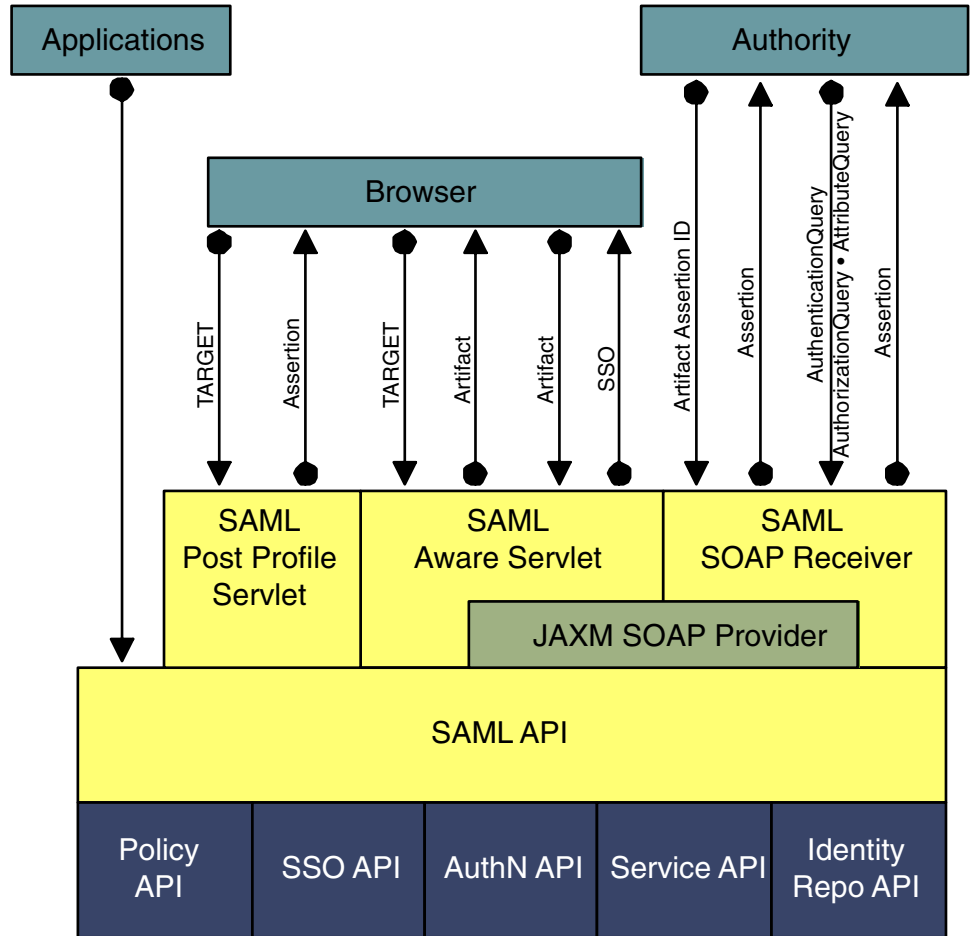
Federated Access Manager can be configured to use SAML v1.x to achieve interoperability between vendor platforms that provide SAML v1.x assertions. The SAML v1.x protocol defines the *assertion* security token format as well as *profiles* that standardize the HTTP exchanges required to transfer XML requests and responses between a SAML v1.x authority and a trusted partner site. An *assertion* is a package of verified security information that supplies one or more statements concerning a subject’s authentication status, access authorization decisions, or identity attributes. (A person identified by an email address is a subject as might be a printer.) Assertions are issued by a SAML v1.x asserting authority (a platform or application that declares whether a subject has been authenticated into its system), and received by relying parties (partner sites defined by the authority as *trusted*). SAML v1.x authorities use different sources to configure the assertion information, including external data stores or assertions that have already been received and verified. SAML v1.x can be used to allow Federated Access Manager to:

- Authenticate users and access trusted partner sites without having to reauthenticate; in effect, single sign-on. This single sign-on process independent of the process enabled by Access Manager user session management.
- Act as a policy decision point (PDP), allowing external applications to access user authorization information for the purpose of granting or denying access to their resources. For example, employees of an organization can be allowed to order office supplies from suppliers if they are authorized to do so.
- Act as both an attribute authority that allows trusted partner sites to query a subject’s attributes, and an authentication authority that allows trusted partner sites to query a subject’s authentication information.

- Validate parties in different security domains for the purpose of performing business transactions.
- Build Authentication, Authorization Decision, and Attribute Assertions using the SAML v1.x API.
- Permit an XML-based digital signature signing and verifying functionality to be plugged in.

Note – Although Liberty ID-FF (as described in About the Identity Federation Framework) integrates aspects of the SAML v1.x specifications, its usage of SAML v1.x is independent of the SAML v1.x framework as described in this section.

The following figure illustrates how SAML v1.x interacts with the other components in Federated Access Manager.



The lighter-shaded boxes are components of the SAML module.

FIGURE 5-6 SAML v1.x Interaction in Federated Access Manager

Comparison of SAML v1.x and the Liberty Alliance Project Specifications

SAML v1.x was designed to address the issue of cross-domain single sign-on. The Liberty Alliance Project was formed to develop technical specifications that would solve business process problems. These issues include single sign-on, but also incorporate protocols for account linking and consent, among others. SAML v1.x, on the other hand, does not solve issues such as privacy, single logout, and federation termination.

The SAML v1.x specifications and the Liberty Alliance Project specifications do not compete with one another. They are complementary. In fact, the Liberty Alliance Project specifications

leverage profiles from the SAML specifications. The decision of whether to use SAML v1.x or the Liberty specifications depends on your goal. In general, SAML v1.x should suffice for single sign-on basics. The Liberty Alliance Project specifications can be used for more sophisticated functions and capabilities, such as global sign-out, attribute sharing, web services. The following table compares the benefits of the two.

TABLE 5-1 Comparison of the SAML v1.x and Liberty Alliance Project Specifications

SAML v1.x Uses	Liberty Alliance Project Uses
Cross-domain single sign-on	Single sign-on <i>only</i> after user federation
No user federation	User federation
No privacy control, best for use within one company	Built on top of SAML
User identifier is sent in plain text	User identifier is sent as a unique handle

Using the Liberty ID-FF

The Liberty Alliance Project was formed to develop technical specifications that would solve business process issues including single sign-on, federation and consent, among others. The Liberty Alliance Project Identity Federation Framework (Liberty ID-FF) uses a name identifier to pass identity data between identity providers and service providers. The *name identifier* is a randomly generated character string that is assigned to a principal and used to federate the principal's accounts at the identity provider and service provider sites. This pseudonym allows all providers to identify a principal without knowing the user's actual identity. The name identifier has meaning only in the context of the relationship between providers. SAML v1.x is used for provider interaction.

Note – Liberty ID-FF was initially defined as an extension of SAML 1.0 (and later SAML 1.1). The extensions have now been contributed back into SAML v2 which, going forward, will be the basis on which the Liberty Alliance Project builds additional federated identity applications. See [“Using SAML” on page 13](#) for more information on this convergence.

The following sections contain information about the Liberty ID-FF and the features implemented in Federated Access Manager.

- [“About the Liberty ID-FF Process” on page 23](#)
- [“Liberty ID-FF Features” on page 26](#)

About the Liberty ID-FF Process

The Liberty ID-FF is designed to work with heterogeneous platforms, various networking devices (including personal computers, mobile phones, and personal digital assistants), and emerging technologies. The process of Liberty ID-FF federation begins with authentication. A user attempting to access a resource protected by Federated Access Manager are redirected to the proprietary Authentication Service via an Federated Access Manager login page. After the user provides credentials, the Authentication Service allows or denies access to the resource based on the outcome.

Note – For more information about the proprietary Authentication Service, see the [Chapter 3, “Authentication.”](#)

When the user attempts access to a resource that is a trusted member provider of a configured circle of trust using the Liberty ID-FF, the process of user authentication begins with the search for a valid Federated Access Manager session token from the proprietary Authentication Service. The process can go in one of two directions based on whether a session token is found.

- If no session token is found, the principal is redirected to a location defined by the pre-login URL to establish a valid session. See [“Pre-login Process” on page 25](#) for details.
- If a session token is found, the principal is granted (or denied) access to the requested page. Assuming access is granted, the requested page contains a link so the principal may federate the Federated Access Manager identity with the identity local to the requested site. If the principal clicks this link, federation begins. See [“Liberty ID-FF Federation and Single Sign-On” on page 25](#) for details.

The following figure illustrates these divergent paths. The process shown is the default process when no application has been deployed. When an application is deployed and using Federated Access Manager, the process will change based on the query parameters and preferences passed to Federated Access Manager from the participating application. For more information, see XXXXXX The Pre-login URL.

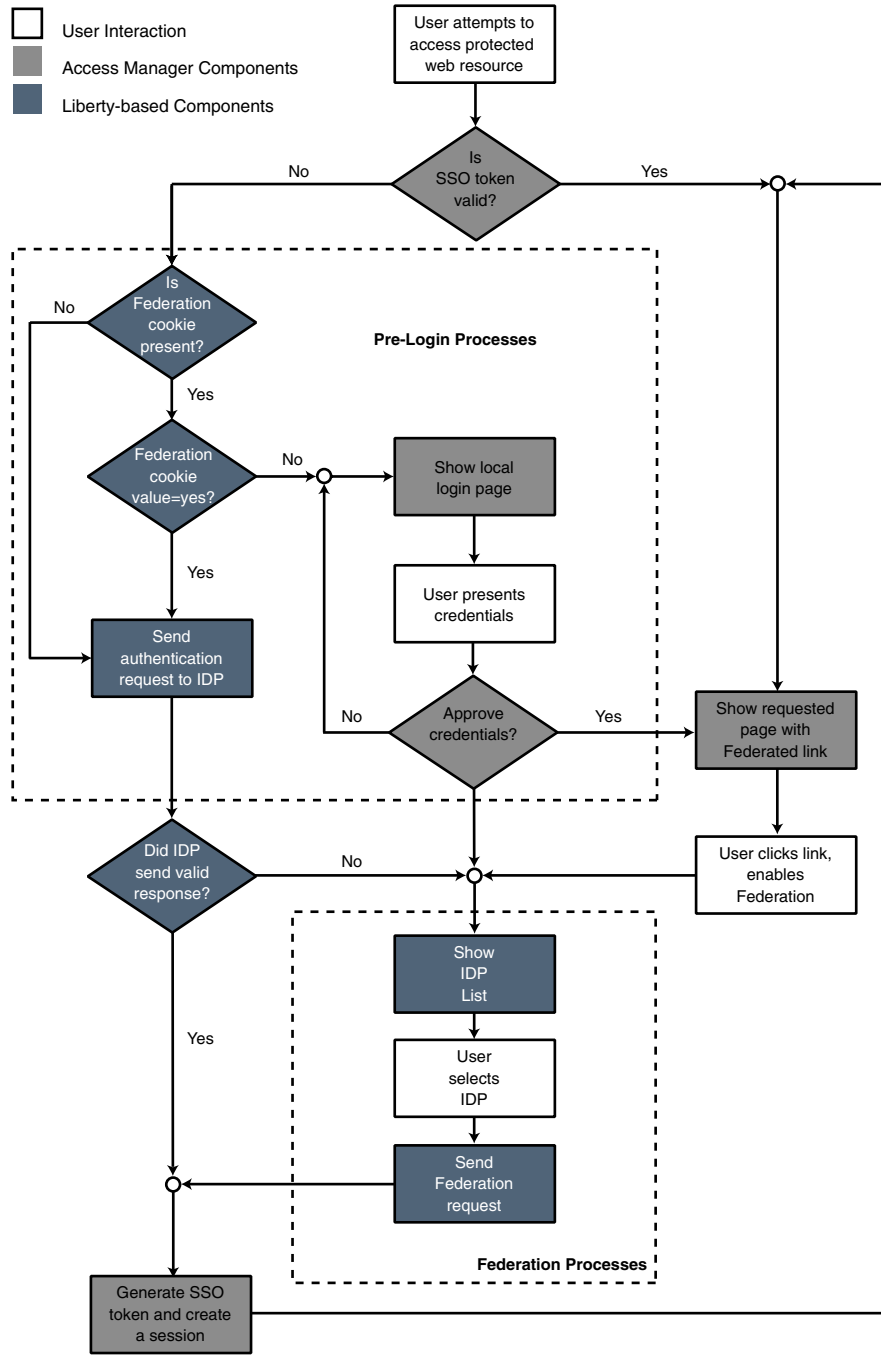


FIGURE 5-7 Default Process of Federation

Pre-login Process

The pre-login process establishes a valid Federated Access Manager session. When a principal attempts to access a service provider site and no Federated Access Manager session token is found, Federated Access Manager searches for a federation cookie. A *federation cookie* is implemented by Federated Access Manager and is called `fedCookie`. It can have a value of either `yes` or `no`, based on the principal's federation status.

Note – A federation cookie is *not* defined in the Liberty Alliance Project specifications.

At this point, the pre-login process may take one of the following paths:

- If a federation cookie is found and its value is `no`, a Federated Access Manager login page is displayed and the principal submits credentials to the proprietary Authentication Service. When authenticated by Federated Access Manager, the principal is redirected to the requested page, which might contain a link to allow for identity federation. If the principal clicks this link, federation begins. See [“Liberty ID-FF Federation and Single Sign-On” on page 25](#) for details.
- If a federation cookie is found and its value is `yes`, the principal has already federated an identity but has not been authenticated by an identity provider within the circle of trust for this Federated Access Manager session. Authentication to Federated Access Manager is achieved on the back end by sending a request to the principal's identity provider. After authentication, the principal is directed back to the requested page.
- If no federation cookie is found, a *passive* authentication request (one that does not allow identity provider interaction with the principal) is sent to the principal's identity provider. If an affirmative authentication is received back from the identity provider, the principal is directed to the Federated Access Manager Authentication Service, where a session token is granted. The principal is then redirected to the requested page. If the response from the identity provider is negative (for example, if the session has timed out), the principal is sent to a common login page to complete either a local login or Liberty ID-FF federation. See [“Liberty ID-FF Federation and Single Sign-On” on page 25](#) for details.

Note – This pre-login process is the default behavior of Federated Access Manager. This process might change based on parameters passed to Federated Access Manager from the participating application. For more details, see the section on XXXXXX The Pre-login URL.

Liberty ID-FF Federation and Single Sign-On

When a principal logs in to access a protected resource or service, Federated Access Manager sends a request to the appropriate identity provider for authentication confirmation. If the identity provider sends a positive response, the principal gains access to all provider sites within the circle of trust. If the identity provider sends a negative response, the principal is directed to authenticate again using the Liberty ID-FF process.

In the Liberty ID-FF process, a principal selects an identity provider and sends credentials for authentication. After authentication is complete and access is granted, the principal is issued a session token from the Federated Access Manager Authentication Service and redirected to the requested page. As long as the session token remains valid, the principal can access other service providers in the authentication domain without having to authenticate again.

Note – The Common Domain for Identity Provider Discovery is used by a service provider to determine the identity provider used by a principal in a circle of trust that contains multiple identity providers. See [“Creating a Common Domain for Identity Provider Discovery” on page 33](#) for details.

Liberty ID-FF Features

The following sections contain information about the Liberty ID-FF features implemented in Federated Access Manager.

- [“Identity Federation and Single Sign-On” on page 26](#)
- [“Authentication and Authentication Context” on page 28](#)
- [“Identifiers and Name Registration” on page 30](#)
- [“Global Logout” on page 30](#)
- [“Dynamic Identity Provider Proxying” on page 30](#)

Identity Federation and Single Sign-On

Let's assume that a principal has separate user accounts with a service provider and an identity provider in the same circle of trust. In order to gain access to these individual accounts, the principal would authenticate with each provider separately. If federating with the Liberty ID-FF though, after authenticating with the service provider, the principal may be given the option to federate the service provider account with the identity provider account. Consenting to the federation of these accounts links them for SSO, the means of passing a user's credentials between applications without the user having to reauthenticate. SSO and federated SSO have different processes. With Federated Access Manager, you can achieve SSO in the following ways:

- Install a policy agent in a web container to protect the application and pass the HTTP_HEADER and REMOTE_USER variables to the application to capture the user credentials. You may or may not need a custom authentication module.
- Customize the application's authentication module to create an SSOToken from the request object or from the SSO cookie. Afterwards, retrieve the user credentials using the SSO API and create a session data structure using the application's API.

To set up federated SSO, you must first establish SSO. Following that, enable federation in the metadata for the service provider entity and the identity provider entity using Federated Access Manager. Liberty ID-FF providers differentiate between federated users by defining a unique

identifier for each account. (They are not required to use the principal's actual provider account identifier.) Providers can also choose to create multiple identifiers for a particular principal. However, identity providers must create one handle per user for service providers that have multiple web sites so that the handle can be resolved across all of them.

Note – Because both the identity provider entity and the service provider entity in a federation need to remember the principal's identifier, they create entries that note the value in their respective user repositories. In most scenarios, the identity provider's identifier is conveyed to a service provider and not vica versa. For example, if a service provider does not maintain its own user repository, the identity provider's identifier is used.

Federated Access Manager can accommodate the following SSO and federation-related functions:

- Providers of either type notify the principal upon identity federation or defederation.
- Providers of either type notify each other regarding a principal's defederation.
- Identity providers notify the appropriate service providers regarding a principal's account termination.
- Providers of either type display a list of federated identities to the principal.
- Users can terminate federations or defederate identities.

Additionally, Federated Access Manager can accommodate the features explained in the following sections.

- [“Auto-Federation” on page 27](#)
- [“Bulk Federation” on page 27](#)

Auto-Federation

Auto federation will automatically federate a user's disparate provider accounts based on a common attribute. During SSO, if it is deemed a user at provider A and a user at provider B have the same value for the defined common attribute (for example, an email address), the two accounts will be federated without consent or interaction from the principal. For more information, see XXXXXXXX Auto-Federation.

Bulk Federation

Federating one user's service provider account with their identity provider account generally requires the principal to visit both providers and link them. An organization though needs the ability to federate user accounts behind the scenes. Federated Access Manager provides a script for federating user accounts in bulk. The script allows the administrator to federate many (or all) of a principal's provider accounts based on metadata passed to the script. Bulk federation is useful when adding a new service provider to an enterprise so you can federate a group of existing employees to the new service. For more information, see XXXXXXXX Bulk Federation.

Authentication and Authentication Context

SSO is the means by which a provider of either type can convey to another provider that a principal has been authenticated. Authentication is the process of validating user credentials; for example, a user identifier accompanied by an associated password. You can authenticate users with Federated Access Manager in the following ways:

- Use a policy agent to insert HTTP header variables into the request object. This functions for web applications only.
- Use the authentication API to validate and retrieve user identity data. This will work with either web or non-web applications.

Identity providers use local (to the identity provider) session information mapped to a user agent as the basis for issuing Security Assertion Markup Language (SAML) authentication assertions to service providers. Thus, when the principal uses a user agent to interact with a service provider, the service provider requests authentication information from the identity provider based on the user agent's session information. If this information indicates that the user agent's session is presently active, the identity provider will return a positive authentication response to the service provider. Federated Access Manager allows providers to exchange the following minimum set of authentication information with regard to a principal.

- Authentication status (active or not)
- Instant (time authenticated)
- Authentication method
- Pseudonym (temporary or persistent)

SAML v1.x is used for provider interaction during authentication but not all SAML assertions are equal. Different authorities issue SAML assertions of different quality. Therefore, the Liberty ID-FF defines how the consumer of a SAML assertion can determine the amount of assurance to place in the assertion. This is referred to as the *authentication context*, information added to the SAML assertion that gives the assertion consumer the details they need to make an informed entitlement decision. For example, a principal uses a simple identifier and a self-chosen password to authenticate to a service provider. The identity provider sends an assertion to a second service provider that states how the principal was authenticated to the first service provider. By including the authentication context, the second service provider can place an appropriate level of assurance on the associated assertion. If the service provider were a bank, they might require stronger authentication than that which has been used and respond to the identity provider with a request to authenticate the user again using a more stringent context. The authentication context information sent in the assertion might include:

- The initial user identification mechanism (for example, face-to-face, online, or shared secret).
- The mechanisms for minimizing compromise of credentials (for example, private key in hardware, credential renewal frequency, or client-side key generation).
- The mechanisms for storing and protecting credentials (for example, smart card, or password rules).

- The authentication mechanisms (for example, password or smart card with PIN).

The Liberty ID-FF specifications define authentication context *classes* against which an identity provider can claim conformance. The Liberty ID-FF authentication contexts are listed and described in the following table.

TABLE 5-2 Authentication Context Classes

Class	Description
MobileContract	Identified when a mobile principal has an identity for which the identity provider has vouched.
MobileDigitalID	Identified by detailed and verified registration procedures, a user's consent to sign and authorize transactions, and DigitalID-based authentication.
MobileUnregistered	Identified when the real identity of a mobile principal has not been strongly verified.
Password	Identified when a principal authenticates to an identity provider by using a password over an unprotected HTTP session.
Password-ProtectedTransport	Identified when a principal authenticates to an identity provider by using a password over an SSL-protected session.
Previous-Session	<p>Identified when an identity provider must authenticate a principal for a current authentication event and the principal has previously authenticated to the identity provider. This affirms to the service provider a time lapse from the principal's current resource access request.</p> <p>Note – The context for the previously authenticated session is not included in this class because the user has not authenticated during this session. Thus, the mechanism that the user employed to authenticate in a previous session should not be used as part of a decision on whether to now allow access to a resource.</p>
Smartcard	Identified when a principal uses a smart card to authenticate to an identity provider.
Smartcard-PKI	Identified when a principal uses a smart card with an enclosed private key and a PIN to authenticate to an identity provider.
Software-PKI	Identified when a principal uses an X.509 certificate stored in software to authenticate to the identity provider over an SSL-protected session.

TABLE 5-2 Authentication Context Classes (Continued)

Class	Description
Time-Sync-Token	Identified when a principal authenticates through a time synchronization token.

For more information, see the [Liberty ID-FF Authentication Context Specification](#). Additionally, there is an XML schema defined which the identity provider authority can use to incorporate the context of the authentication in the SAML assertions it issues.

Identifiers and Name Registration

Federated Access Manager supports name identifiers that are unique across all providers in a circle of trust. This identifier can be used to obtain information for or about the principal without requiring the user to consent to a long-term relationship with the service provider. When beginning federation, the identity provider generates an opaque value that serves as the initial name identifier that both the service provider and the identity provider use to refer to the principal when communicating with each other. After federation, the identity provider or the service provider may register a different opaque value. If a service provider registers a different opaque value for the principal, the identity provider must use the new identifier when communicating with the service provider about the principal. The reasons for changing an identifier would be implementation-specific. The initial name identifier defined by the identity provider is always used to refer to the principal unless a new name identifier is registered.

Global Logout

A principal may establish authenticated sessions with both an identity provider and individual service providers, based on authentication assertions supplied by the identity provider. When the principal logs out of a service provider session, the service provider sends a logout message to the identity provider that provided the authentication for that session. When this happen, or the principal manually logs out of a session at an identity provider, the identity provider sends a logout message to each service provider to which it provided authentication assertions under the relevant session. The one exception is the service provider that sent the logout request to the identity provider.

Dynamic Identity Provider Proxying

An identity provider can choose to proxy an authentication request to an identity provider in another authentication domain if it knows that the principal has been authenticated with this identity provider. The proxy behavior is defined by the local policy of the proxying identity provider. However, a service provider can override this behavior and choose not to proxy. This function can be implemented as a form of authentication when, for instance, a roaming mobile user accesses a service provider that is not part of the mobile home network. For more information see XXXXXX Dynamic Identity Provider Proxying.

Using WS-Federation

WS-Federation is part of the larger Web Services Security (WS-Security) framework which provides a means for applying security to web services through the use of security tokens. WS-Security describes how to attach signature and encryption headers as well as security tokens (including binary security tokens such as X.509 certificates and Kerberos tickets) to SOAP messages. WS-Trust, another specification in the WS-Security framework, provides for federation by defining a Security Token Service (STS) and a protocol for requesting and issuing the security tokens. WS-Federation, as implemented in Federated Access Manager, uses the Federated Access Manager Security Token Service (modelled on the WS-Trust specification) to allow providers in different security realms to broker trust using information on identities, identity attributes and authentication, and provider federation. A principal requests a token from the Security Token Services. This token, which may represent the principal's primary identity, a pseudonym, or the appropriate attributes, is presented to the service provider for authentication and authorization. WS-Federation uses several security tokens as well as the mechanism for associating them with messages. This release of Federated Access Manager has implemented the following features of the WS-Federation specification.

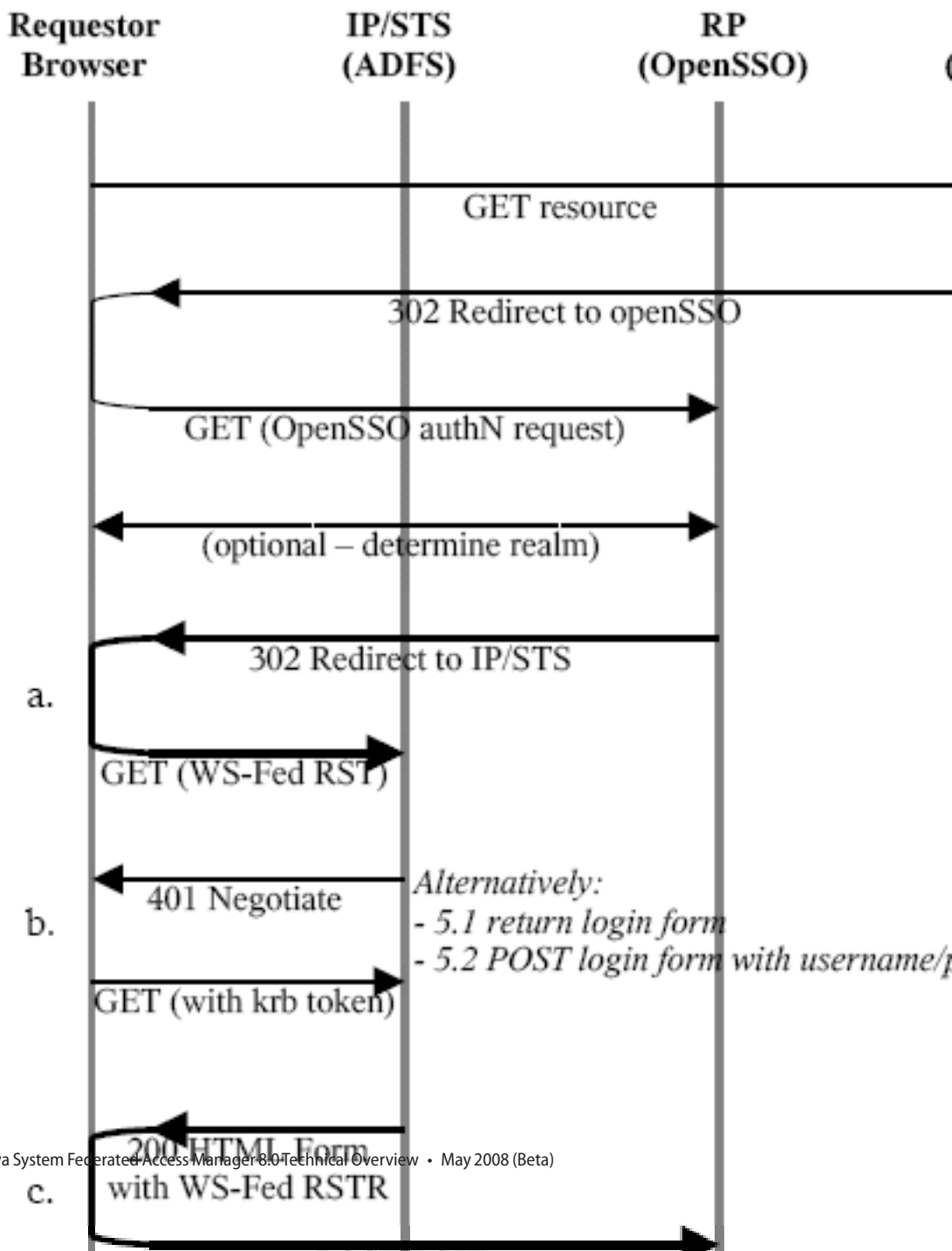
- The *Web (Passive) Profile* defines single sign-on, single logout, attribute and pseudonym token exchanges for passive requestors; for example, a web browser that supports HTTP. For the passive mechanisms to provide a single or reduced sign-on, there needs to be a service that will verify that the claimed requestor is really the requestor. Initial verification **MUST** occur in a secure environment; for example, using SSL/TLS or HTTP/S. The token is abstract and the token exchange is based on the Security Token Service model of WS-Trust.
- Tokens based on the *Web Services-Interoperability Basic Security Profile* (WS-I BSP) define security that is implemented inside a SOAP message; and security implemented at the transport layer, using HTTPS. The protocol covers how you generate or handle security tokens.

The WS-Federation implementation in Federated Access Manager is based on the application's SAML v2 code and uses WS-Federation 1.1 metadata. Authentication request parameters are represented directly as GET parameters, and the authentication response is a WS-Trust `RequestSecurityTokenResponse` element.

Note – There is no authentication context mapping, persistent or transient NameID identifiers or auto-federation in the Federated Access Manager implementation of WS-Federation.

The entry points for all WS-Federation functionality will be implemented as servlets. JavaServer Pages (JSP) are used only for HTML content (for example, the HTML form used to send the WS-Federation single response from the identity provider to the service provider). The following figure illustrates the flow of messages between Federated Access Manager (acting as the service provider) and the Active Directory (acting as the identity provider).

Single Sign-On – ADFS as IP/STS, OpenSSO as RP:



In a WS-Federation interaction, an unauthenticated user attempting to access a protected web site. The site redirects the user to the Active Directory for Federation Services (ADFS) identity provider. After the user is authenticated (either by a back-end single sign-on or by entering credentials), ADFS posts a form containing a signed SAML assertion to the service provider. The service provider validates the assertion, copies the attributes into the user's session, and gives the appropriate access.

Note – Microsoft Active Directory Federation Services supports single sign-on via WS-Federation.

Creating a Common Domain for Identity Provider Discovery

Service providers need a way to determine which identity provider is used by a principal requesting authentication. Because circles of trust are configured without regard to their location, this function must work across DNS-defined domains. A common domain is configured, and a common domain cookie written, for this purpose.

Let's suppose a circle of trust contains more than one identity provider. In this case, a service provider trusts more than one identity provider so, when a principal needs authentication, the service provider with which the principal is communicating must have the means to determine the correct identity provider. To ascertain a principal's identity provider, the service provider invokes a protocol exchange to retrieve the *common domain cookie*, a cookie written for the purpose of *introducing* the identity provider to the service provider. If no common domain cookie is found, the service provider will present a list of trusted identity providers from which the principal can choose. After successful authentication, the identity provider writes (using the configured Writer Service URL) a common domain cookie and, the next time the principal attempts to access a service, the service provider finds and reads the common domain cookie (using the configured Reader Service URL), to determine the identity provider. More information on the Common Domain for Identity Provider Discovery is available in the following sections:

- [“The Common Domain” on page 33](#)
- [“The Common Domain Cookie” on page 34](#)
- [“The Writer Service and the Reader Service” on page 34](#)

The Common Domain

The *common domain* is established for use only within the scope of identity provider discovery in a defined circle of trust. In Federated Access Manager deployments, the identity provider discovery WAR is deployed in a web container installed in a predetermined and pre-configured *common* domain so that the common domain cookie is accessible to all providers in the circle of trust. For example, if an identity provider is available at `http://www.Bank.com`, a service

provider is available at `http://www.Store.com`, and the defined common domain is `RetailGroup.com`, the addresses will be `Bank.RetailGroup.com` and `Store.RetailGroup.com`, respectively. If the HTTP server in the common domain is operated by the service provider, the service provider will redirect the user agent to the appropriate identity provider.

The Common Domain Cookie

After an identity provider authenticates a principal, the identity provider sets a URL-encoded cookie defined in a predetermined domain common to all identity providers and service providers in the circle of trust. The *common domain cookie* is named `_liberty_idp` for Liberty ID-FF and `_saml_idp` for SAML v2. After successful authentication, a principal's identity provider appends their particular encoded identifier to a list in the cookie. If their identifier is already present in the list, the identity provider may remove the initial appearance and append it again. The intent is that the service provider reads the last identifier on the cookie's list to find the principal's most recently established identity provider.

Note – The identifiers in the common domain cookie are a list of SuccinctID elements encoded in the Base64 format. One element maps to each identity provider in the circle of trust. Service providers then use this SuccinctID element to find the user's preferred identity provider.

The Writer Service and the Reader Service

After a principal authenticates with a particular identity provider, the identity provider redirects the principal's browser to the configured Writer Service URL using a parameter that indicates they are the identity provider for this principal. The Writer Service then writes a cookie using the parameter. Thereafter, all providers configured in this common domain will be able to tell which identity provider is used by this principal. Thus, the next time the principal attempts to access a service hosted by a service provider in the same common domain, the service provider retrieves and reads the common domain cookie, using the configured Reader Service URL, to determine the identity provider.

The Writer Service URL and the Reader Service URL can be defined for use with the Liberty ID-FF or the SAML v2 federation protocol. The URLs are defined when you create a circle of trust for federation. The Common Domain for Identity Provider Discovery for Liberty ID-FF is based on the Identity Provider Introduction Profile detailed in the [Liberty ID-FF Bindings and Profiles Specifications](#). The Common Domain for Identity Provider Discovery for SAML v2 is an implementation of the Identity Provider Discovery Profile as described in the [Profiles for the OASIS Security Assertion Markup Language \(SAML\) V2.0](#) specification.

Index

A

- API, SAML v2, 18
- application programming interfaces, *See* API
- architecture, SAML v1.x, 19-22
- authentication context, overview, 28-30
- auto-federation, 27

B

- bulk federation, 27

C

- circle of trust, definition, 10-12
- common domain, 33-34
 - reader service, 34
 - writer service, 34
- common domain cookie, 34
- cookies, common domain, 34

D

- definitions
 - circle of trust, 10-12
 - entity provider, 10-12
 - federation, 8-9
 - identity, 8
 - identity federation, 9
 - identity provider, 10-12
 - principal, 10-12

- definitions (*Continued*)

- provider federation, 9
 - service provider, 10-12
 - trust, 9-10
- dynamic identity provider proxying, Liberty ID-FF, 30

E

- entity providers, 10-12

F

- federated identity, 7-10
- federation, 7-34
 - common domain, 33-34
 - definition, 8-9
 - identity federation and single sign-on, 26-27

G

- global logout, Liberty ID-FF, 30

I

- identifiers, Liberty ID-FF, 30
- identity, definition, 8
- identity federation, 10-12, 26-27
 - definition, 7-10
- identity providers, definition, 10-12

J

JavaServer Pages, *See* JSP
JSP, SAML v2, 19

L

Liberty Alliance Project, SAML v1.x
 comparison, 21-22
Liberty Alliance Project Identity Federation
 Framework, *See* Liberty ID-FF
Liberty ID-FF, 22-30
 and single sign-on, 25-26
 auto-federation, 27
 bulk federation, 27
 convergence with SAML, 15-19
 dynamic identity provider proxying, 30
 federation and single sign-on, 25-26
 global logout, 30
 identifiers and name registration, 30
 pre-login process, 25
local identity, 7-10

N

name registration, Liberty ID-FF, 30

O

overview, authentication and authentication
 context, 28-30

P

PDP, in SAML, 19
pre-login process, Liberty ID-FF, 25
principal, definition, 10-12
provider federation, definition, 9

R

reader service, 34

S

SAML, convergence with Liberty ID-FF, 15-19
SAML v1.x
 architecture, 19-22
 federation, 13-22
 Liberty Alliance Project comparison, 21-22
SAML v2, 15-19
 administration, 18
 API, 18
 basic configuration, 18
 federation, 13-22
 JSP, 19
 SPI, 18-19
service provider interfaces, *See* SPI
service providers, definition, 10-12
single sign-on, 26-27
single sign—on, and Liberty ID-FF, 25-26
SPI, SAML v2, 18-19

T

trust, definition, 9-10
trust agreements, 9-10

W

writer service, 34