Sun OpenSSO Enterprise 8.0 Deployment Planning Guide

Beta

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

Part No: 820–3746
November 2008

081106@21288

Contents

Preface

The *Sun™ OpenSSO Enterprise Deployment Planning Guide* provides information to help you determine which OpenSSO Enterprise to use in your deployment. This guide contains deployment architecture diagrams, process flow diagrams, and benefits/tradeoff analysis for various OpenSSO Enterprise features.

The following topics are contained in this Preface:

Who Should Use This Guide

This guide is intended for a wide audience including: system administrators, system integrators, and others who architect the OpenSSO Enterprise environment and deploy OpenSSO Enterprise and related components.

- IT professionals responsible for architecting enterprise solutions.
- Company executives responsible for evaluating enterprise solutions and for approving IT vendors and purchases.
- System integrator and administrators responsible for deploying and managing OpenSSO Enterprise and related components.

Before You Read This Guide

Readers should be familiar with the following components and concepts:

- OpenSSO Enterprise technical concepts, as described in the *Sun OpenSSO Enterprise 8.0 Technical Overview*
- Deployment platform: Solaris™, Linux, or Windows operating system
- Web container that will run OpenSSO Enterprise, such as Sun Java System Application Server, Sun Java System Web Server, BEA WebLogic, or IBM WebSphere Application Server
- Technical concepts: Lightweight Directory Access Protocol (LDAP), Java™ technology, JavaServer Pages™ (JSP™) technology, HyperText Transfer Protocol (HTTP), HyperText Markup Language (HTML), and eXtensible Markup Language (XML)

How This Guide Is Organized

This guide is organized in two parts as follows:

Part I: Planning the Overall Deployment

- Chapter 1, "Seeing the Big Picture"
- Chapter 2, "Building the Deployment Architecture"
- Chapter 3, "Building the Implementation Plan"

Part II: Determining Which Features to Deploy

- Chapter 4, "Using a Policy Agent and the Client SDK to Integrate Applications with OpenSSO Enterprise"
- Chapter 5, "Using the OpenSSO Enterprise Fedlet to Enable Identity Federation"
- Chapter 6, "Implementing a Virtual Federation Proxy (Secure Attributes Exchange) "
- Chapter 7, "Implementing a SAMLv2 Identity Provider Proxy"
- Chapter 8, "Using a Multi-Federation Protocol Hub"
- Chapter 9, "Enabling Web Services Federation Between Active Directory Federation Service and OpenSSO Enterprise"
- Chapter 10, "Securing Web Services Using ID-WSF (Liberty Alliance Specifications)"
- Chapter 11, "Securing Web Services Using Security Token Service (WS-* Specifications)"
- Chapter 12, "Enabling Single Sign-On Between Sun Identity Manager and OpenSSO Enterprise"
- Chapter 13, "Enabling Single Sign-On Using CA SiteMinder and OpenSSO Enterprise"
- Chapter 14, "Enabling Single Sign-On Using Oracle Access Manager and OpenSSO Enterprise"
- Chapter 15, "Using the Embedded Configuration Data Store for OpenSSO Enterprise"

Related Documentation

Related documentation is available as follows:

- "OpenSSO Enterprise Documentation Set" on page 13
- "Related Product Documentation" on page 14

OpenSSO Enterprise Documentation Set

The following table describes the OpenSSO Enterprise documentation set.

TABLE P–1 OpenSSO Enterprise Documentation Set

Title	Description
Sun OpenSSO Enterprise 8.0 Release Notes	Describes new features, installation notes, and known issues and limitations. The Release Notes are updated periodically after the initial release to describe any new features, patches, or problems.
Sun OpenSSO Enterprise 8.0 Installation and Configuration Guide	Provides information about installing and configuring OpenSSO Enterprise including OpenSSO Enterprise server, Administration Console only, client SDK, scripts and utilities, Distributed Authentication UI server, and session failover.
Sun OpenSSO Enterprise 8.0 Technical Overview	Provides an overview of how components work together to consolidate access control functions, and to protect enterprise assets and web-based applications. It also explains basic concepts and terminology.
Sun OpenSSO Enterprise 8.0 Deployment Planning Guide(This book)	(This book) Provides planning and deployment solutions for OpenSSO Enterprise.
Deployment Example: Single Sign-On, Load Balancing and Failover Using Sun OpenSSO Enterprise 8.0	Provides step-by-step instructions for deploying OpenSSO Enterprise in a single sign-on environment using load balancers and redundant systems for high availability.
Deployment Example: SAML v2 Using Sun OpenSSO Enterprise 8.0	Provides step-by-step instructions for deploying OpenSSO Enterprise to achieve identity federation among an Identity Provider and multiple Service Providers.
Sun OpenSSO Enterprise 8.0 Administration Guide	Describes how to use the OpenSSO Enterprise Administration Console as well as how to manage user and service data using the command-line interface (CLI).
Sun OpenSSO Enterprise 8.0 Administration Reference	Provides reference information for the OpenSSO Enterprise command-line interface (CLI), configuration attributes, log files, and error codes.

TABLE P–1 OpenSSO Enterprise Documentation Set *(Continued)*

Title	Description
Sun OpenSSO Enterprise 8.0 Developer's Guide	Provides information about customizing OpenSSO Enterprise and integrating its functionality into an organization's current technical infrastructure. It also provides details about the programmatic aspects of the product and its API.
Sun OpenSSO Enterprise 8.0 C API Reference for Application and Web Policy Agent Developers	Provides summaries of data types, structures, and functions that make up the public OpenSSO Enterprise C APIs.
Sun OpenSSO Enterprise 8.0 Java API Reference	Provides information about the implementation of Java packages in OpenSSO Enterprise.
Sun OpenSSO Enterprise 8.0 Performance Tuning Guide	Provides information about how to tune OpenSSO Enterprise and its related components for optimal performance.
Sun OpenSSO Enterprise 8.0 Integration Guide	Provides information about how to integrate Sun Identity Manager, CA SiteMinder, or Oracle Access Manager with OpenSSO Enterprise.
Sun OpenSSO Enterprise Policy Agent 3.0 User's Guide for J2EE Agents	Provides an overview of version 3.0 policy agents.

Related Product Documentation

The following table provides links to documentation collections for related products.

TABLE P–2 Related Product Documentation

Product	Link
Sun Java System Directory Server 6.3	`http://docs.sun.com/coll/1224.4`
Sun Java System Web Server 7.0 Update 3	`http://docs.sun.com/coll/1653.3`
Sun Java System Application Server 9.1	`http://docs.sun.com/coll/1343.4`
Sun Java System Message Queue 4.1	`http://docs.sun.com/coll/1307.3`
Sun Java System Web Proxy Server 4.0.6	`http://docs.sun.com/coll/1311.6`
Sun Identity Manager 8.0	`http://docs.sun.com/app/docs/coll/1514.5`

Searching Sun Product Documentation

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

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

For example, to search for "broker," type the following:

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

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

Related Third-Party Web Site References

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

Note – Sun is not responsible for the availability of third-party web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Documentation, Support, and Training

The Sun web site provides information about the following additional resources:

- Documentation (`http://www.sun.com/documentation/`)
- Support (`http://www.sun.com/support/`)
- Training (`http://www.sun.com/training/`)

Typographic Conventions

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

TABLE P–3 Typographic Conventions

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

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

aabbcc123	Placeholder: replace with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file. **Note:** Some emphasized items appear bold online.

Shell Prompts in Command Examples

The following table shows the default UNIX® system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P–4 Shell Prompts

Shell	Prompt
C shell	`machine_name%`
C shell for superuser	`machine_name#`
Bourne shell and Korn shell	`$`
Bourne shell and Korn shell for superuser	`#`

Default Paths and Directory Names

The OpenSSO Enterprise documentation uses the following terms to represent default paths and directory names:

TABLE P–5 Default Paths and Directory Names

Term	Description
zip-root	Represents the directory where the `opensso.zip` file is unzipped.

OpenSSO-Deploy-base	Represents the deployment directory where the web container deploys the `opensso.war` file.
	This value varies depending on the web container. To determine the value of *OpenSSO-Deploy-base*, view the file name in the `.openssocfg` directory, which resides in the home directory of the user who deployed the `opensso.war` file. For example, consider this scenario with Application Server 9.1 as the web container:
	■ Application Server 9.1 is installed in the default directory: `/opt/SUNWappserver`.
	■ The `opensso.war` file is deployed by super user (`root`) on Application Server 9.1.
	The `.openssocfg` directory is in the `root` home directory (`/`), and the file name in `.openssocfg` is:
	`AMConfig_opt_SUNWappserver_domains_domain1_applications_j2ee-modules_openss`
	Then, the value for *OpenSSO-Deploy-base* is:
	`/opt/SUNWappserver/domains/domain1/applications/j2ee-modules/opensso`

ConfigurationDirectory	Represents the name of the configuration directory specified during the initial configuration of OpenSSO Enterprise server instance using the Configurator. The default is `opensso` in the home directory of the user running the Configurator. Thus, if the Configurator is run by `root`, *ConfigurationDirectory* is `/opensso`.

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions.

To share your comments, go to http://docs.sun.com and click Send comments. In the online form, provide the document title and part number. The part number is a seven-digit or nine-digit number that can be found on the title page of the guide or at the top of the document.

For example, the title of this guide is the *Sun OpenSSO Enterprise Deployment Planning Guide*, and the part number is 820-3746.

PART I

Planning the Overall Deployment

CHAPTER 1

Seeing the Big Picture

Sun OpenSSO Enterprise 8.0 provides secure and centralized access control and single sign-on (SSO) in a unified solution. OpenSSO Enterprise is one component of the Sun Identity Management infrastructure. The Sun Identity Management infrastructure enables your organization to manage secure access to web applications and other resources both within your enterprise and across business-to-business (B2B) value chains.

This chapter provides an introduction to the core OpenSSO Enterprise functions that will form your deployment. The information in this chapter is designed help you envision how OpenSSO Enterprise can meet your business needs. Topics contained in this chapter are:

- "Understanding Identity and Access Management" on page 21
- "Leveraging Identity Federation" on page 24
- "Securing Web Services" on page 25
- "Using Identity as a Service" on page 31

Understanding Identity and Access Management

The growing number of web-enabled applications and the changing roles of different user communities creates challenges for the modern enterprise. These challenges include controlling access to network resources, maintaining the consistency of user identity between different applications, and making new applications easy to manage.

Companies typically develop and implement network applications in individual silos. Each application is deployed with its own provisioning and identity-management interfaces, and with its own security systems. This method of deployment results in a heterogeneous environment with no centralized management systems and no flexibility to scale as the enterprise changes. The silo approach to deployment can negate the benefits of Internet applications, and instead increase the costs of deployment, administration, and ownership. The silo approach can also expose the enterprise to security risks.

As companies deploy electronic business applications and services, management costs for existing information technology systems escalate. Identity information and security policies are distributed across many applications, and repositories are controlled by a variety of internal and external groups. The need for more flexible access and stronger security is hampered by administration redundancies. Administration redundancies can result in inconsistent identity data across the enterprise, increased operating costs, and an ad hoc security strategy.

OpenSSO Enterprise can help to ease the problems associated with the silo approach to deployment.

Dealing with Widely Distributed Identity Information

Environments with disparate sources of identity information have different approaches for organizing user entries, security practices, access control, and other essential aspects of information architecture. As complicated as internal identity issues can be, identity management issues also extend outside the individual enterprise. Enterprises with affiliate business and consumer relationships potentially have user populations that reach into the tens or hundreds of millions. You can use OpenSSO Enterprise in a federated identity management model to manage your business affiliate's users, and to ensure efficient and secure operating policies between your company and its business partners. The enterprise must also extend privacy and ease of use to the consumer and must consider the scalability requirements necessary to effectively manage hundreds of millions of users

Eliminating Ad Hoc Security Strategies

When new applications are deployed without a common identity infrastructure, security decisions are often made in an ad hoc manner by developers and system administrators. Line-of-business managers cannot ensure that the right people see the right content at the right time. Managers must enforce security policies centrally and apply them locally. Security policies define how users are identified or authenticated, and also which users are authorized to access specific information.

Some services or transactions require stronger forms of authentication than others. For some applications, a name and password might be sufficient. Other applications, for example, those that enable high-value financial transactions, can require increased levels of security. These stronger levels of security can be in the form of a digital certificate and personal identification number (PIN), depending on the information and transactions involved.

Authorization to view certain uniform resource locators (URLs) should be restricted to different sets of users based on the users roles. When roles change, changes to privileges should be propagated across all systems. For example, when an employee changes departments or leaves the company, information about that user should be modified or deleted across all accounts immediately. Inconsistent processes for account deactivation is one of the major security risks that enterprises face every day. When you develop applications with your own individual security and access controls, OpenSSO Enterprise provides a centralized security policy and infrastructure to mitigate the risks from both internal users and external threats.

Reducing Operational Inefficiency

Scattered identity data, duplication of identity infrastructure functions across multiple applications, and random security contribute to operational inefficiencies across the enterprise. As companies bring new applications and services online, they often create a separate identity infrastructure for each one. This duplication of effort increases costs, delays time to market, and reduces revenues.

New applications must be able to leverage the identity infrastructure easily and quickly, without affecting how existing services or systems work. As your company delivers new products and services, you can count on OpenSSO Enterprise to update role definitions and access privileges across multiple partners, suppliers, and customers. OpenSSO Enterprise provides an integrated identity management infrastructure solution that delivers economies of scale that lead to better overall operational efficiencies.

Enabling Effective Access Management

When you use OpenSSO Enterprise to extend your current infrastructure, you can bring together disparate identity data into a managed network identity to better serve customers, suppliers, employees, and partners.

- For the enterprise, network identity enables employees who have single sign-on (SSO) capability to access disparate applications, such as benefits registration and provisioning. At the same time, network identity simplifies integration between applications, and sets security levels across all of them.
- For customer management, network identity can assist in capturing customer interactions. This ensures tighter one-to-one relationships, including access to custom offerings, affinity marketing, and data mining.
- For the business partner, network identity helps provide integrated enterprise relationships with reduced risk of fraudulent transactions.

Common identity infrastructures enable administrators to consolidate redundant tasks that normally occur across many applications by multiple administrators. This consolidation of administration makes it possible to consistently delegate management tasks to partners, customers, and internal company departments based on business requirements. The result is an environment that is integrated, flexible, easy to manage, and secure. Security implementations and access control rules that are typically contained within each application can be consolidated to provide centralized authentication and authorization to resources.

The transition to an intelligent applications infrastructure requires you to implement a system that incorporates access management and user management. This implementation lets you centralize the administration or management of user identity and security policy information across multiple resources and enterprise applications. You can expect to respond to the ever-changing network environments and applications with an integrated, cost-effective solution.

Leveraging Identity Federation

Identity federation enables partner organizations to trust and share digital identities and attributes of employees, customers, and suppliers across domains. Identity federation is the means to providing single sign-on among partner sites.

Through identity federation, transactions involving multiple organizations can be managed and completed using a single identity. Customers or members can access a variety of online services through just one organization, using just one password. And employees of that organization and its partners can be given secure, as-needed access to selected information on partner sites. A federated identity allows a user from one federation partner to seamlessly access resources from another partner in a secure and trusted manner.

Why We Need It

Industries such as telecommunications or financial services are eager to meet customers' demands for online services. To meet these needs, companies seek partnerships with other companies to deliver the widest variety of services to customers. The growing customer demand for everything from ringtones to on-demand video, from online banking to investments, and much more requires partners to join forces to compete successfully.

How It Works

Service providers and other companies have agreed to a common set of rules for sharing identity information securely and privately. Identity federation is based on these standards. They allow multiple partners to access one personal identity on multiple sites at the same time and to authenticate that identity in order to deliver services securely. A common set of standards allows partnerships to repeat the same information-sharing processes with every partner. Otherwise, anytime a company wanted to create a partnership, it would have to create a whole new set of processes, based on the prospective partner's IT infrastructure, security policies, and other unique characteristics. This quickly becomes impossible as the number of partners increases. But with standards, the ability to partner is infinitely scalable. Using federation standards, organizations can create circles of trust in which a given provider at the center of the circle, such as an wireless provider, is surrounded by and connected to a multitude of other companies that offer value-added services the provider wants to deliver to customers.

How Identity Federation Can Benefit Your Business

The following are ways in which OpenSSO Enterprise identity federation can create a wide variety of new business opportunities for your company.

- Creates new revenue streams

  Identity federation means being able to create new sources of revenue by quickly meeting customers' ever-growing demand for online services. In addition to applying a standard set of protocols across partner domains, identity federation automates many manual processes within a secure identity framework, helping to deliver revenue-enhancing services to customers more quickly
- Improves allocation of resources

  Some organizations may want to outsource certain operations so that they can focus their resources on core competencies. Identity federation enables organizations to easily turn over such operations to partners without fear of breaching information security or privacy.
- Reduces operational cost and complexity

  Identity federation enables organizations to collaborate freely without the cost, complexity, and limitations of compiling and sharing manual lists of users or using proprietary web access management tools. It also makes it easier to ensure the security and privacy of shared information.
- Improves user experience

  Identity federation promotes loyalty by enabling users such as customers, employees, and suppliers to enjoy more services and products, more quickly and easily than ever before. In particular, single sign-on enables an exceptional online experience, eliminating the need to use multiple passwords for access to online services and products.
- Enhances enterprise security

  Identity federation enables employees of partner organizations use only one login. When a user has just one password to remember, he or she is less likely to have two write down that password, thus reducing the risk of the password being used by unauthorized entities.

Securing Web Services

Your enterprise solution must include a means for securing your web services from unauthorized use. OpenSSO Enterprise supports web services security using the Identity Web Services Framework (ID-WSF), part of the Liberty Specification. OpenSSO Enterprise also supports web services security using the Secure Token Service, which is defined in the WS-* Specification.

Web services are self-contained, modular applications that can be described, published, located, and invoked over a network. Web services perform encapsulated business functions, ranging from a simple request-reply to complete business process interactions. Web services based on the following allow data and applications to interact without manual intervention:

- eXtensible Markup Language (XML), SOAP (previously known as Simple Object Access Protocol), and related open standards
- Service Oriented Architectures (SOA)

A typical Web services application consists of a service consumer, a service provider, and optionally a registry for storing the Web services definitions. Web services are accessible over standard Internet protocols that are independent of platforms and programming languages. Web services technology can be implemented in a wide variety of architectures, can co-exist with other technologies and software design approaches, and can be adopted in an evolutionary manner without requiring major transformations to legacy applications and databases.

A number of technologies such as Remote Procedure Call (RPC) Common Object Requesting Broker Architecture (CORBA), Microsoft Distributed Component Object Model (DCOM) have been developed for application integration. However, the web services technology based on XML, SOAP and HTTP(S) has been accepted as an industry standard and has seen wide industry adoption. Interoperability has been a key reason for the success of web services because it is based on open standards. Enhancements to web services should preserve the interoperability and should be based on open standards.

Many of the features that make Web services attractive, including greater accessibility of data, dynamic application-to-application connections, and relative autonomy or lack of human intervention are at odds with traditional security models and controls. Network security technologies such as firewalls are inadequate to protect SOAs for the following reasons:

- SOAs are dynamic and can seldom be fully constrained to the physical boundaries of a single network.
- SOAP is transmitted over HyperText Transfer Protocol (HTTP), which is allowed to flow without restriction through most firewalls.
- Transport Layer Security technologies (like SSL/TLS) and Network Layer Security technologies (like TLS), which are used to authenticate and encrypt Web-based messages, are inadequate for protecting SOAP messages because they are designed to operate between two endpoints.
- SSL/TLS cannot accommodate Web services' inherent ability to forward messages to multiple other Web services simultaneously.

The Web service processing model requires the ability to secure SOAP messages and XML documents as they are forwarded along potentially long and complex chains of consumer, provider, and intermediary services. The nature of Web services processing makes those services subject to unique attacks, as well as variations on familiar attacks. According to WS-I, the top threats facing Web services are:

- Message alteration

  An attacker inserts, removes or modifies information within a message to deceive the receiver.
- Loss of confidentiality

Information within a message is disclosed to an unauthorized individual

- Falsified messages

  Fictitious messages that an attacker intends the receiver to believe are sent from a valid sender.
- Man in the middle

  A third party sits between the sender and provider and forwards messages such that the two participants are unaware, allowing the attacker to view and modify all messages
- Principal spoofing

  An attacker constructs and sends a message with credentials such that it appears to be from a different, authorized principal
- Forged claims

  An attacker constructs a message with false credentials that appear valid to the receiver.
- Replay of message

  An attacker resends a previously sent message
- Replay of message parts

  An attacker includes portions of one or more previously sent messages in a new message
- Denial of service.

  An attacker causes the system to expend resources disproportionately such that valid requests cannot be met.

The importance of these threats varies depending upon your company's needs and purpose. For most companies, internal messages must be kept confidential and loss of confidentiality is a primary concern. However, many companies offer web services to the public at large. For some services, identity authentication is not a significant concern. For example, a web service provider that serves information about the current weather forecast need not be concerned if a request is from a falsified sender. Regardless, it is important to understand these threats and what technologies are available to mitigate them.

Web Services Security Industry Specifications

OpenSSO Enterprise is based upon the following industry-recognized specifications:

- *Confidentiality of Web Service Messages Using XML Encryption*

  Produced by the World Wide Web Consortium (W3C). Describes a mechanism to encrypt XML documents.
- *Web Service Authentication and Authorization Using XML Signature*

Describes Secure Assertion Markup Language (SAML) and eXtensible Access Control Markup Language (XACML) as proposed by the Organization for Advancement of Structured Information Standards (OASIS) group. SAML and XACML provide mechanisms for authentication and authorization in a Web services environment.

- *Integrity of Web Service Messages Using XML Signature*Produced jointly by the W3C and the Internet Engineering Task Force (IETF). The power of XML Signature is in it ability to selective sign XML data.
- *Web Services (WS)-Security*

  Produced by OASIS. Defines a set of SOAP header extensions for end-to-end SOAP messaging security. WS-Security supports message integrity and confidentiality by allowing communicating partners to exchange signed encrypted messages in a web services environment.
- *Security for Universal Description, Discovery and Integration (UDDI)*

  Produced by OASIS. UDDI enables web services to be easily located and subsequently invoked. Security for UDDI enables publishers, inquirers and subscribers to authenticate themselves and to authorize the information published in the directory.

Security Infrastructure Requirements

In a simple web service transaction, a request is sent from the Web Service Client to a Web Service Provider through intermediaries such as load balancers and firewalls. Similarly, the response from the Web Service Provider to the Web Service Client is also sent through the same intermediaries. In order to protect the web service request, application-level end-to-end security must be enabled in addition to transport-level security.

The following diagram shows a simple web service call between the Web Service Client and Web Service Provider.

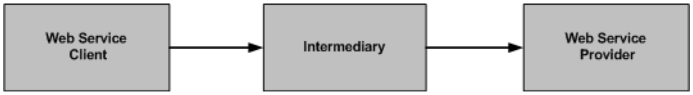


FIGURE 1–1 Simple Web Service Call

In order to secure the message, the Web Service Client must determine which security mechanisms are required by the Web Service Provider. One solution is to pre-configure the Web Service Client with the security requirements for Web Service Provider. Although simple, this approach would not scale and could lead to other misconfigured Web Service Clients.

An alternative architecture for web service security is an architecture based on Security Token Service (STS). The Liberty Alliance Discovery Service and WS-Trust are examples. A security token service that coordinates security-based interactions between a Web Service Client and Web Service Provider.

First, the Web Service Provider registers its acceptable security mechanisms with the security token service. Then, before making a call to the Web Service Provider, the Web Service Client connects with the Security Token Service to determine the required security mechanisms. The Web Service Client might also obtain the security tokens required by the Web Service Provider. Before validating the incoming SOAP request, Web Service Provider checks with the security token service to determine its security mechanisms. The following figure illustrates interactions between the Web Service Client, Web Service Provider, and Security Token Service.

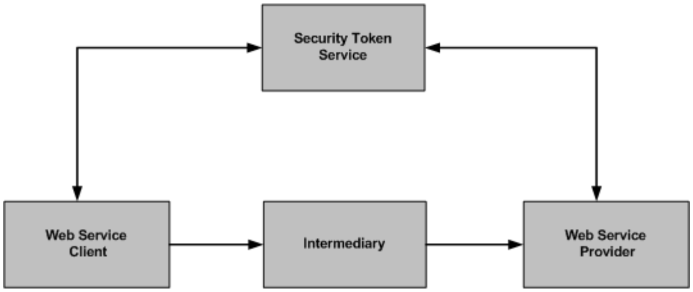


FIGURE 1–2 Web Service Call with Security Token Service Enabled

Although this security model requires the security token service, it helps in coordinating security mechanisms between the Web Service Client and Web Service Provider. Additionally, it enables runtime decisions for both Web Service Client and Web Service Provider. This makes the configuration dynamic and more responsive than a static configuration. However it does introduce the extra overhead of the Web Service Client and the Web Service Provider to communicating with the security token service. It also introduces the complexities of notification mechanisms when the Web Service Provider changes its security mechanisms. Your decision to either the static or dynamic configuration of Web Service Clients must be based on your deployment environment. The architecture proposed in this document addresses both the scenarios.

Security Token Service

The purpose of the security token service is to orchestrate secure communications between the Web Service Client and Web Service Provider with minimal performance penalties. The following are required for a security token service:

- Interfaces that enable the Web Service Provider to manage its entry, or resource offering. This includes interfaces that enable the Web Service Provider to store supported security mechanisms, and optionally the service end points.
- Interfaces that enable the Web Service Client to query for security mechanisms supported by a Web Service Provider.
- Interfaces that enable a Web Service Client to obtain security tokens for communicating with the Web Service Provider.

Liberty Alliance's Discovery Service and WS-Trust are the emerging standards specifications, and either one can play the role of the security token service. Both the specifications define the wire protocols for the Web Service Client to query and obtain the security tokens to communicate with the Web Service Provider. One important difference exists between the two. The Liberty Alliance Discovery Service provides the interfaces for the Web Service Provider to manage its entry in the secure token service. In WS-Trust specification, the WS-Trust entry is managed by the Web Service Provider itself. The WS-Trust entry is provided to the Web Service Client through a WS-Trust Meta-Data Exchange (MEX) Protocol.

Web Service Client

The Web Service Client which makes the web service call provides support for securing the outgoing communication, and also validates the incoming response for Web Service Provider. The Web Service Client security infrastructure requires the following:

- Configurations to determine STS and credentials to authenticate and obtain WSP resource offerings.

  Optionally there should be provision to statically configure the resource offering locally
- Interfaces to obtain WSP resource offering either from STS or optionally from the local configuration
- Interfaces to secure the request. This could be accomplished by calling the STS for the security token or should be locally generated. The security token generated could be either that of the WSC itself or it could be that of the authenticated entity (impersonalization)
- In addition to adding the security token it should be possible to add additional attributes of the identity, for example roles and memberships
- Interface to validate the response received from WSP.

Two kinds of interfaces are needed at the Web Service Client. One interface is needed for configuration and administration. One interface is used at run time for securing requests and validating responses.

Web Service Provider

The Web Service Provider provides support for validating the incoming request, and also secures the outgoing responses. The Web Service Provider security infrastructure requires the following:

- Configuration for its supported security mechanisms. This configuration can be optionally stored in STS, thereby providing dynamic discovery for WSCs. This is supported by Liberty Alliance's Discovery Service, but it in the case of WS-Trust this would have to locally configured for WS-Trust MEX calls.
- Interfaces to authenticate the incoming request from the Web Service Client
- After authentication, if configured, the Web Service Provider should also authorize the request for the web service operation by calling the policy component.
- Interfaces to secure the response back to the Web Service Client

Similar to interfaces needed by Web Service Client, the Web Service Provider also requires two kinds of interfaces. One interface is needed for configuration, and another interface is needed for validating requests and securing responses. Supporting a Web Service Client and Web Service Provider security infrastructure should be accomplished in either a pluggable manner such that it does not require reconfiguring the existing web services framework. Or it can be accomplished programmatically by calling well-defined interfaces to secure requests and validate responses. Additionally, the infrastructure should enable customers to easily build and configure interoperable solutions using heterogeneous systems.

Using Identity as a Service

OpenSSO Enterprise enables you to use Identity as a Service in your environment. Identity as a Service is a set of reusable, standardized services that provide applications with identity management. Typically based on the service-oriented architecture (SOA), Identity as a Service is system of discrete functional components of identity management. It is derived from the traditional set of functionally overlapping applications such as authentication, authorization, work flow, policy management, attribute management, provisioning, and password management.

The Identity As A Service environment contains these simplified services and makes them openly available to systems and applications. The services exist independently of one another, but together comprise a foundation of identity services upon which the overall IT environment relies. The primary advantage of Identity as a Service model is that the components can work in an independent fashion, or can be coupled together in the manner of an Enterprise Service Bus. Examples of Identity As a Service include:

- Authentication and Authorization Services
- Provisioning Services
- Taskflow/Workflow Services
- Role Management Services
- Audit Services

Simplifying Deployment and System Administration

Identity As a Service provides both IT and business benefits to enterprises. The IT benefits include:

- Easier Administration

  Performing administrative tasks and adding newer tasks is simplified as appropriate modular components can be invoked with ease.
- Flexible Deployment Architecture

  Allows deployments to effectively unify duplicate code and services and put forth a flexible and unified services.
- Simplified Outsourced and Federated Identity and Access Management

  Allows enterprises to leverage and outsource identity management services to system integrators and partners with core competency in the area.

Business benefits of Identity as a Service include:

- Reduced Operational risk and Maintenance Cost

  The Identity services layer is useful when adding new applications or services to an existing deployment, reconciling different identity management solutions acquired through mergers and acquisitions. It also allows you to centralize various Identity Management functions such as access management, resulting in reduced operational risk.
- Increased compliance

  With all applications using the same set of auditing services, audit log aggregation, and detection of violations of regulatory compliance, rules become easier to manage. With a common policy framework that spans applications, Identity as a Service simplify the management and improves the enforcement of complex segregation of duties policies.
- More Business Insight into Identity Management

  The traditional developer centric nomenclature used in identity management products has long been found hard by common users. With prevalent use of Identity As A Service and easy to use available interfaces, the deployments will be simpler to manage.

CHAPTER 2

Building the Deployment Architecture

A deployment architecture identifies the software components needed to meet your company's enterprise requirements, showing the interrelationships among the components. This chapter provides an overview of a typical OpenSSO Enterprise environment, and the technical requirements you need to consider as you plan your OpenSSO Enterprise deployment architecture.

The following topics are contained in this chapter:

- "Setting Deployment Goals" on page 33
- "Examining a Single Sign-On Deployment Example" on page 39
- "Designing the Deployment Architecture" on page 48

Setting Deployment Goals

You should consider several key factors when planning OpenSSO Enterprise deployment. These considerations generally deal with risk assessment and a growth strategy. For example:

- How many users is your deployment expected to support, and what is your projected growth rate?

  It is critical that user growth and system usage are monitored and that this data is compared with the projected data to ensure that the current capacity is capable of handling the projected growth.
- Do you have plans to add additional services that might impact the current design?

  The architecture you have in place now may be optimized for your company's current needs. Examine your future needs as well.

The following sections describe some basic functionality you should also consider when planning your OpenSSO Enterprise deployment.

Security

Consider the following options when you are planning for a secure internal and external OpenSSO Enterprise environment:

- Server-based firewalls provide an additional layer of security by locking down port-level access to the servers. As with standard firewalls, server-based firewalls lock down incoming and outgoing TCP/IP traffic.
- Minimization refers to removing all unnecessary software and services from the server in order to minimize the opportunity for exploitation of the vulnerabilities of a system.
- A Split-DNS infrastructure has two zones that are created in one domain. One zone is used by an organization's internal network clients, and the other is used by external network clients. This approach is recommended to ensure a higher level of security. The DNS servers can also use load balancers to improved performance.

High Availability

High availability refers to a system or component in the OpenSSO Enterprise environment that is continuously operational for a specified length of time. It is generally accomplished with multiple host servers that appear to the user as a single highly available system. Successful deployments strive for no single point of failure as well as for continuos availability to its users. Different products achieve availability in different ways. For example, clustering is the use of multiple computers to form a single, highly available system. Clustering is often crucial for the Sun Directory Server data store. A clustered multi-master replication (MMR) server pair can increase the availability of each master instance by ensuring availability.

In an OpenSSO Enterprise deployment that meets the minimal requirements, the single points of failure might include:

- Access manager web container
- Directory Server
- Java™ Virtual Machine (JVM)
- Directory Server hard disk
- Access Manager hard disk
- Policy agents

Planning for high availability centers around backup and failover processing as well as data storage and access. OpenSSO Enterprise provides session failover and SAML assertion failover functionality. For storage, a redundant array of independent disks (RAID) is one approach. For any system to be highly available, the parts of the system should be well-designed and thoroughly tested before they are used. A new application program that has not been thoroughly tested is likely to become a frequent point-of-breakdown in a production system.

Scalability

Horizontal scaling is achieved in the OpenSSO Enterprise environment by connecting multiple host servers so they work as one unit. A load-balanced service is considered horizontally scaled because it increases the speed and availability of the service. Vertical scaling, on the other hand, is increasing the capacity of existing hardware by adding resources within a single host server. The types of resources that can be scaled include CPUs, memory, and storage. Horizontal scaling and vertical scaling are not mutually exclusive; they can work together for a deployment solution. Typically, servers in an environment are not installed at full capacity, so vertical scaling is used to improve performance. When a server approaches full capacity, horizontal scaling can be used to distribute the load among other servers.

Dedicated Data Stores

OpenSSO Enterprise requires two data stores. During installation, you must specify the location of each data store. For detailed information, see Chapter 4, "Configuring OpenSSO Enterprise Using the GUI Configurator," in *Sun OpenSSO Enterprise 8.0 Installation and Configuration Guide*

Configuration Data Store

The configuration data store contains information about how users are authenticated, which resources users can access, and what information is available to applications after users are given access to resources. You can use the OpenSSO Enterprise configuration store that is automatically embedded in each OpenSSO Enterprise. Or you can use the Sun Directory Server configuration data store.

User Data Store

During OpenSSO Enterprise installation, you must specify which user data store you want to use.

OpenSSO Enterprise User Data Store	Use this option when you want to store user data in the OpenSSO Enterprise user data store.
Other User Data Store	Use this option when you want to store user data in a data store such as Sun Java System Directory Server.

OpenSSO Enterprise uses an identity repository to store user data such as users and groups. You can use Sun Directory Server or a supported LDAPv3 compliant directory server as the identity repository. Use the tables in this section to help you determine which user data store meets your needs.

In the following table, a Policy Subject refers to the "who" part of the policy definition. The Policy Subject specifies the members or entities to which the policy applies. Policy Condition refers to the additional restrictions with which the policy applies. Examples are a specified window of time in a day, a specified IP address, or a specified authentication method.

TABLE 2–1 Supported Features for Various Directory Servers

OpenSSO Enterprise Feature	Sun Directory Server LDAPv3	Microsoft Active Directory LDAPv3	IBM Tivoli Directory	Generic LDAPv3
User Data Storage	Yes	Yes	Yes	No
Configuration Data Storage	Yes	No	No	No
AMSDK (legacy)	Yes	No	No	No
LDAP Authentication	Yes	Yes	Yes	Yes
Membership Authentication	Yes	No	No	No
AD Authentication	Not Applicable	Yes, with limitations	Not Applicable	Not Applicable
Policy Subjects and Policy LDAP Filter Condition	Yes	Yes	Yes	Yes
Password Reset	Yes (with Access Manager SDK only)	No	No	No
Account Lockout	Yes	No	No	No
Cert Authentication	Yes	Yes	Yes	Yes
MSISDN Authentication	Yes	Yes	Yes	Yes
Data Store Authentication (through LDAPv3 user store configuration)	Yes	Yes	Yes	Yes
User creation with Password and Password Management	Yes	No	Yes	Yes

The following table summarizes the user management operations supported through the IDRepo interface for various user data stores. An interface has been implemented specifically for Sun Directory Server and Microsoft Active Directory. The default implementation of this interface can be used and supported for any LDAPv3 user repository.

TABLE 2–2 Data Stores and Supported Operations

Feature	Sun Directory Server LDAPv3	Microsoft Active Directory LDAPv3	IBM Tivoli Directory	Generic LDAPv3	AMSDK (Legacy)
Create User	Yes	Yes*	Yes	No	Yes
Modify User	Yes	Yes*	Yes	No	Yes
Delete User	Yes	Yes*	Yes	No	Yes
Create Role	Yes	No	No	No	Yes
Modify Role	Yes	No	No	No	Yes
Delete Role	Yes	No	No	No	Yes
Assign Role	Yes	No	No	No	Yes
Evaluate Role for Membership	Yes	No	No	No	Yes
Create Group	Yes	Yes*	Yes**	No	Yes
Modify Group	Yes	Yes*	Yes**	No	Yes
Delete Group	Yes	Yes*	Yes**	No	Yes
Evaluate Group for Membership	Yes	Yes*	Yes**	No	Yes
Create Agent	Yes	No	No	No	No
Delete Agent	Yes	No	No	No	No
Modify Agent	Yes	No	No	No	No
Federation Attributes	Yes	Yes	Yes	No	Yes

*Some limitations exist, or additional configuration is required.

** See limitations in the next section “Additional Information About Using IBM Tivoli Directory Server.”

Additional Information About Using IBM Tivoli Directory Server Configured as the IDRepo Data Store

IBM Tivoli Directory Server's groups can be Static, Dynamic, and Nested. However, the OpenSSO Enterprise IDRepo framework (IDRepo DataStore) supports only the Static group. A Static group defines each member individually using either of the the following:

- Structural ObjectClass: `groupofNames`, `groupOfUniqueNames`, `accessGroup`, or `accessRole`
- Auxilary ObjectClass: `ibm-staticgroup` or `ibm-globalAdminGroup`

A Static group using the Structural ObjectClass groupOfNames and groupOfUniqueNames requires at least one member for ObjectClass groupOfNames or one uniquemember for groupOfUniqueNames. The Static group using the ObjectClass ibm-staticgroup does not have this requirement. The ObjectClass ibm-staticgroup is the only ObjectClass for which members are optional; all other object classes require at least one member.

OpenSSO Enterprise supports only one ObjectClass for groups. If you choose a type of group with an ObjectClass that requires at leas one member, then a user value must be present. This user will automatically be added to the group when a group is created. You can remove this user from the group afterward if you don't want this user to be a member of the group.

The value for the filter for searching of groups must the value specified by the chosen LDAP Group ObjectClass.

Most IBM Tivoli groups require at least one member when the group is created. When a group is created using the OpenSSO Enterprise console, no users are assigned to the group by default. Since IBM Tivoli has this restriction, when a group is created, the default user or member cn=auser1,dc=opensso,dc=java,dc=net is always automatically created and added to the group.

Additional Information for Determining Which User Data Store to Use

- Account Lockout locks a user account based on the policies defined in the Directory Server.

  For example, the user account can be locked when a specified number of login failures occurs.
- The key difference between using a policy LDAP subject and the IDRepo interface subject is that policy LDAP subjects don't provide caching and notification updates. The AMIdentity Subject does provide caching an notification updates.

  The policy LDAP subjects provide LDAP Organization, Role (if Sun Directory Server), Group, and User subjects to evaluate membership of a user and determine if the user belongs to one of these subjects. The same result can be obtained using the Identity Repository (IDRepo) interface subject named AMIdentity Subject. This interface subject was introduced when the product was named Access Manager 7.0. You can develop a policy subject for a JDBC user store. Authentication also supports the JDBC repository through the JDBC authentication module.
- The IDRepo interface provides basic user management features for user, group, role, and Access Manager policy agent entities.

  This interface enables OpenSSO Enterprise to support any user repository through the development of new plug-ins. Although limited to Sun Directory Server, Microsoft Active Directory, and IBM Tivoli Directory today, the IDRepo interface could potentially be expanded to include any LDAPv3 directory server such as OpenLDAP or Novel Directory for JDBC, flat files, and so forth.

- Prior to Access Manager 7.0, user management was supported using Access Manager object classes and attributes in addition to using specific features from Sun Directory Server. This support still exists through the legacy AMSDK interface. But this support is deprecated and will be removed future releases.

Notification Support for the User Data Store

The data change in the directory server must be propagated to OpenSSO Enterprise in a timely manner to ensure that OpenSSO Enterprise represents the correct data. The data in OpenSSO Enterprise is updated two ways. One way is by receiving notifications from the directory servers, and the other way is by polling the directory servers. For notification, directory servers typically provide persistent search notifications which OpenSSO Enterprise subscribes to. For polling, OpenSSO Enterprise provides configurable parameters to specify the intervals. OpenSSO Enterprise supports persistent search notifications with Sun Directory Server, Microsoft Active Directory, and IBM Tivoli Directory.

Examining a Single Sign-On Deployment Example

The most basic OpenSSO Enterprise deployment is designed to achieve single sign-on in a secure, highly-available, and scalable environment that includes dedicated configuration and user data stores. Keep these goals in mind as you build your deployment architecture.

Use the following deployment example to get a sense of how you can map your enterprise requirements to a deployment architecture.

Identifying the Major Components

The following figure illustrates the most basic deployment architecture for OpenSSO Enterprise in single sign-on environment. A list of the components that comprise the architecture follows.

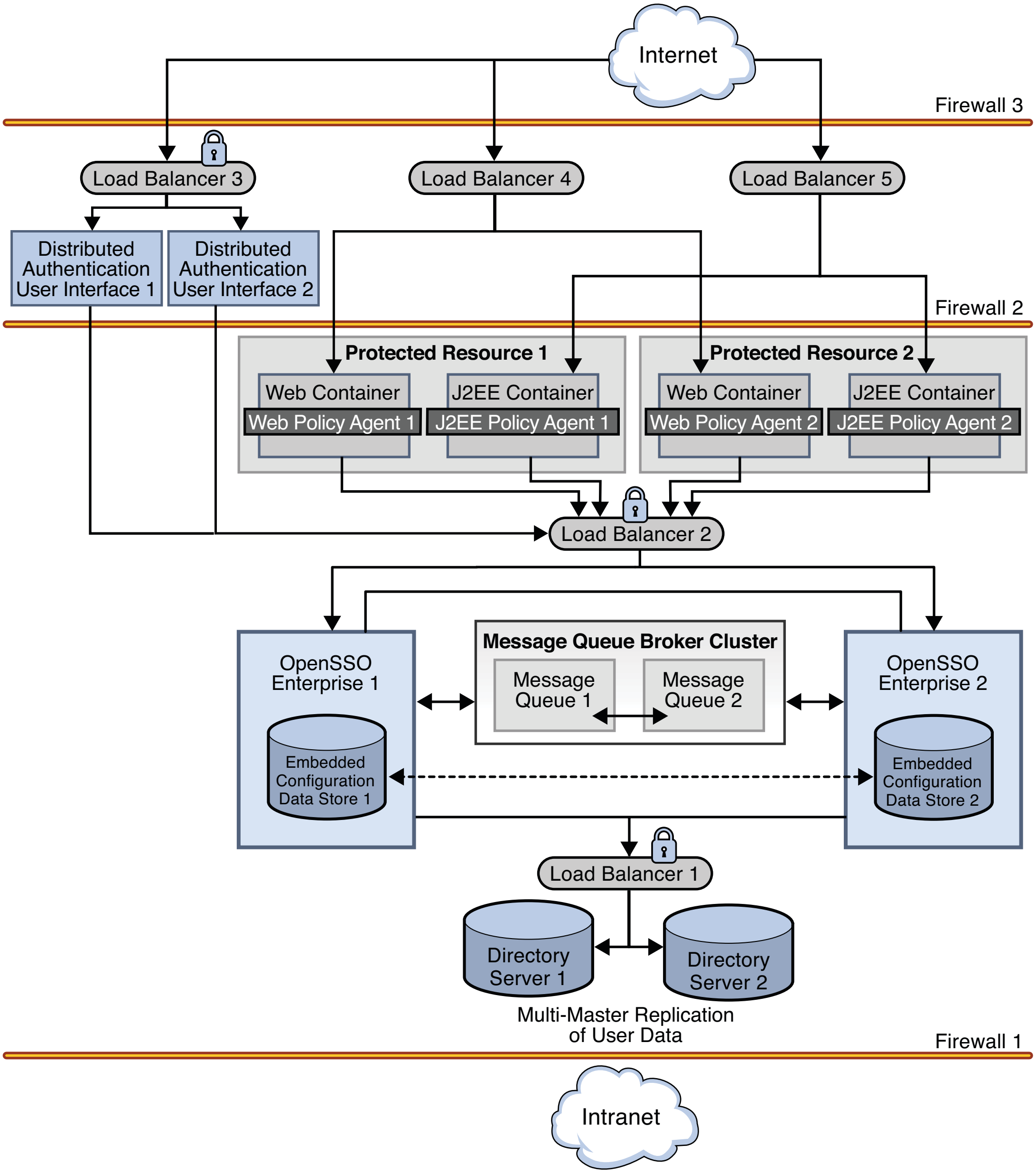


FIGURE 2–1 Single Sign-On Deployment Architecture Example

Sun OpenSSO Enterprise

Two instances of OpenSSO Enterprise provide the core functionality. Each instance is configured with its own embedded configuration data store. Configuration data includes information about services, administrative users, realms, policies, and more. User data is accessed through a single load balancer deployed in front of two instances of Sun Java System Directory Server.

Distributed Authentication User Interface

The Distributed Authentication User Interface is a component of OpenSSO Enterprise that provides a thin presentation layer for user authentication. During user authentication, the Distributed Authentication User Interface interacts with OpenSSO Enterprise to retrieve credentials from the user data store, thus protecting the OpenSSO Enterprise servers from direct user access. The Distributed Authentication User Interface does not directly interact with the user data store.

Sun Java System Directory Server

Two instances of Directory Server provide storage for the OpenSSO Enterprise user data. Both instances of Directory Server are masters that engage in multi-master replication. Multi-master replication allows data to be synchronized in real time between two directories, providing high availability to the OpenSSO Enterprise layer.

Sun OpenSSO Enterprise Policy Agents 3.0

Policy agents are used to restrict access to hosted content or applications. The policy agents intercept HTTP requests from external users and redirect the request to OpenSSO Enterprise for authentication. Web policy agents protect any resources under the `doc` root of the web container. J2EE policy agents protect a variety of hosted J2EE applications; in this deployment, `agentsample` is used. The agents communicate with the OpenSSO Enterprise instances through one of two configured load balancers.

Protected Resource Host Machines

The protected resources host machines contain the content for which access is restricted. Towards this end, web servers, application servers and policy agents will be installed. Two load balancers are configured in front of the host machines to balance traffic passing through the policy agents.

Sun Java System Message Queue

OpenSSO Enterprise uses two instances of Message Queue to form a cluster for distributing client connections and delivering messages. The Berkeley Database by Sleepycat Software, Inc. is the session store database. When an instance of OpenSSO Enterprise goes down and session failover is enabled, the user's session token can be retrieved from one of the Message Queues by the available instance of OpenSSO Enterprise. This ensures that the user remains continuously authenticated, allowing access to the protected resources without having to reauthenticate.

Load Balancers

The load balancer hardware and software used for this deployment is BIG-IP® manufactured by F5 Networks. They are configured for *simple persistence* and deployed as follows:

Distributed Authentication User Interface Load Balancer. This external-facing load balancer exposes the remote, web-based Distributed Authentication User Interface for user authentication and self-registration.

OpenSSO Enterprise Load Balancer. This internal-facing load balancer exposes the web-based OpenSSO Enterprise console to internal administrators. Alternatively, internal administrators can bypass this load balancer and log in directly.

J2EE Policy Agents Load Balancer. The load balancer in front of the J2EE policy agents installed on the Protected Resource machines provides round-robin load balancing and a single virtual server by balancing traffic passing through the agents.

Web Policy Agents Load Balancer. The load balancer in front of the web policy agents installed on the Protected Resource machines provides round-robin load balancing and a single virtual server by balancing traffic passing through the agents.

Directory Server Load Balancer. The load balancer in front of the Directory Server instances provide round-robin load balancing and a single virtual Directory Server host name for the instances of OpenSSO Enterprise. It detects individual Directory Server failures and recoveries, taking failed servers off the load balancer list.

For detailed instructions on how to deploy these components, see *Deployment Example: Single Sign-On, Load Balancing and Failover Using Sun OpenSSO Enterprise 8.0*.

Designing the Single Sign-On Deployment Architecture

Once you've identified the major components you need in your environment, you can build your deployment architecture to map to your enterprise needs. In this deployment example, the deployment architecture is designed to meet the goals of the most basic OpenSSO Enterprise single sign-on environment:

- All components (including installations of OpenSSO Enterprise and Directory Server, the Distributed Authentication User Interface, and policy agents) are redundant to achieve high availability.
- All components use load-balancing for session failover and high performance.
- Each instance of OpenSSO Enterprise is installed with an embedded configuration data store.
- Each instance of Directory Server contains `am-users` to serve as the user data store.
- The environment is configured for system failover capability, ensuring that when one instance of OpenSSO Enterprise goes down, requests are redirected to the second instance.
- The environment is configured for session failover capability. Session failover ensures that when the instance of OpenSSO Enterprise *where the user's session was created* goes down, the user's session token can still be retrieved from a backend session database. Thus, the user is continuously authenticated, and does not have to log into the system again unless the session is invalidated as a result of logout or session expiration.

- Communications to the OpenSSO Enterprise load balancer, to the Distributed Authentication User Interface load balancer, and to the Directory Server load balancer are in Secure Sockets Layer (SSL).
- Policy agents are configured with a unique agent profile to authenticate to OpenSSO Enterprise.
- The Distributed Authentication User Interface uses a custom user profile to authenticate to OpenSSO Enterprise instead of the default `amadmin` or `UrlAccessAgent`.

Examining a SAMLv2 Identity Federation Deployment Example

In a deployment configured for communication using SAMLv2, a Service Provider and an Identity Provider must be created within a *circle of trust*. The circle of trust enables business providers to easily conduct cross-network transactions for an individual while protecting the individual's identity.

Use the following deployment example to get a sense of how you can map your enterprise requirements to a SAMLv2 Identity Federation deployment architecture.

Identifying the Major Components

- "Identity Provider Deployment" on page 43
- "Service Provider Deployment" on page 45

Identity Provider Deployment

An identity provider specializes in providing authentication services. As the administrating service for authentication, an identity provider maintains and manages identity information. It establishes trust with a service provider in order to exchange user credentials, enabling single sign-on between the providers. Authentication by an identity provider is honored by all service providers with whom the identity provider is partnered. The following image illustrates the identity provider architecture in this deployment.

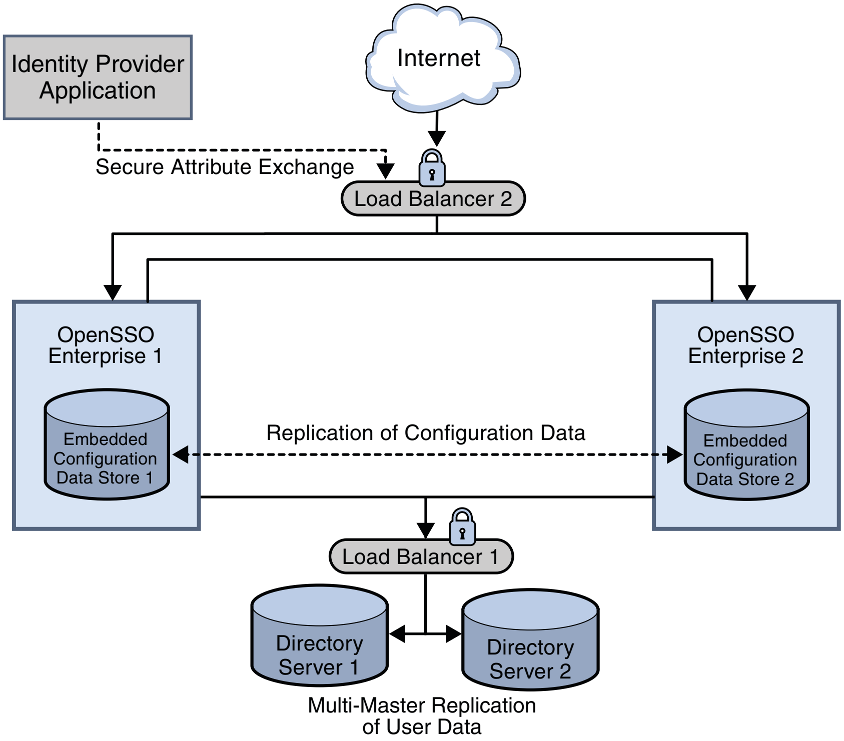

Identity Provider Application
Internet
Secure Attribute Exchange
Load Balancer 2
OpenSSO Enterprise 1
Embedded Configuration Data Store 1
Replication of Configuration Data
OpenSSO Enterprise 2
Embedded Configuration Data Store 2
Load Balancer 1
Directory Server 1
Directory Server 2
Multi-Master Replication of User Data

FIGURE 2–2 Identity Provider Deployment Architecture

The identity provider domain in this deployment is `idp-example.com`. The identity provider application represents a legacy system which relies on OpenSSO Enterprise to act as a secure gateway through which identity information can be transferred to another application in a different domain. This functionality is provided by the Secure Attribute Exchange feature of OpenSSO Enterprise which uses SAMLv2 without having to deal with federation protocol and processing.

The following list of components will be installed and configured in the Identity Provider environment.

Sun OpenSSO Enterprise

Two instances of OpenSSO Enterprise provide the core functionality. Each instance is created with a configuration data store. Configuration data includes information about services, administrative users, realms, policies, and more. Two instances of Sun Java System Application Server are installed on the OpenSSO Enterprise host machines into which the OpenSSO Enterprise WAR is then deployed.

User data is accessed through a single load balancer deployed in front of two instances of Sun Java System Directory Server.

Sun Java System Directory Server

Two instances of Directory Server provide storage for user entries that will be created for testing this deployment. Both instances of Directory Server are masters that engage in multi-master replication, providing high availability to the OpenSSO Enterprise layer.

Load Balancers

The load balancer hardware and software used for this deployment is BIG-IP® manufactured by F5 Networks. They are configured for *simple persistence* and deployed as follows:

- OpenSSO Enterprise Load Balancer.

  This load balancer exposes the web-based OpenSSO Enterprise console to internal administrators. Alternatively, internal administrators can bypass this load balancer and log in directly.

- Directory Server Load Balancer.

  The load balancer in front of the Directory Server instances provide round-robin load balancing and a single virtual Directory Server host name. It detects individual Directory Server failures and recoveries, taking failed servers off the load balancer list.

Service Provider Deployment

A service provider offers web-based services to an identity. This broad category can include portals, retailers, transportation providers, financial institutions, entertainment companies, libraries, universities, governmental agencies, and other organizations that consume identity information for purposes of access. The following figure illustrates the Service Provider architecture in this deployment.

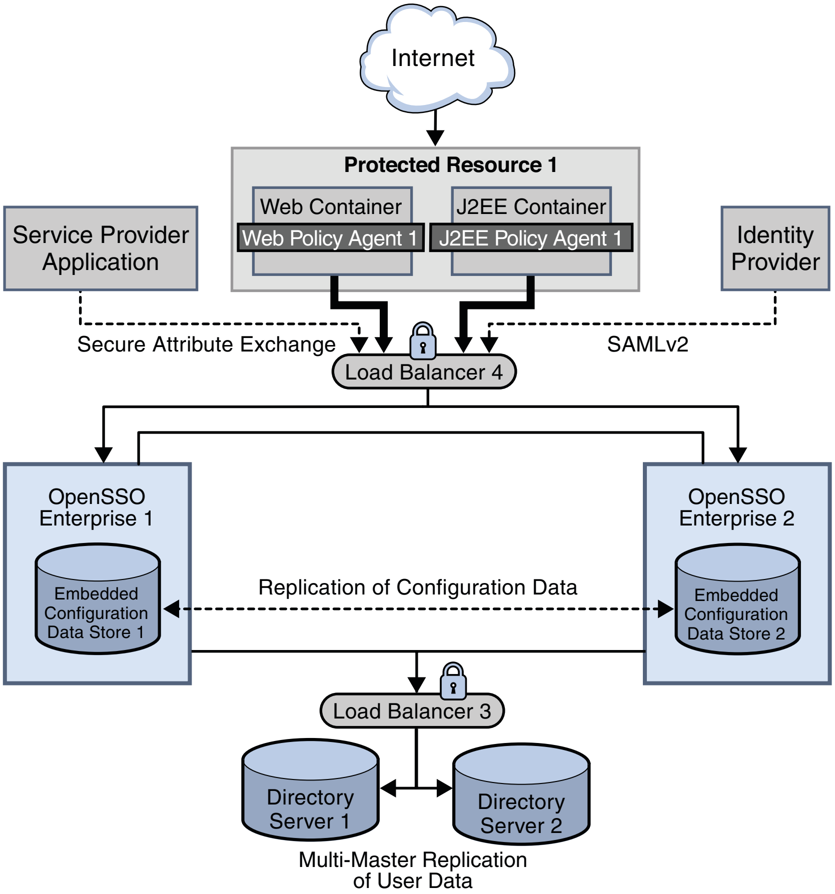


FIGURE 2–3 Service Provider Deployment Architecture

The service provider domain in this deployment is `sp-example.com`. The service provider application represents a legacy system which relies on OpenSSO Enterprise to act as a secure gateway through which identity information can be received from the identity provider. This functionality is provided by the Secure Attribute Exchange feature of OpenSSO Enterprise which uses SAMLv2 without having to deal with federation protocol and processing.

The following list of components will be installed and configured using the procedures documented in *Deployment Example: SAML v2 Using Sun OpenSSO Enterprise 8.0*.

Sun OpenSSO Enterprise

Two instances of OpenSSO Enterprise provide the core functionality. Each instance is created with a configuration data store. Configuration data includes information about services, administrative users, realms, policies, and more.

Sun Java System Directory Server

Two instances of Directory Server provide storage for user entries that will be created for testing this deployment. User data is accessed through a single load balancer deployed in front of two instances of Sun Java System Directory Server. Both instances of Directory Server are masters that engage in multi-master replication, providing high availability to the OpenSSO Enterprise layer.

Sun Java System Application Server

Two instances of Sun Java System Application Server are installed on the OpenSSO Enterprise host machines into which the OpenSSO Enterprise WAR is then deployed.

Load Balancers

The load balancer hardware and software used for this deployment is BIG-IP® manufactured by F5 Networks. They are deployed as follows:

- OpenSSO Enterprise Load Balancer

  This load balancer exposes the web-based OpenSSO Enterprise console to internal administrators. Alternatively, internal administrators can bypass this load balancer and log in directly.

- Directory Server Load Balancer

  The load balancer in front of the Directory Server instances provides round-robin load balancing and a single virtual Directory Server host name. It detects individual Directory Server failures and recoveries, taking failed servers off the load balancer list.

Sun OpenSSO Enterprise Policy Agents

Policy agents are used to restrict access to hosted content or applications. The policy agents intercept HTTP requests from external users and redirect the request to OpenSSO Enterprise for authentication. Web policy agents protect any resources under the `doc` root of the web container. J2EE policy agents protect a variety of hosted J2EE applications; in this deployment, `agentsample` is used. The agents communicate with the OpenSSO Enterprise instances through the configured load balancer.

Protected Resource Host Machine

The protected resource host machine contains the content for which access is restricted. BEA WebLogic Server and a J2EE policy agent is installed. Sun Java System Web Server a web policy agent is also installed. Additionally, a sample JSP is installed to act as the legacy application for purposes of demonstrating the Secure Attribute Exchange feature.

For step-by-step instructions for deploying these components as illustrated in this chapter, see *Deployment Example: SAML v2 Using Sun OpenSSO Enterprise 8.0*.

Designing the SAMLv2 Identity Federation Architecture

Once you've identified the major components you need in the Service Provider and Identity Provider environments, you can build your deployment architecture to map to your enterprise needs. In this deployment example, the deployment architecture is designed to achieve the most basic OpenSSO Enterprise circle of trust. The architecture is designed to meet the following enterprise requirements:

- All instances of OpenSSO Enterprise are deployed behind a load balancer for high-availability.
- Instances of OpenSSO Enterprise acting as an identity provider are configured to work with instances of Sun Directory Server configured as the user data store.
- XML Signing is enabled for all SAMLv2 protocols.
- The SAMLv2 URL end points are exposed through load balancers with SSL termination and regeneration configuration.
- A web policy agent and a J2EE policy agent are deployed in front of the service provider instances of OpenSSO Enterprise; the policy agents work in single sign-on mode only.

Designing the Deployment Architecture

Once you've designed your basic deployment architecture, then you can determine which additional OpenSSO Enterprise features you want to deploy. The chapters in Part II of this manual are designed to help you determine which OpenSSO Enterprise features are suitable for your enterprise. Each chapter in Part II contains the following:

- Brief overview of an OpenSSO Enterprise feature
- Diagram depicting the feature's role in a deployment example
- Diagram illustrating how the feature works
- High-level configuration requirements
- Dependencies, constraints, benefits and tradeoffs you should consider
- Typical business use cases

Once you've developed a deployment architecture that includes the OpenSSO Enterprise features you need, you can proceed to develop your detailed implementation plan.

CHAPTER 3

Building the Implementation Plan

A deployment architecture identifies the software components needed to implement a secure, scalable, high-availability enterprise. The main purpose of a deployment architecture is to illustrate the interrelationships among the major components in the network environment. This *Deployment Planning Guide* provides information to help you evaluate OpenSSO Enterprise solutions, and to build a deployment architecture.

An implementation plan describes both hardware and software components needed to meet specific quality of service requirements. For example, in your network needs assessment, you may have identified quality of service requirements based on a number of factors such as:

- Number and types of users in your enterprise
- Levels of access accorded to various user types
- Geographic locations of your users
- Usage levels and usage trends

Building an implementation plan to address such issues is beyond the scope of this *Deployment Planning Guide*. To build your detailed implementation plan, you need additional system sizing information and specific low-level implementation guidance. Contact your Sun Sales engineering representative for more information.

Contacting Sun

- Sun Sales Offices

  Contact a Sun sales office in your area.

  `http://www.sun.com/contact/office_locations.jsp`
- Sun Services and Solutions

  Find out about consulting, IT services, and Sun's complete technology solutions.

  `http://www.sun.com/contact/services_solutions.jsp`
- Sun Support Services

Get quick answers to your system support, management and service contract questions.

http://www.sun.com/contact/support.jsp

PART II

Determining Which Features to Deploy

The chapters in Part II of this manual are designed to help you determine which OpenSSO Enterprise features are suitable for your enterprise. Each chapter contains a brief overview of an OpenSSO Enterprise feature, a deployment architecture diagram, a process flow diagram, and high-level configuration requirements. Each chapter also describes dependencies, constraints, benefits, tradeoffs, and typical business use cases to further help you plan various solutions to meet your business needs.

Part II of this book contains the following chapters.

- Chapter 14, "Enabling Single Sign-On Using Oracle Access Manager and OpenSSO Enterprise"
- Chapter 15, "Using the Embedded Configuration Data Store for OpenSSO Enterprise"

CHAPTER 4

Using a Policy Agent and the Client SDK to Integrate Applications with OpenSSO Enterprise

This chapter provides a quick overview of the various ways in which new and existing applications can be integrated with an existing FAM deployment for Authentication, Authorization, Auditing and Single Sign-On (AAA) services, Federation, Web Services, Web Services Security and Identity Services.

The following topics are contained in this chapter:

About the OpenSSO Enterprise Client SDK

The OpenSSO Enterprise Client SDK is the core software component that enables you to integrate OpenSSO Enterprise with other applications. The Client SDK is supplied by OpenSSO Enterprise and provides APIs you can use to access each service hosted by the OpenSSO Enterprise server. The following are common ways of using the Client SDK :

1. Embedded directly in the business logic of a standalone application.
2. Embedded directly in a container-hosted application such as a .Net or a J2EE application server.
3. Embedded in the container either directly or using a container-provided security plug-in mechanism.
4. Embedded in a proxy server installed in front of the protected application.

OpenSSO Enterprise Policy Agents are prepackaged client software that implement options 3 and 4 above.

About the Centralized Policy Agent Configuration

The Centralized Policy Agent Configuration is new in OpenSSO Enterprise 8.0. This feature provides a policy agent interface for managing multiple policy agent configurations from a single, centralized place. The policy agent configurations are stored in the OpenSSO Enterprise data store. A policy agent administrator can use either the OpenSSO Enterprise command-line interface (CLI), or the administration console to manage stored data.

Most policy agent configuration changes are conveyed to the participating policy agents without requiring the policy agents to be restarted. The policy agents respond to the changes based on the nature of the updated properties.

In the Centralized Policy Agent Configuration, policy agent configurations are separated into two sets. One set contains a few policy agent properties that are absolutely required for the policy agent start and to initialize itself properly. A file that contains these properties remains at the local host on which the policy agent is installed. This properties file acts as bootstrapping file for the policy agent.

The other set of policy agent configurations contains all remaining agent configuration properties. These configuration properties are stored either at the local policy agent host, or at a centralized data store managed by the OpenSSO Enterprisebased on the agent configuration repository type.

You can configure OpenSSO Enterprise to store policy agent configurations in a local repository or in a remote repository. A local policy agent configuration repository is a property file that contains all the policy agent configuration data. This option is supported for backward compatibility with legacy deployments. A remote policy agent configuration repository is the newer, more efficient option. When the policy agent configuration is stored in a remote, centralized data store managed by the OpenSSO Enterprise server, during server startup, the policy agent reads the bootstrapping file first to initialize itself. Then the policy agent makes an attribute service request to the OpenSSO Enterprise server to retrieve the policy agent configuration. The policy agent configuration returned by the OpenSSO Enterprise server contains a property that determines the location of the policy agent configuration.

If the property value is centralized, the policy agent uses the configuration just returned. If the property value is local, then the policy agent retrieves the remaining configuration properties from the local policy agent configuration repository and performs its functions accordingly.

The policy agent configuration must be totally stored in either a remote repository or a local repository. Mixed configurations are not supported.

Analyzing the Deployment

The following figure illustrates the deployment architecture for standards-based federated single sign-on using federated web services among independent, trusted partners. The OpenSSO Enterprise Client SDK pictured here includes advanced APIs to enable applications to directly invoke federation features. These APIs are available in the OpenSSO Enterprise Fedlet, a streamlined federation tool. For more information on the OpenSSO Enterprise Fedlet, see Chapter 5, "Using the OpenSSO Enterprise Fedlet to Enable Identity Federation."

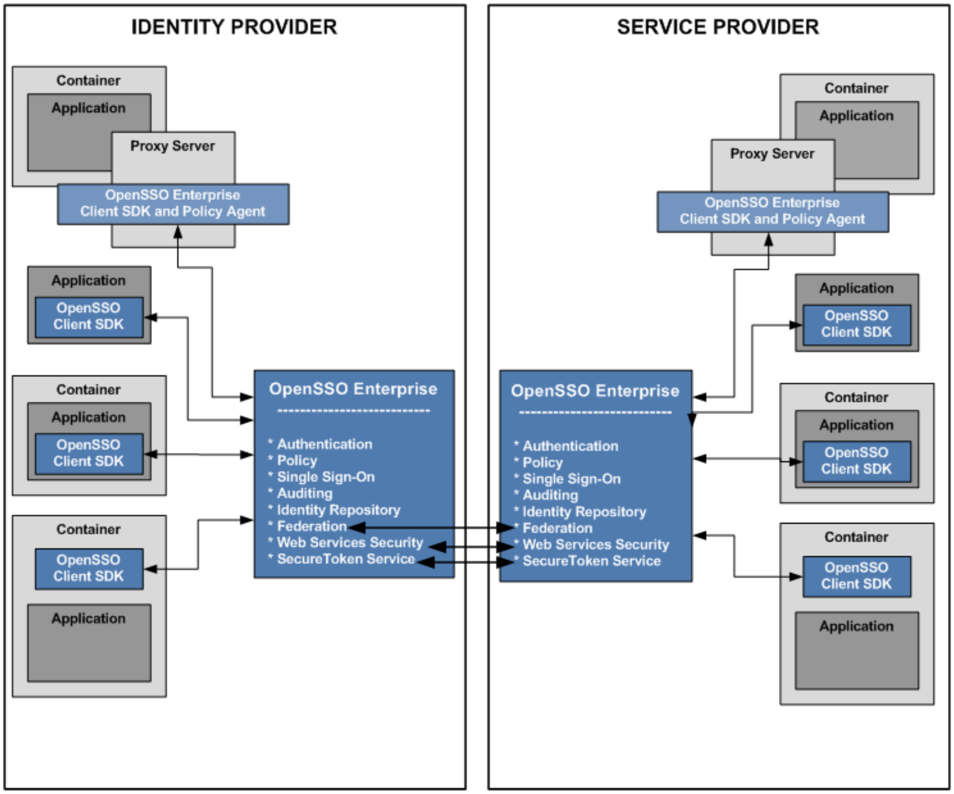


FIGURE 4–1 Deployment Architecture for the Client SDK in Federated Single Sign-On

The following figures illustrate the process flow among the OpenSSO Enterprise server, the OpenSSO Enterprise policy agent, and the Centralized Policy Agent Configuration components.

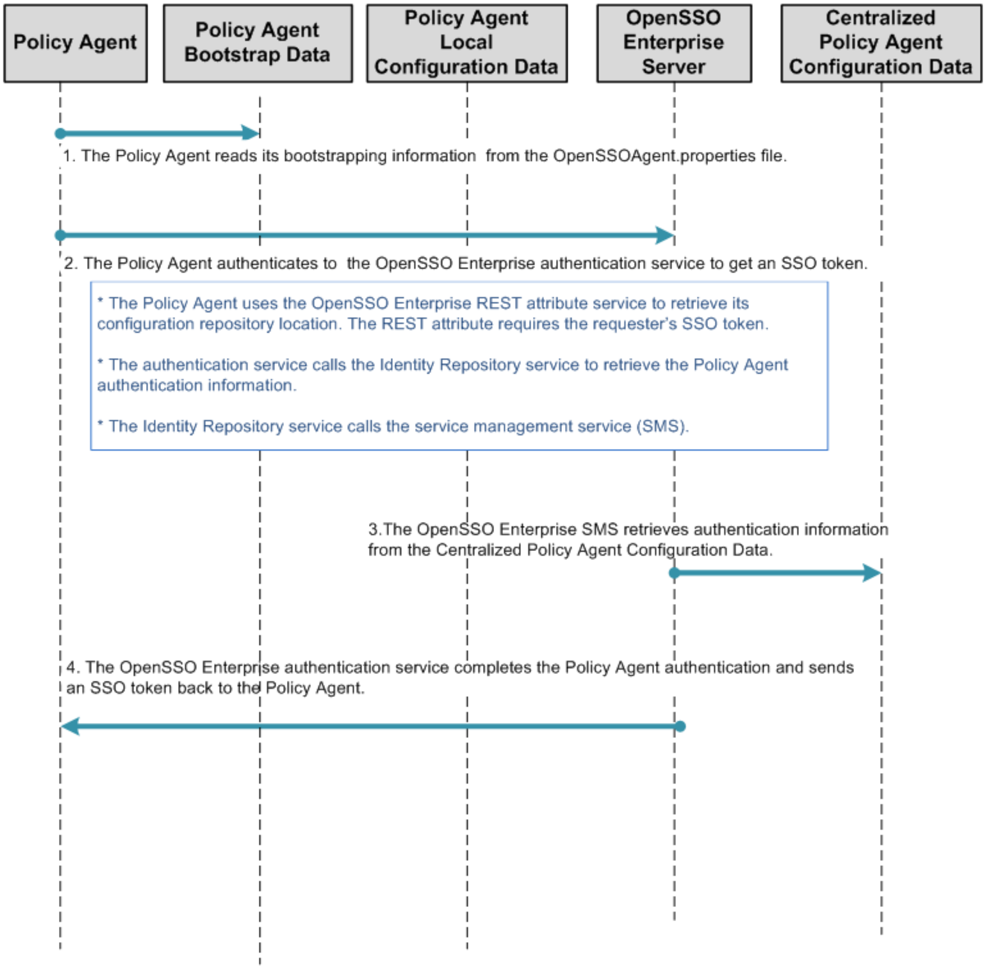
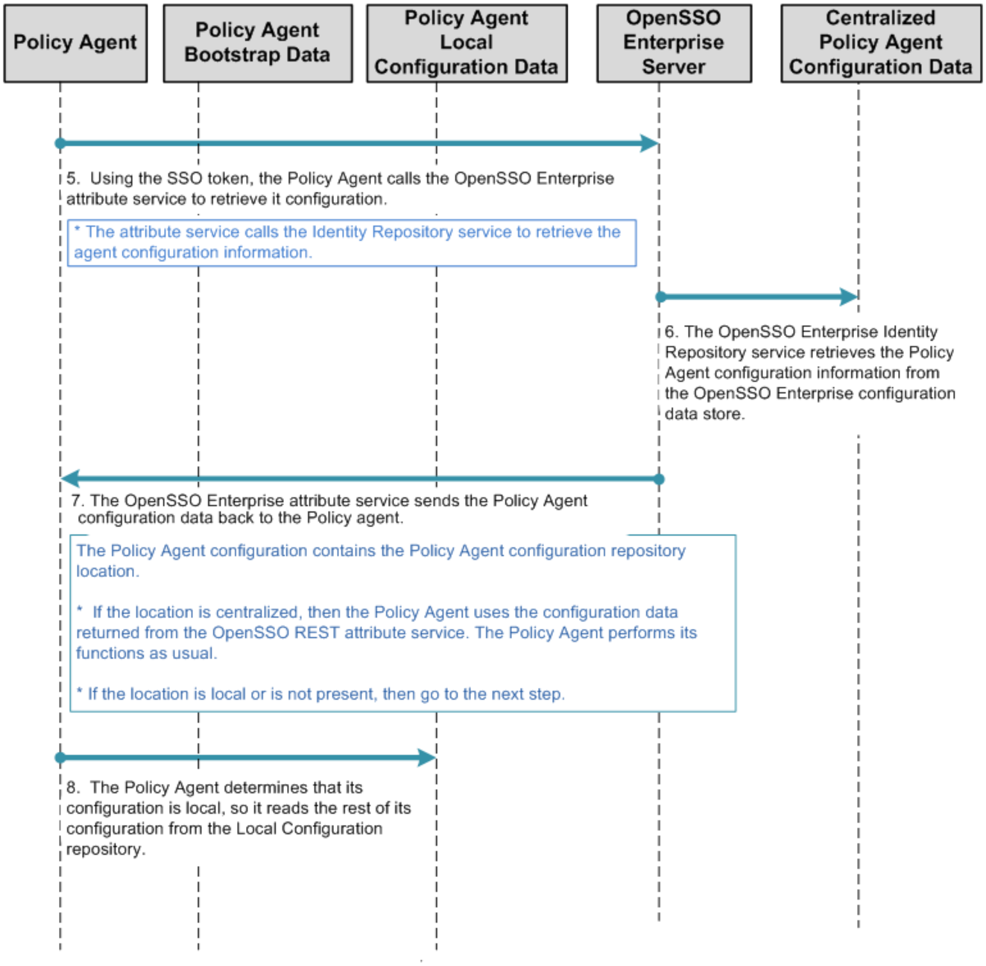

Policy Agent
Policy Agent Bootstrap Data
Policy Agent Local Configuration Data
OpenSSO Enterprise Server
Centralized Policy Agent Configuration Data
1. The Policy Agent reads its bootstrapping information from the OpenSSOAgent.properties file.
2. The Policy Agent authenticates to the OpenSSO Enterprise authentication service to get an SSO token.
* The Policy Agent uses the OpenSSO Enterprise REST attribute service to retrieve its configuration repository location. The REST attribute requires the requester's SSO token.
* The authentication service calls the Identity Repository service to retrieve the Policy Agent authentication information.
* The Identity Repository service calls the service management service (SMS).
3.The OpenSSO Enterprise SMS retrieves authentication information from the Centralized Policy Agent Configuration Data.
4. The OpenSSO Enterprise authentication service completes the Policy Agent authentication and sends an SSO token back to the Policy Agent.

FIGURE 4–2 Process flow for Centralized Policy Agent Configuration


Policy Agent
Policy Agent Bootstrap Data
Policy Agent Local Configuration Data
OpenSSO Enterprise Server
Centralized Policy Agent Configuration Data

5. Using the SSO token, the Policy Agent calls the OpenSSO Enterprise attribute service to retrieve it configuration.

* The attribute service calls the Identity Repository service to retrieve the agent configuration information.

6. The OpenSSO Enterprise Identity Repository service retrieves the Policy Agent configuration information from the OpenSSO Enterprise configuration data store.

7. The OpenSSO Enterprise attribute service sends the Policy Agent configuration data back to the Policy agent.

The Policy Agent configuration contains the Policy Agent configuration repository location.

* If the location is centralized, then the Policy Agent uses the configuration data returned from the OpenSSO REST attribute service. The Policy Agent performs its functions as usual.

* If the location is local or is not present, then go to the next step.

8. The Policy Agent determines that its configuration is local, so it reads the rest of its configuration from the Local Configuration repository.

FIGURE 4–3 Process flow for Centralized Policy Agent Configuration (continued)

Considering Assumptions, Dependencies, and Constraints

You must use the OpenSSO Enterprise administration console or the OpenSSO Enterprise command—line tools to create, delete, and manage the policy agent configurations.

Understanding Typical Business Use Cases

The following are typical use cases for using Policy Agents and the Client SDK to integrate OpenSSO Enterprise with applications in an existing deployment:

- "Using Non-Intrusive, Policy Agent-Based Approaches to Web Resources" on page 58
- "Leveraging Fat Clients, Custom Web Applications, and Enterprise JavaBeans" on page 59
- "Complementing Policy Agent Functionality" on page 59
- "Enabling Identity Federation" on page 59
- "Enabling Web Services Security" on page 60
- "Enabling Identity Services" on page 60
- "Co-Existing with Non-Sun Deployments" on page 60

Using Non-Intrusive, Policy Agent-Based Approaches to Web Resources

The following are examples of capabilities that can be leveraged by using the Policy Agents to integrate OpenSSO Enterprise with other applications:

- Pure J2EE applications

  Pure J2EE applications are deployed as WARs installed on J2EE compliant application servers. The resources to be protected include servlets, JavaServer Pages and Enterprise JavaBeans.
- Apache or Sun Web Server-based web applications

  These applications can be HTML pages, multimedia content, and CGIs such as PHP, Perl, JSP, and servlets hosted on the web server.
- .Net /IIS server-based web applications

  These include .Net/ASP or ASPX applications such as Visual Basic and C#, HTML pages and content accessible via the HTTP protocol.
- Enterprise applications

  These include SAP, Siebel, Domino, PeopleSoft, and Portal middleware.
- Proxied applications

  This use case can include all web applications deployed with a reverse proxy server in front of it. No policy agents or added software are required be deployed on the application and its container. A proxy is installed separately from the application, and the policy agent installation also stays separate. Multiple applications can be proxied by the same proxy server, enabling a single agent to protect them all.

Leveraging Fat Clients, Custom Web Applications, and Enterprise JavaBeans

The following are examples of capabilities that can be leveraged by using the Client SDK to integrate OpenSSO Enterprise with other applications:

- Java or C /C++

  Applications can be enabled to directly invoke the Client SDK for authentication, authorization, and auditing services.
- Password Replay

  This use case is for legacy applications that require the user to submit credentials directly to the application before access to services is allowed.
- Application-initiated Authentication

  In this use case, the authentication authority is the application itself. Once the user is authenticated, the remaining security services such as single sign-on, policy evaluation, and identity federation are provided by OpenSSO Enterprise.

Complementing Policy Agent Functionality

In this use case, an OpenSSO Enterprise policy agent is deployed and the Client SDK is embedded in the application or its container. The following are ways in which this configuration helps to complement policy agent functionality:

- Custom service-based access control is automatically in place.
- Policy agents can handle only URL—based policy. The Client SDK can handle various non-URL based policies.
- The Client SDK provides more finely-grained security control that is not possible with policy agents alone.

Enabling Identity Federation

Companies integrate applications with OpenSSO Enterprise to implement identity federation in various ways.

- OpenSSO Enterprise passes attributes from an Identity Provider application to a Service Provider. In this use case, the Identity Provider passes user attribute value pairs to the Service Provider so the Service Provider can provide services to the user based on those attributes.

- OpenSSO Enterprise receives Identity Provider-asserted attributes in a Service Provider application. In this use case the Service Provider verifies the authenticity of the attributes asserted by the Identity Provider. The Service Provider then updates its session with those attributes.
- The OpenSSO Enterprise Fedlet quickly enables federation without having to install a full-featured OpenSSO Enterprise sever at the Service Provider. In this use case, the Service Provider can participate in Federation with an Identity Provider that does have the full-featured OpenSSO Enterprise server installed on it.

Enabling Web Services Security

Companies integrate applications with OpenSSO Enterprise to take advantage of the following OpenSSO Enterprise Web Services Security measures:

- Protecting a web services provider endpoint
- Protecting a web services client invocation

Enabling Identity Services

Applications integrated with OpenSSO Enterprise are able to consume simple OpenSSO Enterprise services in both the SOAP/WSDL style and the REST style. OpenSSO Enterprise may include one or more of the following:

- Authentication
- Authorization of an authenticated identity to access a resource
- Retrieval of an authenticated identity's attributes
- Logging

Co-Existing with Non-Sun Deployments

Companies typically integrate OpenSSO Enterprise with non-Sun identity services for two purposes:

- To maintain backward compatibility with legacy applications in an existing environment

  In this use case, legacy non-Sun applications for authentication, authorization, and auditing have already been deployed and will continue to be used in the environment. OpenSSO Enterprise is used primarily to execute federation among both legacy and future applications. OpenSSO Enterprise may also be used for identity services when new applications are added to the environment.
- To facilitate complete migration from the non-Sun identity services to OpenSSO Enterprise

In this use case, legacy non-Sun applications for authentication, authorization. and auditing have already been deployed. These applications must be maintained until OpenSSO Enterprise can be deployed and tested. Once OpenSSO Enterprise is successfully deployed, the legacy non-Sun identity services are phased out of the environment.

Setting Up and Configuring the Integrated Environment

Before you can integrate other applications with OpenSSO Enterprise, you must resolve the following issues:

- "Deployment Planning " on page 61
- "Required Hardware and Software" on page 61
- "Downloading the Client SDK" on page 62
- "Downloading the OpenSSO Enterprise Policy Agent 3.0" on page 62

Deployment Planning

The following steps form a very general and high-level guide to determine what approach is best for you.

1. Determine if an OpenSSO Enterprise Policy Agent is available for the container or application you want to use.
2. Determine if the proxy OpenSSO Enterprise Policy Agent is usable in front of the application.
3. Determine if the application or container plug-ins that externalizes security (independent of the application business logic) are available and pluggable. Consider using the Client SDK to implement these plug-ins. This is how an OpenSSO Enterprise policy agent typically starts out.
4. Determine if a signed and encrypted query, post, or XML API is applicable.
5. Determine if you need to embed the Client SDK in your application or container. The obvious example is when "no" is the answer in the all of the four previous steps. You may still need to use this approach if certain functionality is not supplied by an available policy agent. An example is when you must use fine-grained, application-specific policies.
6. Consider using Server SPIs to customize the OpenSSO Enterprise server behavior to your needs.

Required Hardware and Software

The following software components are required to integrate OpenSSO Enterprise with other applications:

- Sun OpenSSO Enterprise 8.0

- Sun OpenSSO Enterprise 8.0 Client SDK
- Sun OpenSSO Enterprise Policy Agent 3.0

Some programming effort using the OpenSSO Enterprise Client SDK is required to implement the following business use cases:

- "Leveraging Fat Clients, Custom Web Applications, and Enterprise JavaBeans" on page 59
- "Complementing Policy Agent Functionality" on page 59
- "Enabling Identity Federation" on page 59
- "Enabling Web Services Security" on page 60

Installing and configuring an OpenSSO Enterprise Policy Agent is required to implement the following business use cases:

- "Using Non-Intrusive, Policy Agent-Based Approaches to Web Resources" on page 58
- "Enabling Identity Services" on page 60
- "Co-Existing with Non-Sun Deployments" on page 60

In OpenSSO Enterprise Policy Agent 3.0 the Centralized Agent Configuration feature enables centralized Policy Agent management. In earlier versions, the Policy Agent configuration is local to the server being protected.

Downloading the Client SDK

Download the OpenSSO Enterprise Client SDK from the following URL:

`https://opensso.dev.java.net/public/use/index.html`

The OpenSSO Enterprise Client SDK is part of the `opensso.zip` distribution, and is present in the `samples/opensso-client.zip` file within that distribution. See the README files inside the `opensso-client.zip` file for instructions on installing the OpenSSO Enterprise SDK. The OpenSSO Enterprise API Javadoc is available in the `docs/opensso-public-javadocs.jar` file.

Downloading the OpenSSO Enterprise Policy Agent 3.0

For download and installation information, go to the OpenSSO Enterprise Policy Agent 3.0 website at the following URL: (`http://wikis.sun.com/display/OpenSSO/PolicyAgents3`).

You will also find other useful articles about Policy Agent agent troubleshooting.

The OpenSSO Enterprise command-line inteface tool `ssoadmin` supports the following Policy Agent operations through it sub-commands:

- Create a Policy Agent configuration

- Delete a Policy Agent configuration
- Update a Policy Agent configuration
- List Policy Agent configurations
- Display a Policy Agent configuration
- Create a Policy Agent group
- Delete a Policy Agent group
- List agent groups
- List Policy Agent group members
- Add a Policy Agent to a group
- Remove a Policy Agent from a group

The OpenSSO Enterprise administration console supports all of the above operations. The table below summarizes the compatibility between the various versions of OpenSSO Enterprise and the OpenSSO Enterprise Policy Agent.

TABLE 4–1 OpenSSO Enterprise Server Compatibility with OpenSSO Enterprise Policy Agents

OpenSSO Enterprise	Policy Agent
OpenSSO Enterprise 8.0 (OpenSSO v1)	Policy Agent 3.0, 2.2
Access Manager 7.0, 7.1	Policy Agent 3.0, 2.2
Access Manager 6.3	Policy Agent 2.2

Evaluating Benefits and Tradeoffs

The following lists may help you determine whether using the Client SDK or using a policy agent is suitable in your environment:

- "Benefits of Using the Client SDK" on page 63
- "Tradeoffs Using the Client SDK" on page 64
- "Benefits of Using a Policy Agent" on page 64

Benefits of Using the Client SDK

- Using the Client SDK at the container or proxy server is a non-intrusive technique.
- Using the proxy-based approach is the least intrusive of all options presented in this chapter. The proxy-based technique does not require interaction with the application container or machine at all. It also has the added advantage of proxying multiple applications with the same proxy server.

Tradeoffs Using the Client SDK

- Embedding the Client SDK directly in a standalone application's business logic is an intrusive technique.
- Embedding the Client SDK directly in a container-hosted application is an intrusive technique.

Benefits of Using a Policy Agent

The Centralized Policy Agent Configuration moves most of the Policy Agent configuration to the OpenSSO Enterprise data repository. Using the Centralized Policy Agent Configuration results in the following benefits:

- Using Policy Agents is a less intrusive approach to application integration than embedding the OpenSSO Enterprise Client SDK in the application.
- Using the proxy-based approach the least intrusive of all options presented in this chapter. The proxy-based approach does not require interaction with the application container or host machine at all. It also has the added advantage of proxying multiple applications with the same proxy server.
- The Centralized Policy Agent Configuration supports all existing Policy Agent functionality including Policy Agent installation and uninstallation options. Using this feature allows separation between agent initialization data and agent configuration data.
- An agent administrator can manage multiple Policy Agent configurations from one central location, and can use either the OpenSSO Enterprise administration console or the command-line interface to do this.
- Any Policy Agent configuration changes are automatically conveyed to the affected agents, and the agents react to changes accordingly based on the nature of the updated properties. The administrator is not required to access the agent server to make this happen.
- Most of the Policy Agent configuration properties are hot-swappable. This means that when any Policy Agent configuration properties are changed in the centralized agent configuration, the affected agent will use the changed property values without having to restart itself. The Policy Agent makes calls to the OpenSSO Enterprise attribute service periodically to retrieve its configuration data.
- Centralized Policy Agent Configuration significantly reduces the time and resources spent on Policy Agent configuration management and Policy Agent patching.

Finding More Information

- Policy Agents 3.0

  http://wikis.sun.com/display/OpenSSO/PolicyAgents3
- OpenSSO Enterprise Wiki Home

  http://wikis.sun.com/display/OpenSSO/Home
- Download OpenSSO Software

  https://opensso.dev.java.net/public/use/index.html
- "Representational State Transfer" (REST)

  By Roy Fielding, who introduced the concept in 2000

  (http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm)

CHAPTER 5

Using the OpenSSO Enterprise Fedlet to Enable Identity Federation

The OpenSSO Enterprise Fedlet is a streamlined Service Provider implementation of SAMLv2 single sign-on (SSO) protocols. The OpenSSO Enterprise Fedlet is designed to be used by Service Providers when a full-featured federation solution is not required, and when the primary goals are to achieve single sign-on with an Identity Provider while also retrieving some user attributes from the Identity Provider.

The following topics are contained in this chapter:

About the OpenSSO Enterprise Fedlet

The OpenSSO Enterprise Fedlet is compliant with SAMLv2 standards so you can embed the Fedlet in a J2EE web application. You can embed the Fedlet SDK into a Service Provider application, and enable the application to accept SAML POSTs from an Identity Provider. The application can then use the Fedlet SDK to pull user attributes into the Service Provider application. The user attributes become part of the SAML Response from the Identity Provider. After the user has successfully authenticated to the Identity Provider, the Identity Provider sends the SAML Response to the Fedlet . The following figures illustrates how OpenSSO Enterprise, as the Identity Provider, determines which user attributes to include in the SAMLv2 Response to the Service Provider.

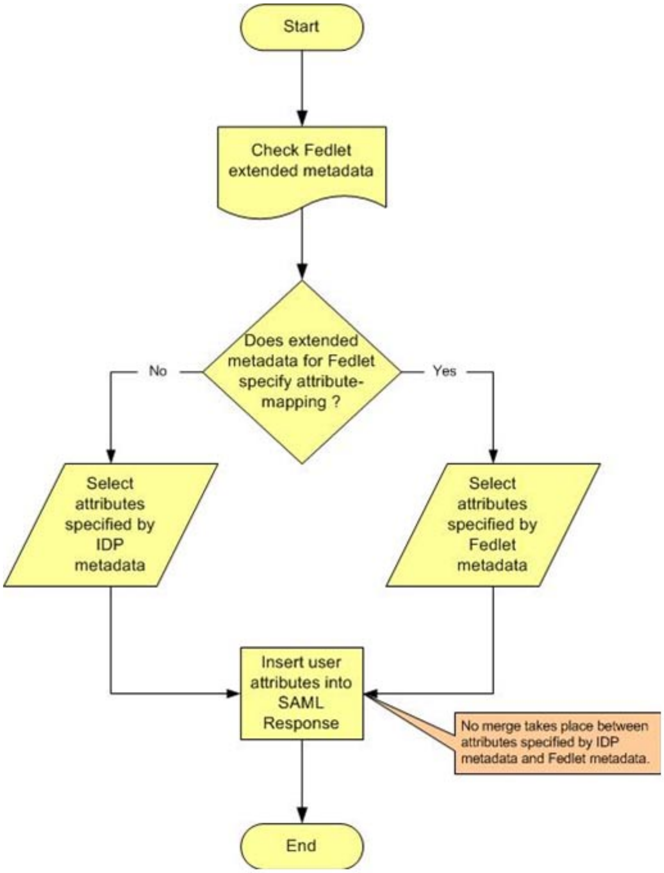

Start
Check Fedlet extended metadata
Does extended metadata for Fedlet specify attribute-mapping ?
No
Yes
Select attributes specified by IDP metadata
Select attributes specified by Fedlet metadata
Insert user attributes into SAML Response
No merge takes place between attributes specified by IDP metadata and Fedlet metadata.
End

FIGURE 5–1 OpenSSO Enterprise Determines Which User Attributes to Include in the SAML Response

Using an Identity Provider Discovery Service with Multiple Identity Providers

The Fedlet supports multiple Identity Providers. Additionally, the Fedlet supports the use of a separate Identity Provider Discovery Service to allow the user to select a preferred Identity Provider to authenticate against. When configured this way, the Identity Provider Discovery Service will remember the user's preferred Identity Provider, and communicate this to the Fedlet. The Fedlet will then be able to determine which Identity Provider to have the user authenticate to. You can deploy an OpenSSO Enterprise instance as an Identity Provider Discovery Service. However, the Fedlet can work with any Identity Provider Discovery Service. The figure below illustrates the process flow in a Fedlet deployment when the Identity Provider Discovery Service is used to set the user's preferred Identity Provider.

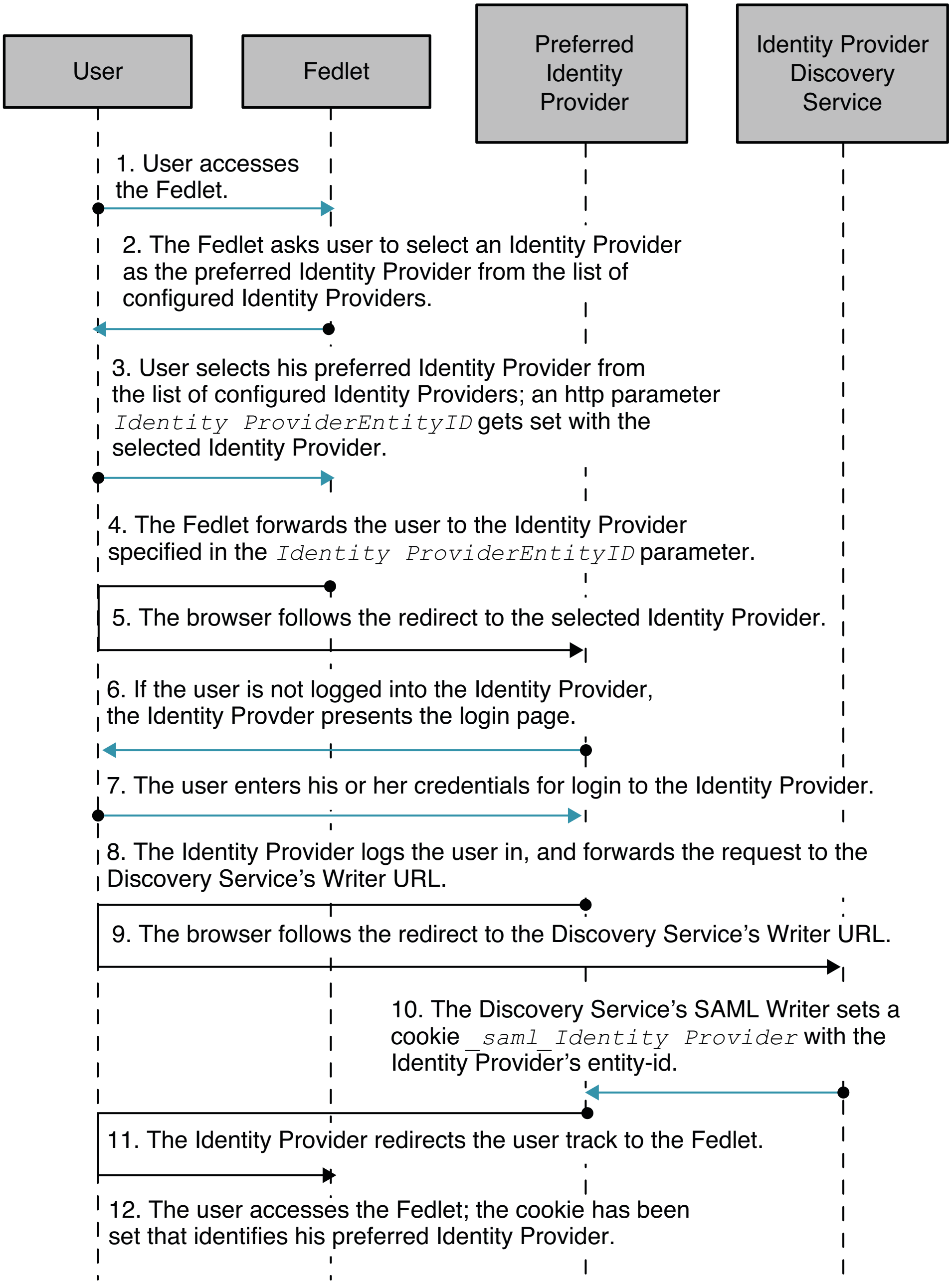

User
Fedlet
Preferred Identity Provider
Identity Provider Discovery Service
1. User accesses the Fedlet.
2. The Fedlet asks user to select an Identity Provider as the preferred Identity Provider from the list of configured Identity Providers.
3. User selects his preferred Identity Provider from the list of configured Identity Providers; an http parameter Identity ProviderEntityID gets set with the selected Identity Provider.
4. The Fedlet forwards the user to the Identity Provider specified in the Identity ProviderEntityID parameter.
5. The browser follows the redirect to the selected Identity Provider.
6. If the user is not logged into the Identity Provider, the Identity Provder presents the login page.
7. The user enters his or her credentials for login to the Identity Provider.
8. The Identity Provider logs the user in, and forwards the request to the Discovery Service's Writer URL.
9. The browser follows the redirect to the Discovery Service's Writer URL.
10. The Discovery Service's SAML Writer sets a cookie _saml_Identity Provider with the Identity Provider's entity-id.
11. The Identity Provider redirects the user track to the Fedlet.
12. The user accesses the Fedlet; the cookie has been set that identifies his preferred Identity Provider.

FIGURE 5–2 Identity Provider Discover Services is Set to User's Preferred Identity Provider

The figure below illustrates the process flow in a Fedlet deployment when the Identity Provider Discovery Service is used to retrieve the user's preferred IDP.

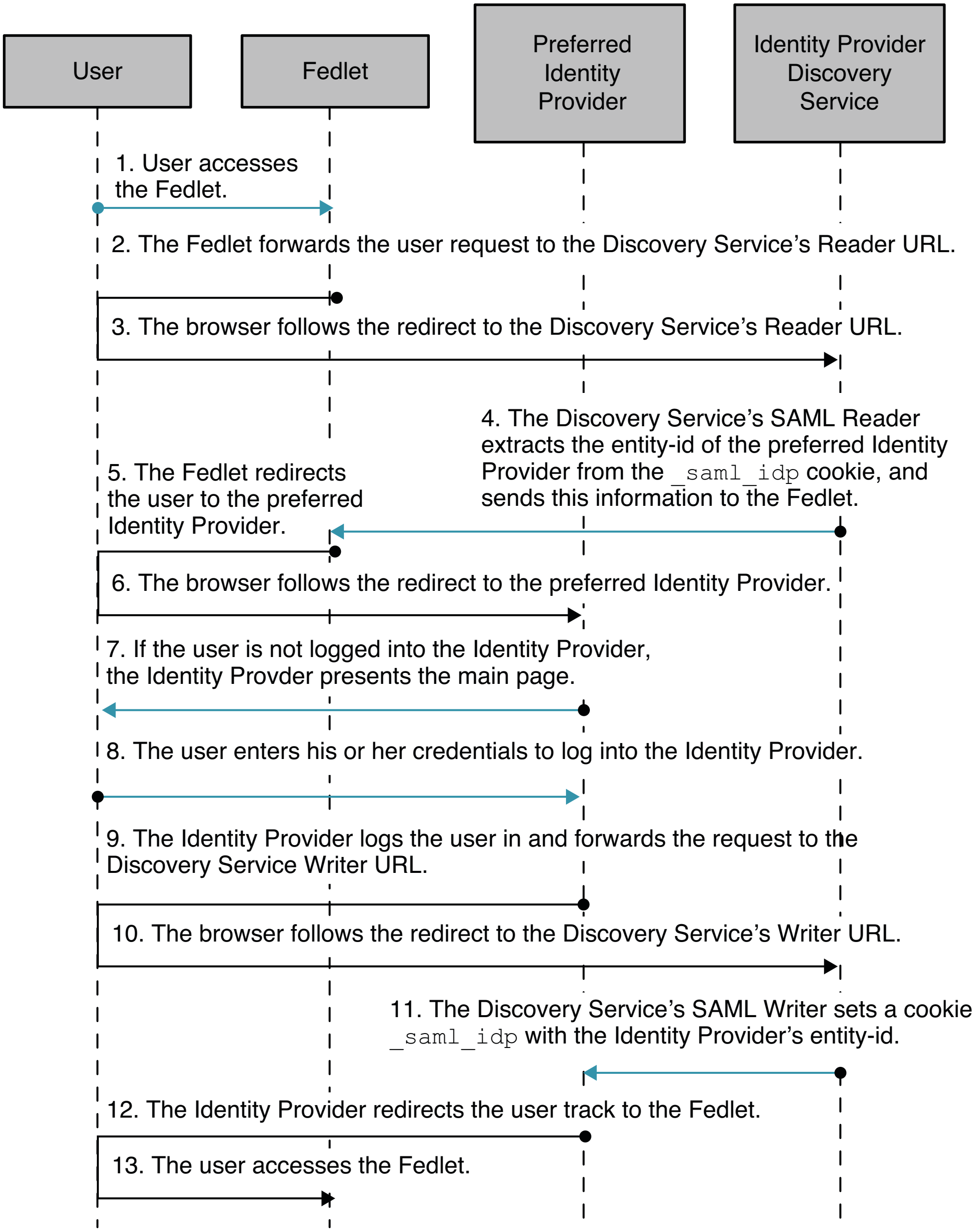

User
Fedlet
Preferred Identity Provider
Identity Provider Discovery Service
1. User accesses the Fedlet.
2. The Fedlet forwards the user request to the Discovery Service's Reader URL.
3. The browser follows the redirect to the Discovery Service's Reader URL.
4. The Discovery Service's SAML Reader extracts the entity-id of the preferred Identity Provider from the _saml_idp cookie, and sends this information to the Fedlet.
5. The Fedlet redirects the user to the preferred Identity Provider.
6. The browser follows the redirect to the preferred Identity Provider.
7. If the user is not logged into the Identity Provider, the Identity Provder presents the main page.
8. The user enters his or her credentials to log into the Identity Provider.
9. The Identity Provider logs the user in and forwards the request to the Discovery Service Writer URL.
10. The browser follows the redirect to the Discovery Service's Writer URL.
11. The Discovery Service's SAML Writer sets a cookie _saml_idp with the Identity Provider's entity-id.
12. The Identity Provider redirects the user track to the Fedlet.
13. The user accesses the Fedlet.

FIGURE 5–3 Identity Provider Discovery Service Retrieves the User's Preferred Identity Provider

Analyzing the Deployment Architecture

The main components of the circle of trust described in this chapter are the telecommunications company which acts as and Identity Provider, and a ringtone Service Provider. The following two use cases are supported by the Fedlet:

- "Identity Provider-Initiated Single Sign-On" on page 73
- "Fedlet Service Provider-initiated Single Sign-On" on page 75

The following table provides a simple comparison of the two use cases.

TABLE 5–1 Comparison of Fedlet Use Cases

Identity Provider-Initiated Single Sign-On	Service Provider-Initiated Single Sign-On
1. Mobile phone user authenticates with Telecommunications Company.	1. Mobile phone user attempts to access the ringtone Service Provider portal.
2. Upon authentication, mobile phone user accesses the ringtone Service Provider portal.	2. Ringtone Service Provider detects whether or not the mobile phone user has been authenticated by the Telecommunications Company. If not, then the ringtone Service Provider redirects the mobile phone user to the Telecommunications Identity Provider.
	3. Telecommunications Company challenges mobile phone user's credentials. Mobile user presents credentials.
	4. Upon authentication, mobile phone user accesses the ringtone Service Provider portal.

Identity Provider-Initiated Single Sign-On

The following illustrates the flow of communication in a federation scenario between a telecommunications company acting as the Identity Provider, and a ringtone provider company acting as the Service Provider.

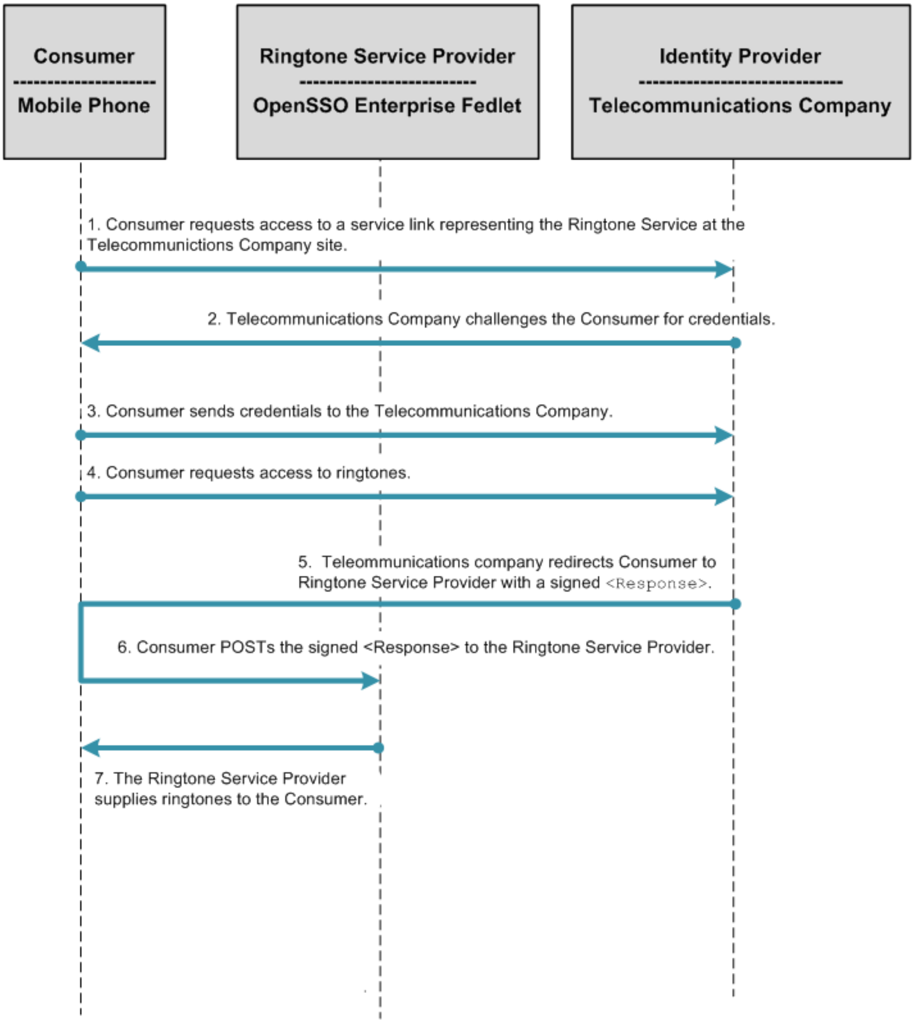

Consumer
Mobile Phone
Ringtone Service Provider
OpenSSO Enterprise Fedlet
Identity Provider
Telecommunications Company
1. Consumer requests access to a service link representing the Ringtone Service at the Telecommunictions Company site.
2. Telecommunications Company challenges the Consumer for credentials.
3. Consumer sends credentials to the Telecommunications Company.
4. Consumer requests access to ringtones.
5. Teleommunications company redirects Consumer to Ringtone Service Provider with a signed <Response>.
6. Consumer POSTs the signed <Response> to the Ringtone Service Provider.
7. The Ringtone Service Provider supplies ringtones to the Consumer.

FIGURE 5–4 Process Flow for the Fedlet in Identity Provider-initiated Single Sign-On

In an Identity Provider-initiated single sign-on scenario, the Identity Provider is configured with specialized links to specific Service Providers. These links actually refer to the local Identity Provider single sign-on service and pass parameters to the service identifying the remote Service Provider. So instead of directly visiting the Service Provider, the user goes to the Identity Provider site and clicks on one of the links to gain access to the remote Service Provider. This triggers the creation of a SAML assertion that is subsequently transported to the Service Provider.

Fedlet Service Provider-initiated Single Sign-On

In a Service Provider-initiated single sign-on scenario, the user attempts to access a resource on the Service Provider. However the user does not have a current logon session on this site, and the user's federated identity is managed by the Identity Provider. The user is sent to the Identity Provider to log on. The Identity Provider creates a SAML assertion for the user's federated identity and sends it back to the Service Provider. The following figure illustrates the process flow.

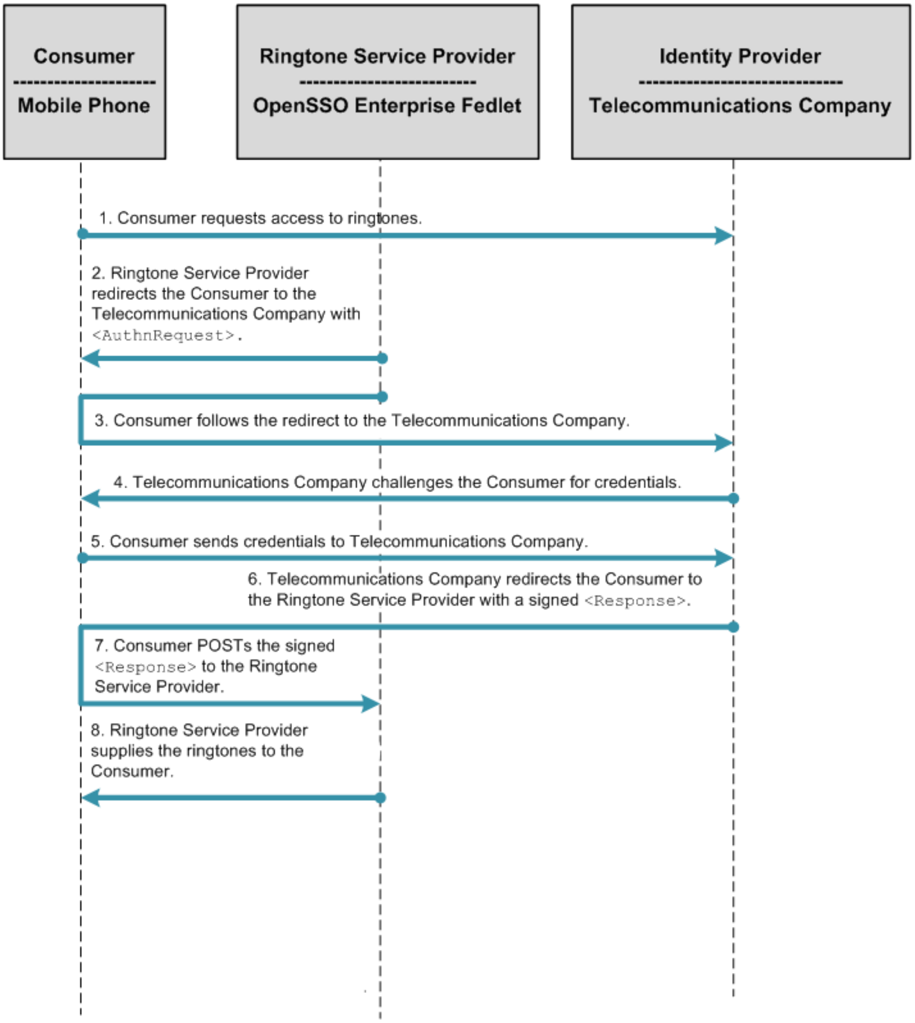

Consumer
Mobile Phone
Ringtone Service Provider
OpenSSO Enterprise Fedlet
Identity Provider
Telecommunications Company
1. Consumer requests access to ringtones.
2. Ringtone Service Provider redirects the Consumer to the Telecommunications Company with <AuthnRequest>.
3. Consumer follows the redirect to the Telecommunications Company.
4. Telecommunications Company challenges the Consumer for credentials.
5. Consumer sends credentials to Telecommunications Company.
6. Telecommunications Company redirects the Consumer to the Ringtone Service Provider with a signed <Response>.
7. Consumer POSTs the signed <Response> to the Ringtone Service Provider.
8. Ringtone Service Provider supplies the ringtones to the Consumer.

FIGURE 5–5 Process Flow for Fedlet Service Provider-initiated Single Sign-On

Considering Deployment Assumptions, Dependencies, and Constraints

As you plan your deployment, consider the following assumptions, dependencies, and constraints to determine if your environment is appropriate for using the Fedlet.

Assumptions and Dependencies

- The configuration file and the metadata for the Fedlet is stored in a flat-file repository at the Service Provider.
- This solution uses the HTTP POST bindings for transport between the Identity Provider and the Service Provider.

Constraints

- The Fedlet supports Identity Provider-initiated single sign-on using only HTTP POST bindings.
- The Fedlet supports Fedlet Service Provider-initiated single sign-on using only HTTP POST bindings.
- In this deployment, no keystore exists to store certificates used for encrypting and signing SAMLv2 message elements. The Fedlet does not support the encryption and signing of SAMLv2 message elements. This capability may be implemented in a future release of OpenSSO Enterprise. This constraint has implications about ensuring the confidentiality and integrity of messages. Until the Fedlet can support the encryption and signing of SAMLv2 message elements, you are encouraged to use SSL/TLS at the message transport layer to secure exchanges between browser and server. The ensures that exchanges are secured at least while the SAML messages are in-transit. The Fedlet does support the verification of XML signature in the SAMLv2 Response from Identity Provider.

Using The Fedlet with Multiple Identity Providers

You can install multiple Fedlet instances at the Service Provider so that each Fedlet instance talks to a different Identity Provider. Or you can deploy a single OpenSSO Enterprise instance at the Service Provider.

If you want to install multiple Fedlet instances so that each Fedlet instance talks to a different Identity Provider, use caution with this approach. Consider the following example. A ringtone provider acts as a Service Provider and conducts business with multiple telecommunications companies. Each telecommunications company acts as its own Identity Provider. The Service Provider might have to deploy multiple instances of its Ringtone Application, each with its own Fedlet instance. Each Fedlet instance would communicate with a different telecommunications company Identity Provider. The result is that each Identity Provider would be using a different instance of the Ringtone Application.

Consider another example. The Fedlet is deployed on Sun Application Server, and the Fedlet home-directory is configured in the Application Server domain configuration file, `domain.xml`. So for each new Fedlet instance, a new Application Server domain must be set up, and an instance of the Ringtone Application must be deployed on this new Application Server domain.

Now the Service Provider has to maintain two Application Server domains for the same Ringtone Application. This presents two possibilities. One possibility is that the same Ringtone Application is run on different ports for different Identity Providers. The second possibility is that the same Ringtone Application is run on the same port on different machines. This could also translate into different Ringtone Application URLs that each Identity Provider will use with the Service Provider. Or the Service Provider would have to implement some logic to route to the correct Ringtone Application based on the particular Identity Provider requesting it.

Understanding Typical Business Use Cases

The following use cases illustrate why companies might choose to use the Fedlet in their environment.

Saving Time and Reducing Overhead

In this use case, the Service Provider needs to single sign-on with an Identity Provider that has OpenSSO Enterprise installed in the Identity Provider environment. But the Service Provider does not want to install the full-featured OpenSSO Enterprise just to enable federation. The Service Provider cites one or more of the following reasons for not installing OpenSSO Enterprise:

- The Service Provider is small company that provides one application as only part of their service.
- The Service Provider wants identity federation at minimum cost.
- Installing OpenSSO Enterprise would require investments in hardware, services, and human resources that the Service Provider does not want to make.
- Installing OpenSSO Enterprise would require the Service Provider system administrators to be proficient in implementing identity federation protocols in order to configure and maintain the OpenSSO Enterprise federation deployment.
- The Service Provider wants to quickly enable federation in their environment in a very short timeframe.
- The Service Provider wants only to implement single sign-on with the Identity Provider and retrieve some user attributes for customizing the service to the user. The Service Provider does not want to install a full-featured federation solution and just to use two features.

Customizing Content Based on User Attributes

In this use case, a telecommunications company acts as an Identity Provider. The telecommunications company subscribers use the custom telecommunications company user portal to receive personalized content. The content is received by the telecommunications company from its business partners such as StockService.com, Weather.com, and so forth. OnCast, a new partner of the telecommunications company, uses Fedlets for its portal user single sign-on. Through the Fedlets, OnCast retrieves specific user attributes over SAML from the telecommunications company. OnCast then uses the user attribute data to customize its content deliver.

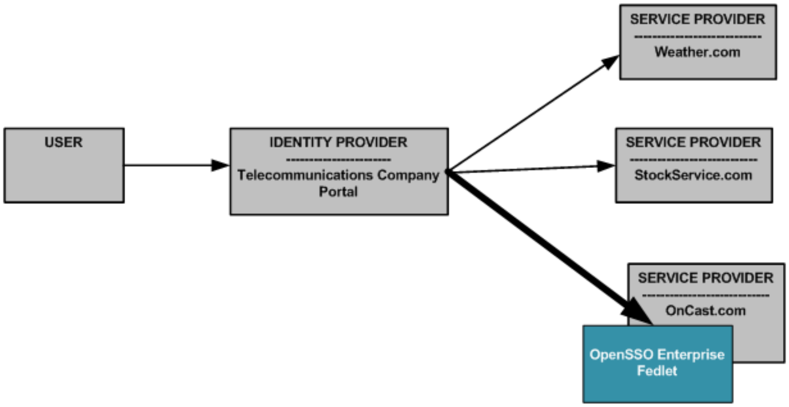


FIGURE 5–6 Fedlet Service Provider Customizes Content Based on User Attributes

See the demonstration of this business use case at `http://blogs.sun.com/sid/resource/fedlet.html` .

Setting Up and Configuring the Fedlet

This section describes the high-level tasks to setup the Fedlet at the Service Provider. For more detailed instructions, see the README file contained in the `Fedlet.zip` and `Fedlet-unconfigured.zip` files.

Technical Requirements

- Any J2EE-complaint server on which to deploy the Fedlet
- JDK 1.5 or higher

Obtaining and Deploying the OpenSSO Fedlet Bundle

You can choose one of two methods for obtaining and deploying the Fedlet Bundle.

If OpenSSO is deployed as an Identity Provider, then use the OpenSSO Enterprise console to create the Fedlet bundle. In this scenario, using the console is the faster and easier method because the Identity Provider follows the same workflow to integrate with any Service Provider.

If multiple Identity Providers exist in the Service Provider circle of trust, and not all Identity Providers use OpenSSO Enterprise, then use the Fedlet Demo. The Fedlet Demo contains a sample JSP is packaged in the `fedlet.war`. The `fedlet.war` file emulates the Service Provider web application. Using the `fedlet.war` file makes it easy to demonstrate a simple JSP receiving the SAMLv2 POST from the Identity Provider.

To Use the OpenSSO Enterprise Console to Create the Fedlet bundle

In the OpenSSO Enterprise console, navigate through a taskflow and provide the following:

1. Name of the Service Provider
2. Destination URL of the Service Provider that will include the Fedlet
3. The circle of trust in which to place the Service Provider

At the end of the taskflow, a `Fedlet.zip` bundle is automatically created. The bundle consists of the `fedlet.war` file and a README file that contains instructions for deploying the Fedlet. Follow the instructions to deploy the Fedlet.

To Use the Pre-Built Fedlet

As the Service Provider, download the `opensso.zip` file. Then follow the instructions in the README file contained in the `Fedlet-unconfigured.zip` file to deploy and configure the Fedlet. The `Fedlet-unconfigured.zip` file is bundled into the `opensso.zip`.

▼ To Set Up the Workflow-based Fedlet

1 **Install and Configure OpenSSO Enterprise on the Identity Provider.**

2 **On the Identity Provider, navigate through the Workflow on the OpenSSO Enterprise console to create the `Fedlet.zip` file.**

The `Fedlet.zip` file contains:

- `README.txt`: A text file that contains instructions for deploying the `fedlet.war` and for integrating the Fedlet into an existing application.
- `fedlet.war`: The Fedlet ready-to-deploy WAR file.

3 **Send the `Fedlet.zip` file to the Service Provider.**

4 **Deploy and configure the `fedlet.war` file, on the Service Provider.**

5 **Verify that the Fedlet was successfully installed.**

a. **Access the index.jsp file on the Fedlet deployment.**

b. **Click the link to create the Fedlet configuration automatically.**

c. **Follow the two links in the page to test the following use-cases :**

- (Fedlet) Service Provider-initiated single sign-on
- Identity Provider-initiated single sign-on through the hyperlinks present on the page.

▼ To Use the Pre-Built Fedlet

1 **Download the Fedlet-unconfigured.zip.**

`Fedlet-unconfigured.zip` is contained in the `opensso.zip` distribution. The `Fedlet-unconfigured.zip` file contains:

- `fedlet.war`

  The Fedlet ready-to-deploy WAR file
- `conf`

  A directory containing the Fedlet metadata template, circle of trust template, and various configuration files
- `README.txt`

  A text file that provides instructions for using the `\conf` files to configure the Fedlet

2 **Extract the `Fedlet-unconfigured.zip` file.**

Follow the instructions in the README file to set local configuration files for the Fedlet.

3 **Send tag-swapped Service Provider metadata files to the Identity Provider, and request the Identity Provider metadata files from the Identity Provider.**

4 **Verify that the Fedlet is successfully installed.**

Access the `index.jsp` file on the Fedlet deployment, and test the following use-cases : Fedlet (SP)-initiated SSO IDP-initiated SSO through the hyperlinks present on the page.

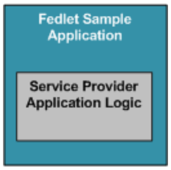
- (Fedlet) Service Provider-initiated single sign-on
- Identity Provider-initiated single sign-on through the hyperlinks present on the page

Embedding the Fedlet into Service Provider Applications

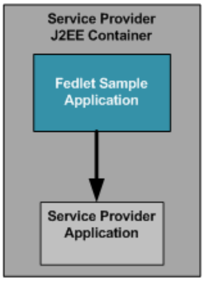
The README file contained in the `Fedlet.zip` and the `Fedlet-unconfigured.zip` files provides instructions for integrating the Fedlet demo into the Service Provider application. You need to embed all the `properties/jars/JSPs/images` and so forth in the demo `fedlet.war` into your existing application WAR. Merge the `fedlet.war` with your existing application WAR. The Fedlet provides a default Assertion Consumer endpoint named `fedletSampleApp.jsp` to process the SAMLv2 Assertion from the Identity Provider.

Use one of the following approaches to embed the Fedlet into the Service Provider applications:

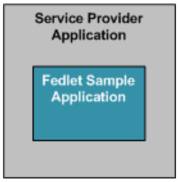
- Use `fedletSampleApp.jsp` as the endpoint on Fedlet side, and modify `fedletSampleApp.jsp` to add the Service Provider application logic.



- Use `fedletSampleApp.jsp` as the endpoint on Fedlet side, and modify `fedletSampleApp.jsp` to forward the request to the Service Provider application URL.


Service Provider J2EE Container
Fedlet Sample Application
Service Provider Application

- Create a new endpoint, for example `servlet.jsp` URL, on the Fedlet side to replace the `fedletSampleApp.jsp` or to embed the Fedlet into the Service Provider application. You can copy some of the code in the `fedletSampleApp.jsp` to the new endpoint code. Details of the actual code you can transfer are described in the README file.



Evaluating Benefits and Tradeoffs

As you design your deployment architecture, be sure to consider the benefits, tradeoffs. The following lists may help you determine if the Fedlet is appropriate to meet your business needs.

Benefits

- The Fedlet does not require additional hardware, thus reducing the cost to the Service Provider and increasing the return on investment on existing hardware.
- The Fedlet is easy to deploy and to embed into the Service Provider application. Configuration on the Fedlet, if needed at all, requires modifying only three to four parameters. This enables you to go live with the application much more quickly than deploying a full-featured federation solution.
- The Fedlet enables the Service Provider to quickly enable federation into their applications, resulting in shorter time-to-market for their applications with the Identity Provider.
- The Fedlet does not require the Service Provider to install any full-featured federation software. This reduces the amount of training required, thus reducing training costs for the Service Provider.
- The Fedlet is ideal for a Service Provider that wants only to achieve single sign-on with an Identity Provider, and to be able to retrieve user attributes from the Identity Provider.
- The Fedlet is compliant with SAMLv2 standards.

Tradeoffs

- The Fedlet will not perform session management on the Service Provider. The application or container must perform session management.
- The Fedlet supports single sign-on using the SAMLv2 protocol only. Other federation protocols such as Liberty ID-FF, WS-Federation, and SAML 1.x, are not supported.
- The Fedlet solution enables only single sign-on with an IDP and retrieval of user attributes. Advanced features, typically available in a full-featured federation product such as OpenSSO Enterprise, are not available in the Fedlet:
  - IDP Proxying
  - Single Logout

- Auto Federation
- Account Linking Auto-creation of users on the SP
- Declarative policy integration with roles asserted from the IDP

Finding More Information

- OpenSSO Homepage

  `http://opensso.org`
- *SAML v2.0 Technical Overview*

  `http://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf`

◆ ◆ ◆ CHAPTER 6

Implementing a Virtual Federation Proxy (Secure Attributes Exchange)

Secure Attributes Exchange, also known as Virtual Federation Proxy, is a new feature in Sun OpenSSO Enterprise 8.0. Secure Attributes Exchange provides the means for an application to securely transfer identity information to another application located in a different domain. Secure Attributes Exchange enables legacy applications to leverage standards-based federation to communicate with other existing applications and services *without having to set up and configure federation protocols and processes for each application*.

This chapter provides information to help you determine whether Secure Attributes Exchange is appropriate for your environment. The following topics are contained in this chapter:

About Virtual Federation Proxy (Secure Attributes Exchange)

Multiple authentication systems often exist in typical legacy environments. Although these authorization systems would work more efficiently if they were federated, implementing single sign-on often requires deploying one federation software instance for each of the authorization systems in the environment (see the following figure). The complexities of such deployments usually impose additional constraints in selecting federation solutions, and impede any progress toward enabling federation among the many authorization systems.

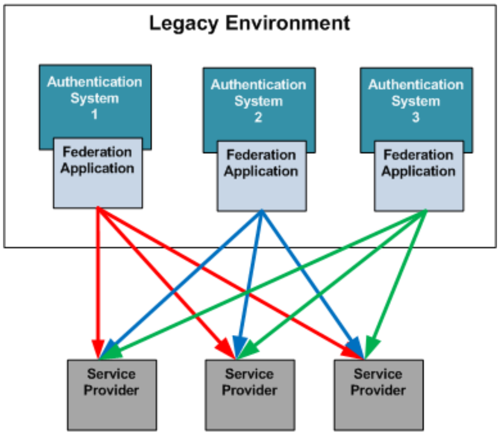


FIGURE 6–1 Multiple Authentication Systems in a Legacy Environment

SAMLv2 and other federation protocols may provide quick, standards-based federation enablement. But legacy identity systems on the enterprise end and existing Identity Provider applications cannot pass user authentication, user profile, and other transaction related data to the local Identity Provider instance. Similarly, the existing framework also limits the ways in which Service Provider applications consume user authentication, profile, and transaction information.

The Secure Attributes Exchange feature introduced in OpenSSO Enterprise 8.0 is designed to meet these business needs. OpenSSO Enterprise enables an OpenSSO Enterprise instance in either the Identity Provider role or in the Service Provider role to act like a pure SAMLv2 protocol gateway. Simple, default security mechanisms are implemented to allow a loose coupling between the existing applications and OpenSSO Enterprise instances. The following figure illustrates how a streamlined solution enables federation among multiple legacy authorization systems with a centralized configuration.

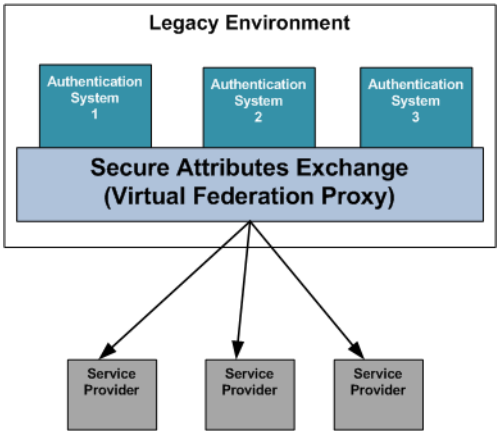


FIGURE 6–2 Multiple Authentication Systems Using Secure Attributes Exchange

A Secure Attributes Exchange interaction enables the following:

- Identity Provider applications push user authentication, user profile, and transaction information to a local instance of OpenSSO Enterprise. The local instance of OpenSSO Enterprise, using federation protocols, passes the data to a remote instance of OpenSSO Enterprise at the Service Provider.
- Service Provider applications consume the received information in a secure manner.

In this first offering of Secure Attributes Exchange, only OASIS SAMLv2 protocol is supported. However, the solution can be extended in the future to be completely protocol-neutral so that other single sign-on protocols such as Liberty ID-FF and WS-Federation can also be supported.

Analyzing the Deployment

Secure Attributes Exchange uses the SAMLv2 protocol to transfer identity data between the communicating entities. The Secure Attributes Exchange client APIs, including both Java and .NET interfaces, run independently of the OpenSSO Enterprise instance. The Secure Attributes Exchange client APIs enable existing applications to handle the SAMLv2 interactions.

The following figure illustrates the deployment architecture for Secure Attributes Exchange.

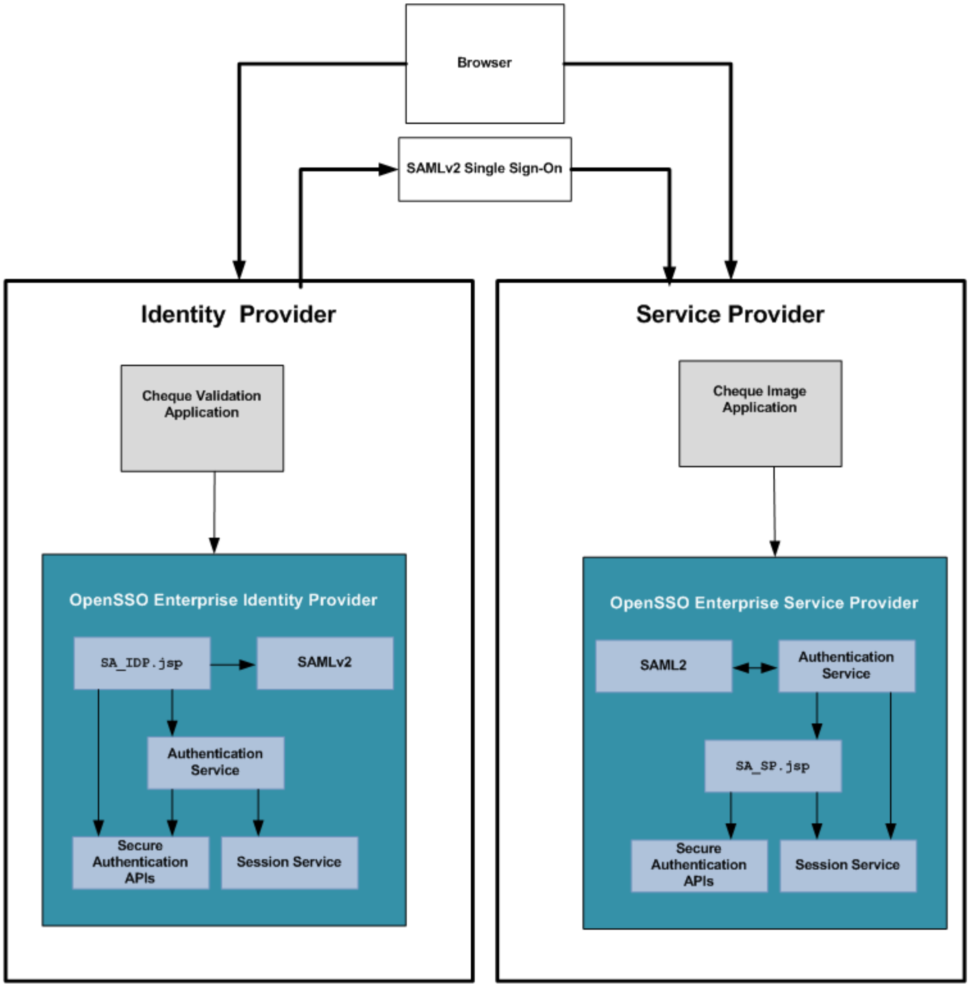


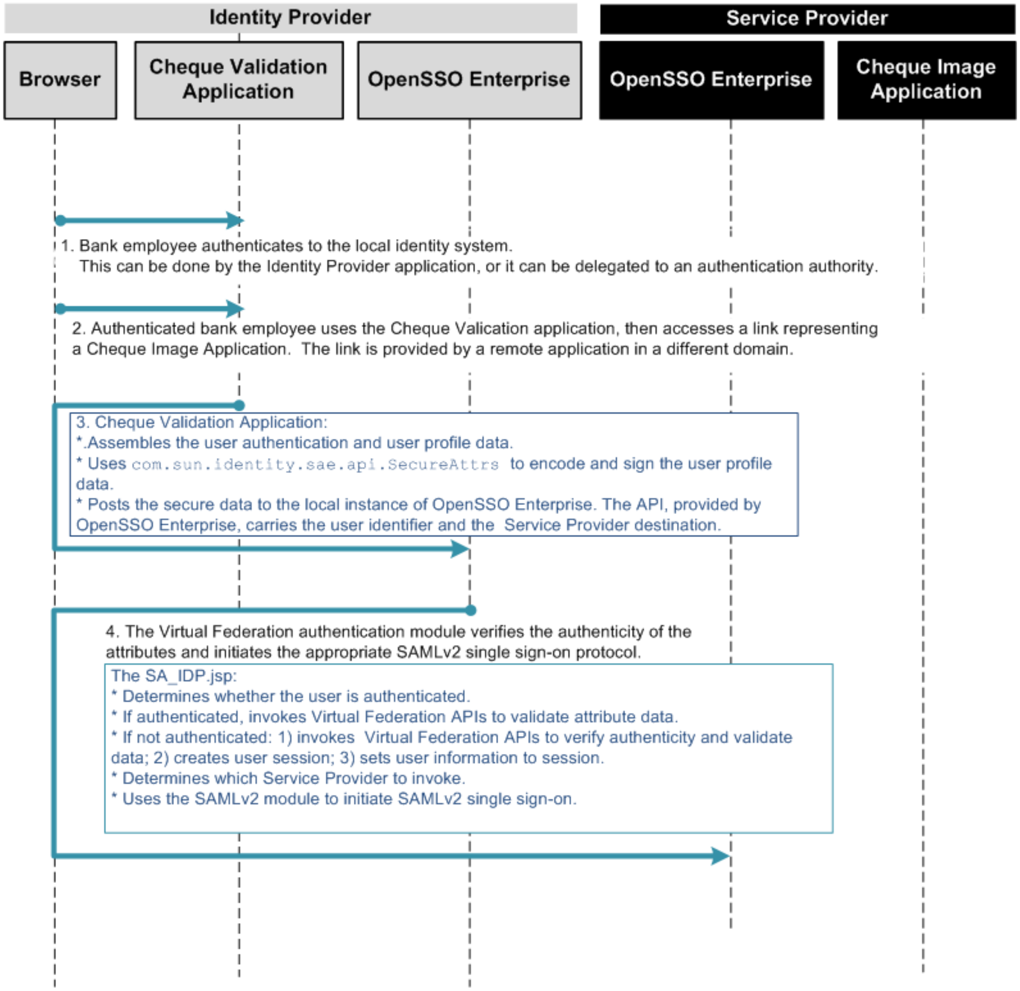
FIGURE 6–3 Deployment Architecture for Secure Attributes Exchange

In this Secure Attributes Exchange example:

- Secure Attributes Exchange acts as a pure SAMLv2 protocol gateway, relying entirely on the existing applications for user authentication and retrieval of the user profile attributes.
- From the perspective of Identity Provider applications and Service Provider applications, Secure Attributes Exchange implements a push-model solution to securely transfer authentication and attributes data.
- Secure Attributes Exchange uses simple HTTP GET/POST and data signing and encryption to securely transfer authentication and attributes data from and to legacy applications.

- Secure Attributes Exchange supports both symmetric-key and asymmetric-key based cryptography to secure authentication and attributes data
- Secure Attributes Exchange supports application-initiated single sign-on and single logout.

The figures Figure 6–4 and Figure 6–5 illustrate the process flow in a typical Secure Attributes Exchange interaction. In this example, bank employees each have a user account in a bank's employee identity system. Employees routinely access an internal application that validates bank customers' personal checks. The bank employees are required to authenticate themselves before accessing the Cheque Validation application. Validating checks involves retrieving the check images which are stored and processed by the Cheque Image application. The Cheque Image application which is hosted by a business partner at a remote site. User identity and attribute data must be supplied by the local Cheque Validation application and passed to the remote Cheque Image application in a secure manner.



1. Bank employee authenticates to the local identity system.
This can be done by the Identity Provider application, or it can be delegated to an authentication authority.

2. Authenticated bank employee uses the Cheque Valication application, then accesses a link representing a Cheque Image Application. The link is provided by a remote application in a different domain.

3. Cheque Validation Application:
*.Assembles the user authentication and user profile data.
* Uses `com.sun.identity.sae.api.SecureAttrs` to encode and sign the user profile data.
* Posts the secure data to the local instance of OpenSSO Enterprise. The API, provided by OpenSSO Enterprise, carries the user identifier and the Service Provider destination.

4. The Virtual Federation authentication module verifies the authenticity of the attributes and initiates the appropriate SAMLv2 single sign-on protocol.

The SA_IDP.jsp:
* Determines whether the user is authenticated.
* If authenticated, invokes Virtual Federation APIs to validate attribute data.
* If not authenticated: 1) invokes Virtual Federation APIs to verify authenticity and validate data; 2) creates user session; 3) sets user information to session.
* Determines which Service Provider to invoke.
* Uses the SAMLv2 module to initiate SAMLv2 single sign-on.

FIGURE 6–4 Process Flow for Secure Attributes Exchange (Continued on next page)

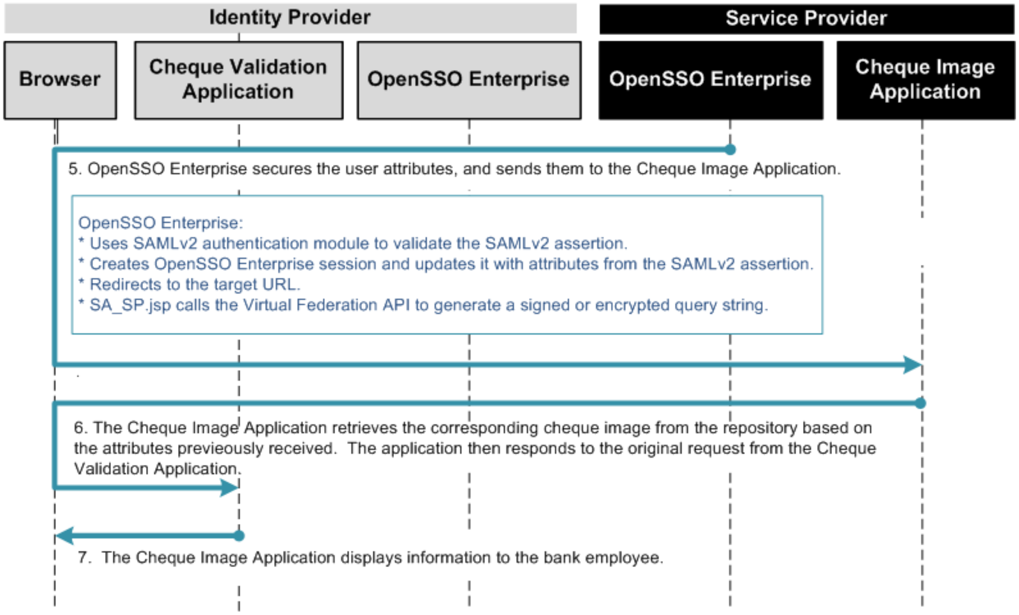

Identity Provider
Service Provider
Browser
Cheque Validation Application
OpenSSO Enterprise
OpenSSO Enterprise
Cheque Image Application
5. OpenSSO Enterprise secures the user attributes, and sends them to the Cheque Image Application.
OpenSSO Enterprise:
* Uses SAMLv2 authentication module to validate the SAMLv2 assertion.
* Creates OpenSSO Enterprise session and updates it with attributes from the SAMLv2 assertion.
* Redirects to the target URL.
* SA_SP.jsp calls the Virtual Federation API to generate a signed or encrypted query string.
6. The Cheque Image Application retrieves the corresponding cheque image from the repository based on the attributes previeously received. The application then responds to the original request from the Cheque Validation Application.
7. The Cheque Image Application displays information to the bank employee.

FIGURE 6–5 Process Flow for Secure Attributes Exchange (Continued)

Considering Assumptions, Dependencies, and Constraints

Before you can implement Secure Attributes Exchange, you must consider the following assumptions and constraints.

Assumptions

- The implementation will automatically use the `authn=Force` functionality to allow Identity Provider applications to repeatedly invoke the secure attribute transfer feature.
- The mappings between the Service Provider application URL prefix to the Service Provider `id` have been explicitly defined in the product configuration.
- Local Service Provider single logout functionality relies on OpenSSO Enterprise be able to redirect to an external URL during SAMLv2 single logout processing, and then resume the original SAMLv2 single logout when redirected back.
- Cryptography APIs are assumed to be available on all targeted platforms and programming languages.

Constraints

- The scope of the current OpenSSO Enterprise implementation is limited to SAMLv2–based single sign-on only. However, it is possible for you to extend this to other single sign-on protocols such as the Liberty Identity Federation Framework (ID-FF) and Web Service Federation (WS-F) specifications.
- The existing implementation supports only simple HTTP GET and POST based mechanisms. However, you should be able to use the same APIs to do XML-based attribute transfer as well.
- Integration with specific proprietary application server technologies such as WebSphere LTPA and WebLogic for single sign-on as well as single logout are not addressed in this document.
- Limited single logout functionality is implemented. Logout from multiple Identity Provider applications and multiple Service Provider applications is not directly addressed in this document. The Identity Provider and Service Provider must implement logout appropriately to meet their own needs.

Secure Attributes Exchange Client APIs

The Secure Attributes Exchange feature provides a set of client APIs in both Java and .NET interfaces. the client APIs are used to enable following:

- Identity Provider applications push user authentication, user profile, and transaction related information to the local OpenSSO Enterprise instance. The information is then passed over to the Service Provider side using the federation protocols,
- Service Provider applications consume the same user profile and transaction information.

The details of Secure Attributes Exchange client APIs are described in detail in the *Sun OpenSSO Enterprise 8.0 Java API Reference* .

Understanding Typical Business Use Cases

Secure Attributes Exchange is used by three types of users:

- Developers

  As application owners, developers use the Secure Attributes Exchange client APIs to connect to a local OpenSSO Enterprise instance.
- Administrators

  Administrators are responsible for configuring Secure Attributes Exchange, setting up and maintaining provision keys, configuring authorization of each application, and so forth.
- End users

End users access the deployed applications.

The figures Figure 6–4 and Figure 6–5 illustrate a typical process flow for Secure Attributes Exchange.

The process flow can be described as the sum of four separate uses cases:

- "Authentication at Identity Provider" on page 93
- "Secure Attribute Exchange at the Identity Provider " on page 93
- "Secure Attribute Exchange at the Service Provider " on page 93
- "Global Single Logout" on page 94

It is not absolutely required for service providers to implement the Secure Attributes Exchange functionality. This is certainly a valid business use case as long as the receiving end is a SAMLv2 compliant Service Provider that is capable of using the information originating from the Identity Provider application and sent by the Identity Provider.

Authentication at Identity Provider

When a user is already authenticated in an enterprise, the legacy identity provider application sends a secure HTTP GET/POST message to OpenSSO Enterprise asserting the identity of the user. OpenSSO Enterprise then verifies the authenticity of the message and establishes a session for the authenticated user. Secure Attributes Exchange can be used to transfer the user's authentication information to the local instance of OpenSSO Enterprise in order to create a new session.

Secure Attribute Exchange at the Identity Provider

When a user is already authenticated by, and attempts access to, a legacy identity provider application, the legacy application sends a secure HTTP POST message to the local instance ofOpenSSO Enterprise. The HTTP POST message asserts the user's identity and contains a set of attribute-value pairs related to the user. For example, the attribute-value pairs may contain data from the persistent store, and the data may represent certain transactional states in the application. OpenSSO Enterprise verifies the authenticity of the message, establishes a session for the authenticated user, and then populates the session with the user attributes.

Secure Attribute Exchange at the Service Provider

When a user is already authenticated by the instance of OpenSSO Enterprise at the Identity Provider, and OpenSSO Enterprise invokes an Identity Provider application that calls for redirection to a Service Provider, the Identity Provider invokes secure attribute exchange at either the Service Provider or Identity Provider as described above. OpenSSO Enterprise encodes a SAMLv2 single sign-on URL as a part of the request. The Identity Provider instance of OpenSSO Enterprise then initiates SAMLv2 single sign-on with the instance of OpenSSO Enterprise at the Service Provider. The Service Provider instance of OpenSSO Enterprise then verifies the SAMLv2 assertion and the included attributes, and redirects to the Service Provider application. The user attributes are securely transferred using a secure HTTP POST message. The Service Provider application consumes the attributes, establishes a session, and then offers the service to the user.

Global Single Logout

Global single logout can be implemented in various ways. In this example, a user is already authenticated and has established single sign-on with the Service Provider instance of OpenSSO Enterprise. The user clicks on a Global Logout link. The Identity Provider will then invalidate its local session, if it's already created, and trigger SAMLv2 single logout by invoking a provided OpenSSO Enterprise URL. The OpenSSO Enterprise Identity Provider executes the SAMLv2 single logout, terminating the session on both Identity Provider and Service Provider instances of OpenSSO Enterprise.

Setting Up and Configuring Secure Attributes Exchange

Before configuring and using the Secure Attributes Exchange, administrators must make some decisions regarding security-related settings such as cryptography type, applicable keys, and application identifiers. Administrators must be familiar with basic SAMLv2 concepts and the SAMLv2 samples bundled with OpenSSO Enterprise.

This section provides a high-level summary you need to resolve before configuring Secure Attributes Exchange.

About Cryptography Type

Secure Attributes Exchange provides symmetric and asymmetric cryptography types to secure identity attributes between an instance of OpenSSO Enterprise and an application.

- Symmetric cryptography

  Involves the use of a shared secret key known only to the participants in the communication. The key is agreed upon beforehand and is used to encrypt and decrypt the message.
- Asymmetric cryptography

  Uses two separate keys for encryption and the corresponding decryption - one public key and one private key. The information is encrypted with a public key known to all, and then decrypted by the recipient only, using a private key to which no one else has access.

This process is known as a public key infrastructure (PKI). On the Identity Provider side, the public key must be added to the OpenSSO Enterprise keystore. The private key must be stored in a protected keystore such as a Hardware Security Module (HSM) for access by the Identity Provider application. On the Service Provider side, the private key must be added to the OpenSSO Enterprise keystore, and the public key stored in a keystore that is local to the Service Provider application.

Overview of Setup Steps

1. Establish trust among the application or multiple applications and the instance of OpenSSO Enterprise on the Identity Provider. This includes the configuring the cryptography type, applicable keys, and application identifiers.
2. Establish trust among the application or multiple applications and the instance of OpenSSO Enterprise on the Service Provider side. This includes configuring the cryptography type, applicable keys, and application identifiers.
3. (Optional) The following steps are specific to using SAMLv2 with auto-federation.
   a. Determine which identity attributes you want transferred as part of the SAMLv2 single sign-on interaction.
   b. Determine which attribute you will use to identify the user on the Service Provider side.
4. Determine which URL on the Service Provider will be responsible for handling logout requests from the Identity Provider.

Configuring Secure Attributes Exchange

Secure Attributes Exchange configuration involves modifying two different OpenSSO Enterprise installations: one OpenSSO Enterprise instance on the Identity Provider side, and one OpenSSO Enterprise instance on the Service Provider side. Before proceeding with the instructions in this chapter, you must download and deploy the OpenSSO Enterprise WAR file to a supported web container.

A SAMLv2 provider with Secure Attributes Exchange can be configured by using one of the following alternatives: *

- In the administration console, use the OpenSSO Enterprise Common Tasks interface to configure SAML configuration and to configure Secure Attribute Exchange.
- Import the metadata using the command-line interface.

About the Software Binaries

The software binaries for Secure Attributes Exchange in OpenSSO Enterprise are included in the following components. Locations are relative within the `opensso.zip` file.

- Deployable `opensso.war`

  The OpenSSO Enterprise server
- `libraries/jars/openssoclientsdk.jar`

  For client applications using Java APIs
- `libraries/dll/famsae.dll`

  For client applications using .NET APIs

High-level Configuration Steps

For detailed instructions for configuring Secure Attributes Exchange, see the `Administration Guide`. For deployment planning purposes, the following provides a high-level overview of steps to configure Secure Attributes Exchange:

1. Configure the instance of OpenSSO Enterprise on the Identity Provider side for the hosted Identity Provider.
   a. Set up trust between the Identity Provider application and the OpenSSO Enterprise Identity Provider instance.

      Determine and configure the cryptography type, applicable keys, and application identifiers.

   b. Determine the Identity Provider application name.

   c. Determine the Identity Provider Secure Attributes Exchange handler URL.

   d. Set up attribute mapping.
2. Configure the instance of OpenSSO Enterprise on the Identity Provider side for the remote Service Provider.
   a. Set up the attribute mapping.
   b. Determine the Service Provider Virtual Federation handler URL.
3. Configure the instance of OpenSSO Enterprise on the Service Provider side for the hosted Service Provider.
   a. Set up trust between Service Provider application and OpenSSO Enterprise Service Provider instance.

      Determine and configure the cryptography type, applicable keys, and application identifiers.

   b. Turn on auto-federation and specify the attribute that will identify the user's identity

   c. Determine the Service Provider Application URL.

d. Set up attribute mapping.

e. Determine the Service Provider logout URL.

Evaluating Benefits and Tradeoffs

Most enterprises today have to deal with various legacy applications and identity systems. It is challenging to make any major infrastructure change simply to accommodate identity federation.

Benefits

- Secure Attributes Exchange allows businesses to quickly implement standards-based federation without having to invest in expensive tooling to bridge incompatibilities.
- The Secure Attributes Exchange feature provides a very simple mechanism that enables an application to communicate identity information with a partner application in two different domains. Communication can take place within a circle of trust using the SAMLv2 protocol. This functionality can be extended to support other federation specifications as well.
- The Secure Attributes Exchange is a major step forward in the direction of identity federation enabling the legacy systems. Secure Attributes Exchange provides integration points, adapters and deployment guidance that can help facilitate the adoption of federation solutions.

Tradeoffs

Although the Secure Attributes Exchange feature in OpenSSO Enterprise makes it easier to implement identity federation among legacy applications, a SAMLv2–compliant Service Provider must already be in place. The Service Provider can be OpenSSO Enterprise or any other vendor solution. But even a small Service Provider requires an identity federation-aware software infrastructure in order to make use of Secure Attributes Exchange.

An alternative to Secure Attributes Exchange is to enable identity federation using the OpenSSO Enterprise Fedlet. The Fedlet is a streamlined Service Provider implementation used to quickly and simply enable identity federation. The Fedlet does not require the installation of any other identity federation software components such as the OpenSSO Enterprise server. For more information about the Fedlet, see Chapter 5, “Using the OpenSSO Enterprise Fedlet to Enable Identity Federation”

CHAPTER 7

Implementing a SAMLv2 Identity Provider Proxy

A SAMLv2 Identity Provider Proxy acts as a conduit connecting federated identity providers with federated services providers. In OpenSSO Enterprise, Identity Provider Proxy enables an Identity Provider to proxy authentication requests from federated Service Providers to various Identity Providers to which the user has previously authenticated.

This chapter provides information to help you determine if Identity Provider Proxy is suitable for your environment. The following topics are contained in this chapter:

About the SAMLv2 Identity Provider Proxy Specification

The OpenSSO Enterprise Identity Provider Proxy is based on the SAMLv2 specification which states the following:

If an identity provider that receives an <AuthnRequest> has not yet authenticated the presenter or cannot directly authenticate the presenter, but believes that the presenter has already authenticated to another identity provider or a non-SAML equivalent, it may respond to the request by issuing a new <AuthnRequest> on its own behalf to be presented to the other identity provider, or a request in whatever non-SAML format the entity recognizes. The original identity provider is termed the proxying identity provider.

Upon the successful return of a <Response> (or non-SAML equivalent) to the proxying provider, the enclosed assertion or non-SAML equivalent MAY be used to authenticate the presenter so that *the proxying provider can issue an assertion of its own in response to the original <AuthnRequest>, completing the overall message exchange.*

See the complete SAMLv2 specifications at `(http://saml.sml.org/saml-specifications)`

About the OpenSSO Enterprise Identity Provider Proxy

The OpenSSO Enterprise Identity Provider Proxy is designed to enable the following:

- Identity Providers can proxy an authentication request from a Service Provider to a different Identity Provider that has already authenticated the user.
- Multiple Identity Provider Proxies can be configured between the Service Provider and the actual Identity Provider.
- Existing SAMLv2 single sign-on and single logout process flows are seamlessly integrated.
- Users can turn off identity proxying per each connection request. This is done by specifying a special URL parameter `idpproxy=false`.
- Administrators can use customized SPI plug-ins with the Identity Provider Proxy to determine the user's preferred Identity Provider.

OpenSSO Enterprise provides the SPI `com.sun.identity.SAMLv2.profile.SAMLv2IDPProxy` and SPI which enables an administrator to customize the plug-in used to find a preferred identity provider. If the Introduction Cookie is enabled, the Identity Provider Proxy relies on the plug-in to determine the user's preferred Identity Provider. The default implementation of this plug-in interface in OpenSSO Enterprise is based on the Identity Provider Discovery Service. The Identity Provider Discovery Service can help retrieve information about the preferred Identity Provider. The details of this SPI are described in the *Sun OpenSSO Enterprise 8.0 Java API Reference*.

In this first offering of Identity Provider Proxy, the same protocol (for example OASIS SAMLv2 or Liberty ID-FF) must be used for all communications between the participating entities. Participating entities may include service providers, intermediate identity provider proxies, and the actual Identity Provider. However, Identity Provider Proxy is planned to be extended in the future to support heterogeneous environments with multiple identity federation protocols. For example, in the future, Identity Provider Proxy may be used in an environment using SAMLv2 between Service Provider and Identity Provider Proxy. In the same environment, Liberty ID-FF might be used between the Identity Provider Proxy and the actual Identity Provider.

Analyzing the Deployment Architecture

Identity Provider Proxy uses the SAMLv2 protocol to transfer identity data among the communicating entities. The following figure illustrates the major components in a typical deployment using Identity Provider Proxy.

In this deployment, the mobile device user is from France and has an account with Telecom1. The mobile device user travels to the United States and wants to access the global positioning service (GPS) provided by Telecom2 . Telecom2 is a United States service provider . The Telecom2 Identity Provider is the sole identity provider with which Telecom2 has a business affiliation.

Telecom2 receives and processes the authentication requests coming from Telecom2 Global Positioning Service, and responds with the required authentication information. Telecom2, like so many other wireless phone service providers in the world, always maintains the trust relationship with other carriers in different countries. Telecom1 is one such trusted partner which provides roaming services to their customers based on bilateral agreements. In this illustration, because of an established business relationship, Telecom2 doesn't need to know the mobile user at all. Telecom2 can process the authentication request from Telecom2 Global Positioning Service on behalf of Telecom1 based on the following trust relationships:

- Telecom2 Global Positioning Service trusts Telecom2 for user authentication.
- Telecom2 and Telecom1 trust each other for authentication and for the roaming services.

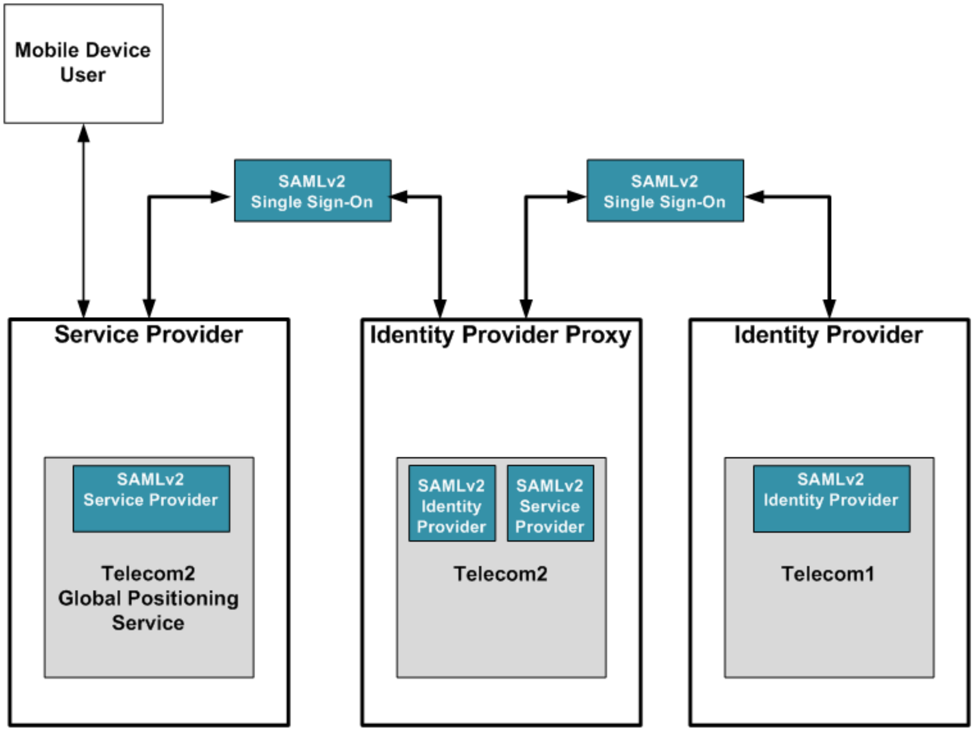


FIGURE 7–1 Deployment Architecture of Identity Provider Proxy

Considering Assumptions, Dependencies, and Constraints

Assumptions and Dependencies

- Both Service Provider and Identity Provider can set up the trust base.
- Service Provider and Identity Provider both achieve single sign-on using the SAMLv2 protocol (persistent and transient).
- Service Provider and Identity Provider must achieve single logout using SAMLv2 protocol.
- The extended configuration metadata define the attributes needed for this feature.
- Required APIs are provided to access the attributes defined in the extended configuration metadata.

Constraints

One protocol such as OASIS SAMLv2 or Liberty ID-FF must be used across all the communications between the participating entities. Participating entities can include Service Provider, intermediate Identity Provider Proxies, and the actual Identity Provider. Currently there is no support for a heterogeneous environment that includes both SAMLv2–compliant systems and non-SAMLv2 equivalents.

Understanding Typical Business Cases

The Identity Provider Proxy feature is designed to be used by two types of users. Administrators configure the SAMLv2 Identity Provider Proxy. End–users access the services provided by service providers that initiate the single sign-on process across different circles of trust.

The following are typical business cases:

- Single Sign-On, Introduction Cookie not enabled
- Single Sign-On, Introduction Cookie enabled
- Single Logout

Single Sign-On, Introduction Cookie is Not Enabled

How the Identity Provider Proxy obtains the information about the actual Identity Provider is determined by whether or not the Introduction Cookie is enabled. Introduction Cookie is turned off, the Identity Provider Proxy retrieves an Identity Provider name from a list of pre-configured Identity Providers specified in the configuration.

The following figure illustrates the process for this use case. In this example, persistent federation is in place. In the transient federation mode, the Identity Provider Proxy does not contain any user information. The Identity Provider Proxy is used for proxying. The user information is only stored in the actual Identity Provider. The following figure illustrates the process flow for this use case.

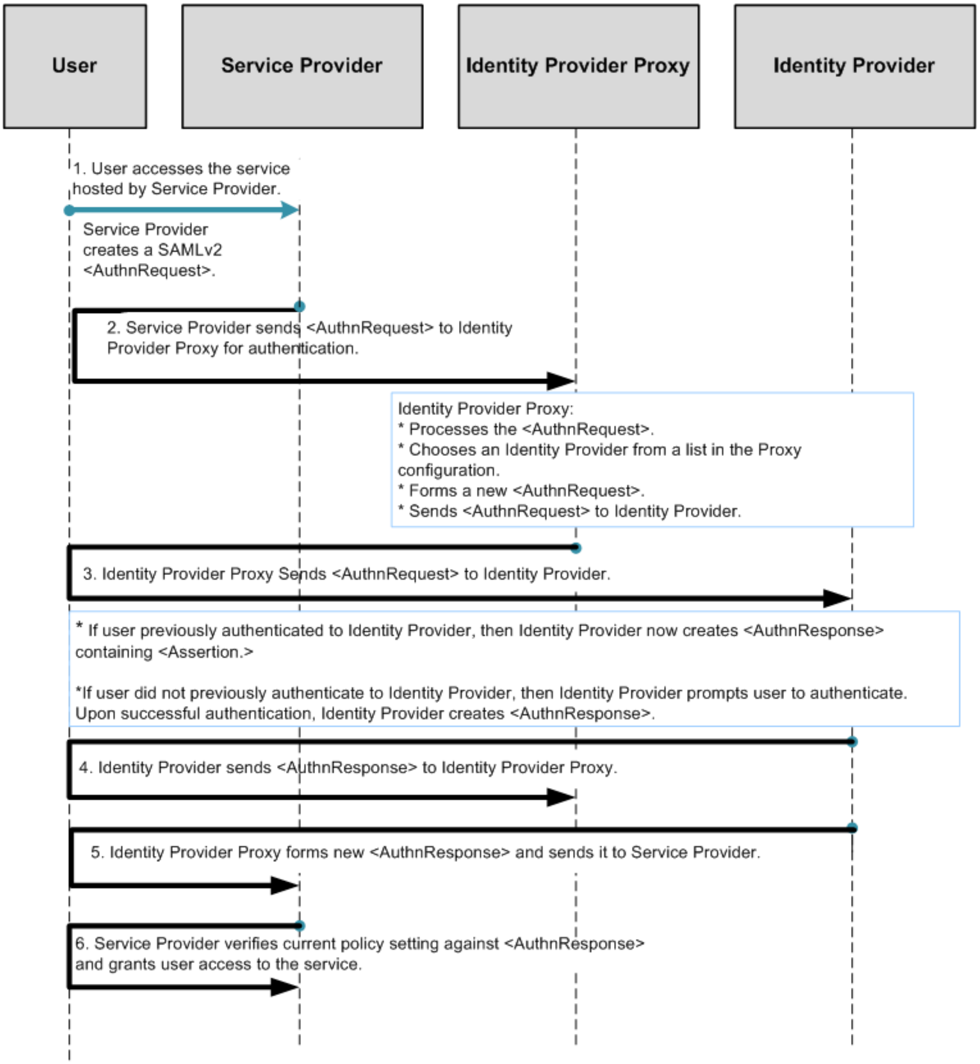
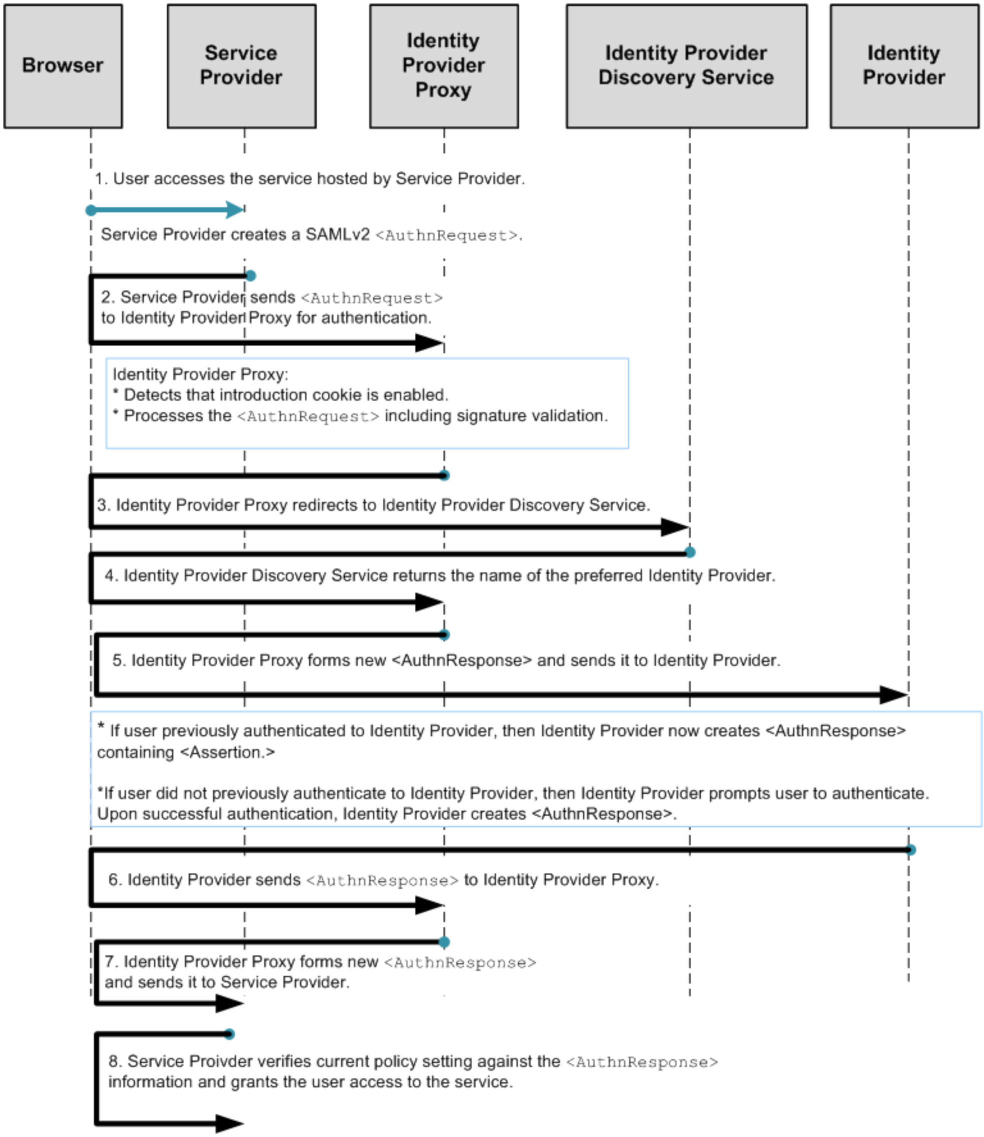

User
Service Provider
Identity Provider Proxy
Identity Provider
1. User accesses the service hosted by Service Provider.
Service Provider creates a SAMLv2 <AuthnRequest>.
2. Service Provider sends <AuthnRequest> to Identity Provider Proxy for authentication.
Identity Provider Proxy:
* Processes the <AuthnRequest>.
* Chooses an Identity Provider from a list in the Proxy configuration.
* Forms a new <AuthnRequest>.
* Sends <AuthnRequest> to Identity Provider.
3. Identity Provider Proxy Sends <AuthnRequest> to Identity Provider.
* If user previously authenticated to Identity Provider, then Identity Provider now creates <AuthnResponse> containing <Assertion.>
*If user did not previously authenticate to Identity Provider, then Identity Provider prompts user to authenticate. Upon successful authentication, Identity Provider creates <AuthnResponse>.
4. Identity Provider sends <AuthnResponse> to Identity Provider Proxy.
5. Identity Provider Proxy forms new <AuthnResponse> and sends it to Service Provider.
6. Service Provider verifies current policy setting against <AuthnResponse> and grants user access to the service.

FIGURE 7–2 Process Flow for Single Sign-On When Introduction Cookie is Not Enabled

Single Sign-On (SSO) with Introduction Cookie Enabled

When the Introduction Cookie is enabled at the Service Provider, the Identity Provider Proxy relies on the `com.sun.identity.SAMLv2.profile.SAMLv2IDPProxy` plug-in to determine the preferred Identity Provider to proxy the authentication request to. The default implementation of this plug-in interface in OpenSSO Enterpris 8.0 is to consult the Identity Provider Discovery Service to get the information about the preferred Identity Provider . The following figure illustrates the process flow for this use case.



1. User accesses the service hosted by Service Provider.

Service Provider creates a SAMLv2 <AuthnRequest>.

2. Service Provider sends <AuthnRequest> to Identity Provider Proxy for authentication.

Identity Provider Proxy:
* Detects that introduction cookie is enabled.
* Processes the <AuthnRequest> including signature validation.

3. Identity Provider Proxy redirects to Identity Provider Discovery Service.

4. Identity Provider Discovery Service returns the name of the preferred Identity Provider.

5. Identity Provider Proxy forms new <AuthnResponse> and sends it to Identity Provider.

* If user previously authenticated to Identity Provider, then Identity Provider now creates <AuthnResponse> containing <Assertion.>

*If user did not previously authenticate to Identity Provider, then Identity Provider prompts user to authenticate. Upon successful authentication, Identity Provider creates <AuthnResponse>.

6. Identity Provider sends <AuthnResponse> to Identity Provider Proxy.

7. Identity Provider Proxy forms new <AuthnResponse> and sends it to Service Provider.

8. Service Proivder verifies current policy setting against the <AuthnResponse> information and grants the user access to the service.

FIGURE 7–3 Process Flow for Single Sign-On (SSO) with Introduction Cookie Enabled

Single SAMLv2 Identity Provider Proxy Logout

The following figure illustrates the process for this use case.

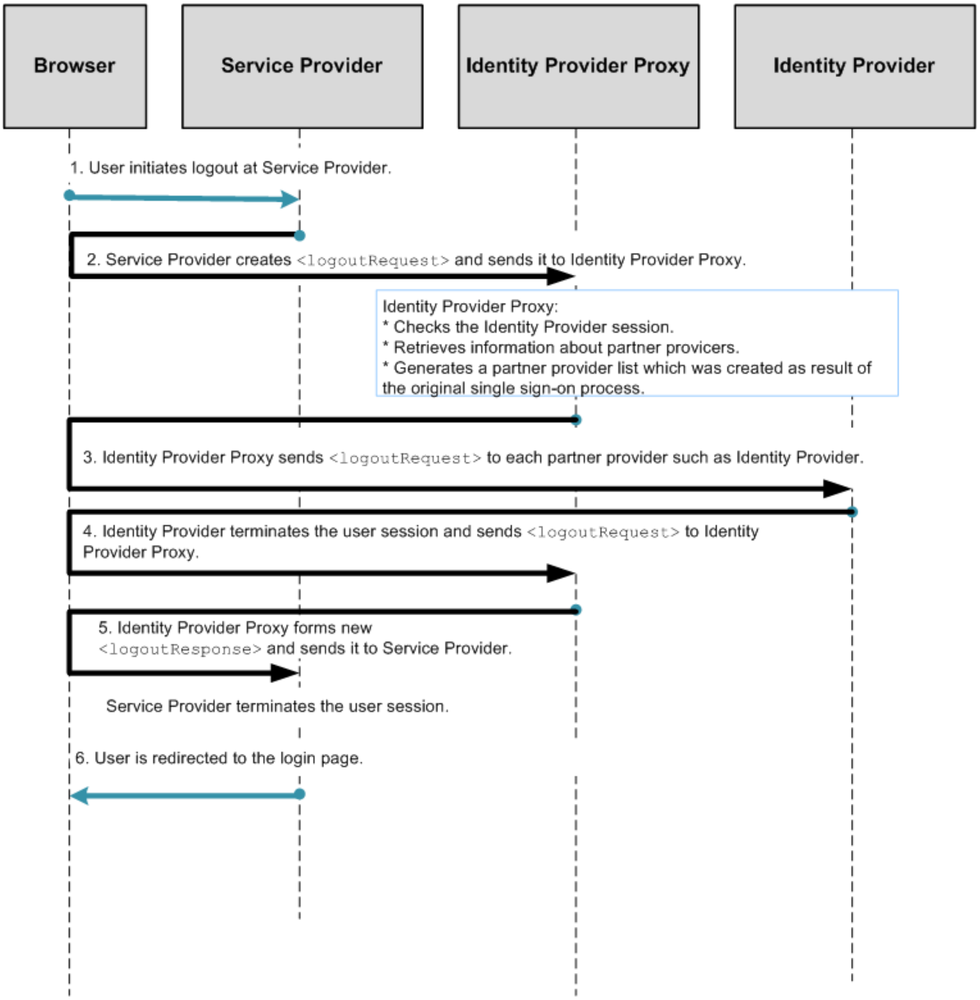

Browser
Service Provider
Identity Provider Proxy
Identity Provider
1. User initiates logout at Service Provider.
2. Service Provider creates <logoutRequest> and sends it to Identity Provider Proxy.
Identity Provider Proxy:
* Checks the Identity Provider session.
* Retrieves information about partner provicers.
* Generates a partner provider list which was created as result of the original single sign-on process.
3. Identity Provider Proxy sends <logoutRequest> to each partner provider such as Identity Provider.
4. Identity Provider terminates the user session and sends <logoutRequest> to Identity Provider Proxy.
5. Identity Provider Proxy forms new <logoutResponse> and sends it to Service Provider.
Service Provider terminates the user session.
6. User is redirected to the login page.

FIGURE 7–4 Process Flow for Single Logout

Setting Up and Configuring SAMLv2 Identity Provider Proxy

The following provides a high-level description of setup and configuration steps. For more detailed instructions, see the *Sun OpenSSO Enterprise 8.0 Administration Guide*.

There is no other software component is required to implement a SAMLv2 Identity Provider Proxy. Everything you need is contained in the OpenSSO Enterprise `fam.war` file.

Setting Up a SAMLv2 Identity Provider Proxy

Install OpenSSO Enterprise instances on three separate host computers, preferably in different domains:

- One OpenSSO Enterprise instance to act as the Service Provider
- At least one OpenSSO Enterprise instance to act as the Identity Provider Proxy.
- One OpenSSO Enterprise instance to act as the actual Identity Provider.

Configuring the SAMLv2 Identity Provider Proxy

You can use the OpenSSO Enterprise administration console or the `ssoadmin` command-line interface to generate and import metadata (steps 3 through 11).

1. Create your own keystore using `keytool`.

   You can also use the `keystore.jks` file created during deployment of OpenSSO Enterprise instance. The `keystore.jks` file is located in the `/opensso/fam` directory. The `keystore.jks` file contains a private key named `test` and an associated public certificate.
2. Encrypt the keystore password for each host machine.
3. Generate Service Provider and Identity Provider metadata.

   In each of the following substeps, save the standard and extended metadata in their respective files.

   a. Generate the Service Provider metadata.
   b. Generate the Identity Provider metadata.
   c. Generate the Identity Provider Proxy metadata.
4. Modify the SAMLv2 extended configuration metadata to include the following:

   `EnabledIDProxy`: The key to turn the SAMLv2 IDP proxy feature on or off.

   `IdpProxyList`: The Identity Providers trusted by the requester (the Service Provider) to authenticate the presenter (the user).

`IdpProxyCount`: The number of proxied misdirected requests permissible between the Identity Provider that receives this `<AuthnRequest>` and the actual Identity Provider that ultimately authenticates the principals. A count of zero means no proxying.

`UseIntroductionForIDPProxy`: When this key is on, the SAMLv2 Introduction Cookie picks a preferred IDP instead of going through the Identity Provider Proxy list.

5. Create a circle of trust on each of the systems.
6. Import the metadata and create the provider entity.

   Specify the name of the circle of trust into where you would like to import the metadata.
7. Import the Service Provider and Identity Provider metadata.
   1. Import the Service Provider metadata to the Identity Provider Proxy.
   2. Import the Identity Provider metadata to the Identity Provider Proxy.
   3. Import the Service Provider portion of the Identity Provider proxy metadata to the Identity Provider.
   4. Import the Identity Provider portion of the Identity Provider Proxy metadata to the Service Provider.
8. As a verification step, on the Service Provider host, log in to the OpenSSO Enterprise administration console and click the Federation tab.

   You should see the profiles for both Service Provider and Identity Provider Proxy.

   Perform the SAMLv2 test cases for single sign-on and single logout through a proxy.

Evaluating Benefits and Tradeoffs

The following may help you determine whether SAMLv2 Identity Provider Proxy is suitable for your environment.

Benefits

- The Identity provider can proxy authentication requests from Service Provider to various Identity Providers to which the user has authenticated.
- Users are granted seamless access to all the available service providers as long as proper trust relationships are established among those Service Providers, Identity Provider Proxies, and the actual Identity Provider.

- Using the SPI implementation, administrators can customize how the preferred Identity Provider is determined.
- End-users can turn off Identity Provider proxying per each connection request.

Tradeoffs

- There is a potential for increased performance overhead.

  Adding intermediaries such as Identity Provider Proxies increase the likelihood of negative impact on overall system performance.
- Using SAMLv2 and non-SAML protocols in the same environment is not currently supported. This can pose a limitation if non-SAML protocols are already in place. However, support for Identity Provider Proxy using multiple protocols is planned for a future release of OpenSSO Enterprise.

◆ ◆ ◆ CHAPTER 8

Using a Multi-Federation Protocol Hub

The Multi-Federation Protocol Hub enables you to use multiple single sign-on protocols such as SAMLv2, IDFF1.2, or WS-Fed together within a circle of trust. In OpenSSO Enterprise 8.0, the Multi-Federation Protocol Hub is supported only at the Identity Provider. The Identity Provider can be configured to support multiple Service Providers, with each Service Provider using a different federation protocol. The session is shared across these heterogeneous protocols, providing single sign-on and single logout within the circle of trust.

The following topics are included in this chapter:

About Identity and Web Services Federation Protocols

Sun OpenSSO identity federation is based on the Liberty Alliance specification which includes the Identity Federation Framework (ID-FF) and SAMLv2 protocols. Microsoft Active Directory Federation Service (ADFS) is based on the Web Services Architecture specification which uses the Microsoft Web Browser Federated Sign-On (MS-MWBF) and Web Services Federation (WS-Federation) protocols. OpenSSO Enterprise provides support for MS-MWBF so that single sign-on can work among OpenSSO and ADFS-based environments. For more information about identity and web service federation protocols, see the *Sun OpenSSO Enterprise 8.0 Technical Overview*.

Analyzing the Deployment

The typical configuration for Multi-Federation Protocol Hub requires one OpenSSO Enterprise instance as an Identity Provider, and two or more OpenSSO Enterprise instances as Service Providers. In this deployment example, the Multi-Federation Protocol Hub is configured with three different Service Providers. Each Service Provider uses a different federation protocol to connect to a single OpenSSO Enterprise instance. Single Logout occurs through a browser redirect when the HTTP post profile is used. If the SOAP binding is used, then a direct SOAP request is sent from the Identity Provider to the Service Provider.

The following illustration illustrates the major components that are involved in the Multi-Federation Protocol Hub.

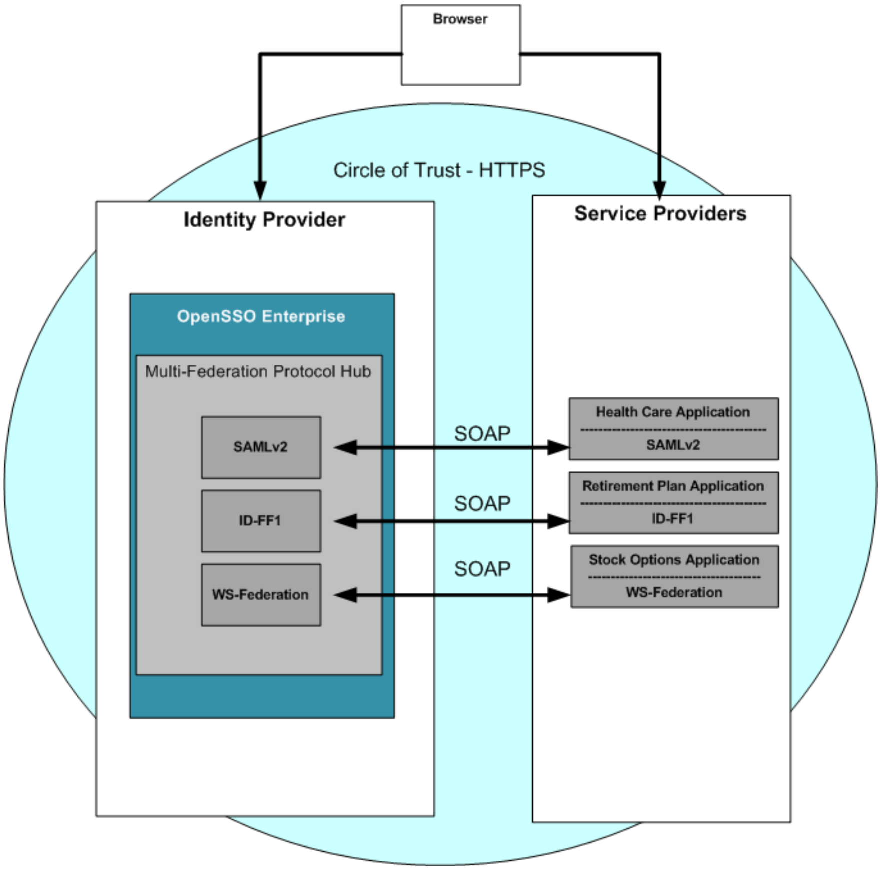


FIGURE 8–1 Deployment Architecture for the Multi-Federation Protocol Hub

Considering Assumptions, Dependencies, and Constraints

The following are issues you must resolve before choosing to use the Multi-Federation Protocol Hub.

Constraints

The Multi-federation Protocol Hub in OpenSSO Enterprise can be configured with only the following federation protocols:

- SAMLv2
- Liberty ID-FF1
- WS-Federation

The Multi-Federation Protocol Hub in Open SSO Enterprise 8.0 is only supported on the Identity Provider configuration.

Assumptions and Dependencies

- Single Logout over HTTP post profile must be supported by the Service Providers running HTTP.
- Single Logout over SOAP must be supported by the Service Providers running SOAP.

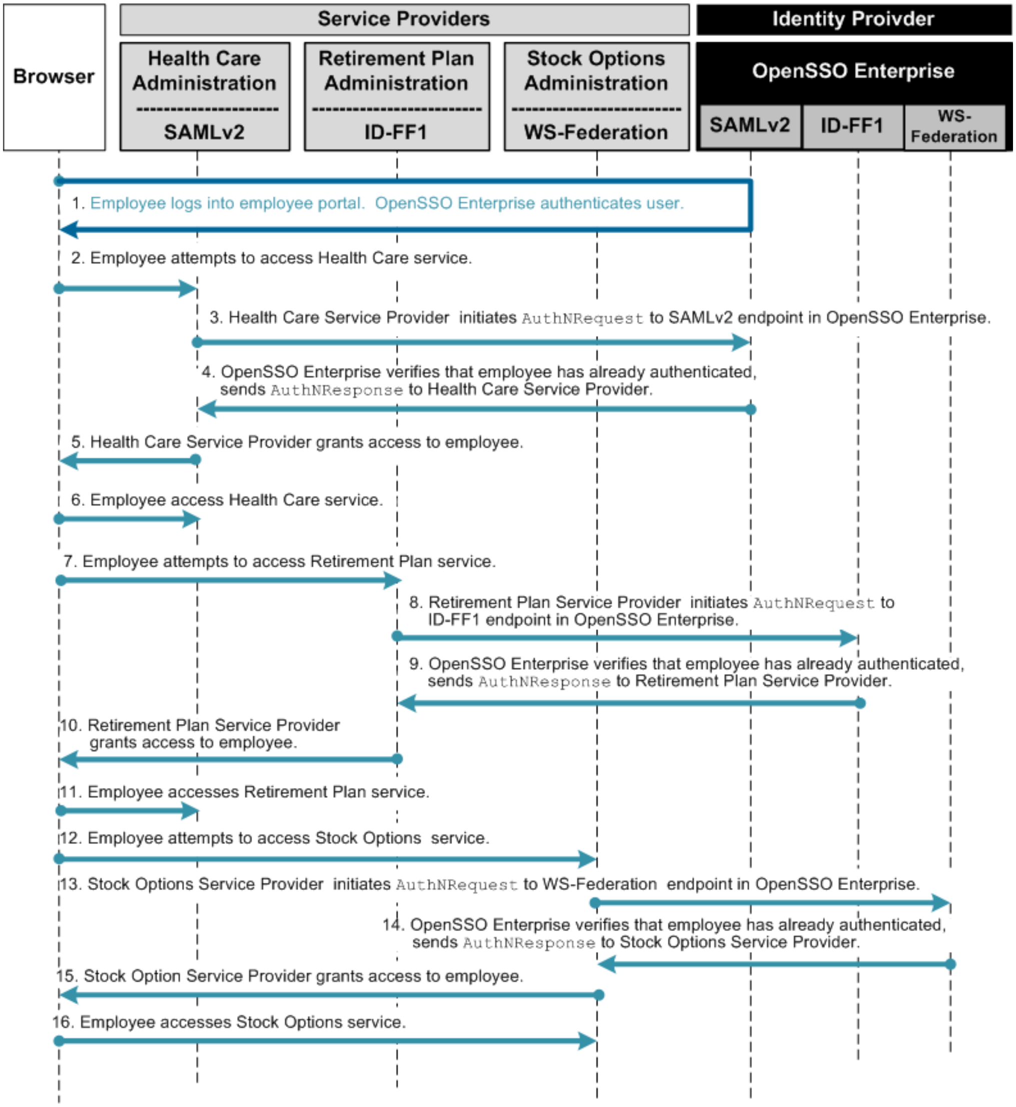
Understanding Typical Business Use Cases

A company uses the following services and federation protocols to manage employee benefits:

- Health Care Administration (SAMLv2)
- Retirement Plan Administration (ID-FF1)
- Stock Plan Administration (WS-Federation)

The company itself acts as an Identity Provider, managing employee information in its corporate user database. The Identity Provider enables employees to access any of the three Service Providers through an employee portal. The Health Care Service Provider uses the SAMLv2 federation protocol. The Retirement Plan Service Provider uses ID-FF1, and the Stock Plan Service Provider uses WS-Federation. The Identity Provider is configured as a Multi-Federation Protocol Hub and provides single sign-on and single logout across all these services.

The following figures illustrates a typical Multi-Federation Protocol Hub process flow.



1. Employee logs into employee portal. OpenSSO Enterprise authenticates user.

2. Employee attempts to access Health Care service.

3. Health Care Service Provider initiates `AuthNRequest` to SAMLv2 endpoint in OpenSSO Enterprise.

4. OpenSSO Enterprise verifies that employee has already authenticated, sends `AuthNResponse` to Health Care Service Provider.

5. Health Care Service Provider grants access to employee.

6. Employee access Health Care service.

7. Employee attempts to access Retirement Plan service.

8. Retirement Plan Service Provider initiates `AuthNRequest` to ID-FF1 endpoint in OpenSSO Enterprise.

9. OpenSSO Enterprise verifies that employee has already authenticated, sends `AuthNResponse` to Retirement Plan Service Provider.

10. Retirement Plan Service Provider grants access to employee.

11. Employee accesses Retirement Plan service.

12. Employee attempts to access Stock Options service.

13. Stock Options Service Provider initiates `AuthNRequest` to WS-Federation endpoint in OpenSSO Enterprise.

14. OpenSSO Enterprise verifies that employee has already authenticated, sends `AuthNResponse` to Stock Options Service Provider.

15. Stock Option Service Provider grants access to employee.

16. Employee accesses Stock Options service.

FIGURE 8–2 Process Flow for Single Sign-On Using a Multi-Federation Protocol Hub

The following figure illustrates the process flow for Single Logout using the Multi-Federation Protocol Hub.

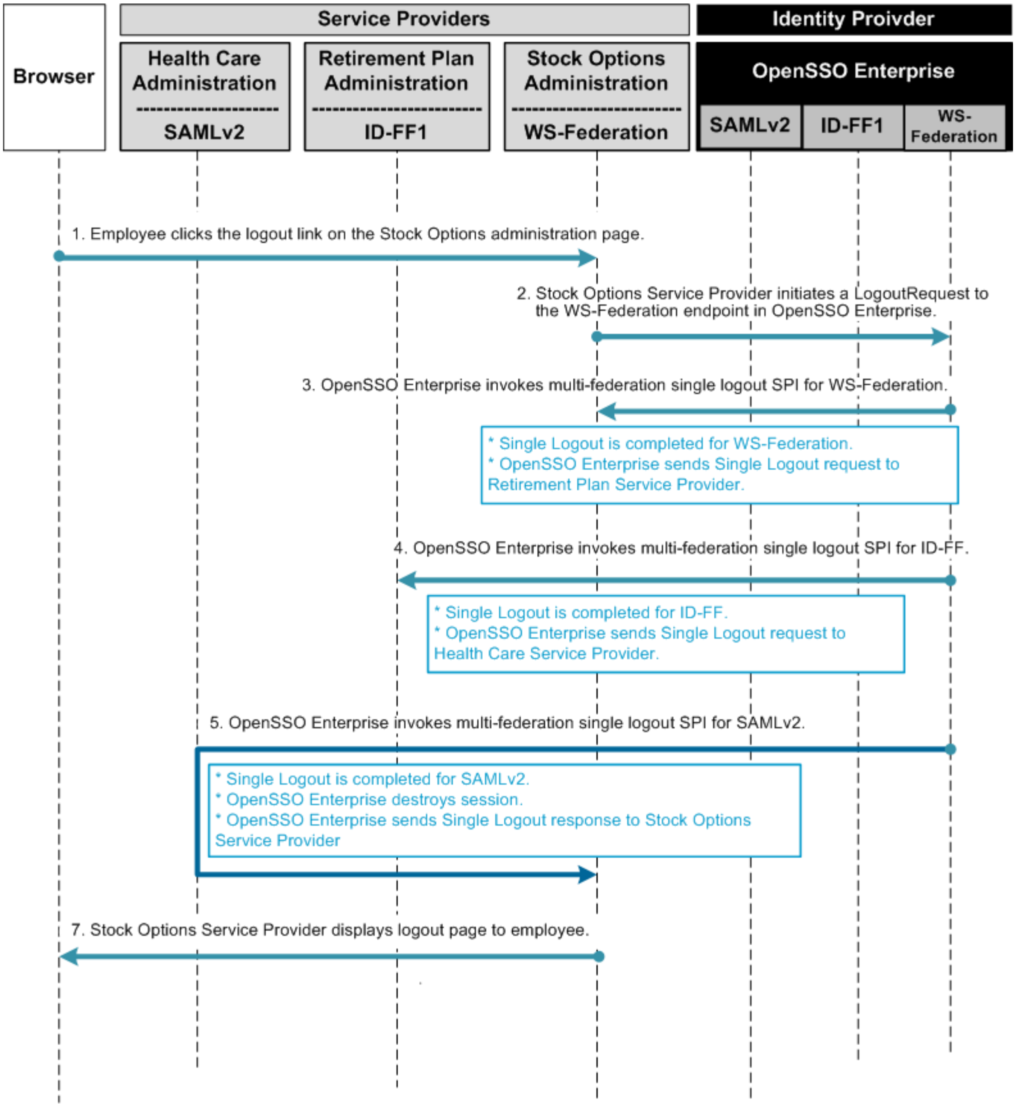

Browser
Service Providers
Health Care Administration
SAMLv2
Retirement Plan Administration
ID-FF1
Stock Options Administration
WS-Federation
Identity Proivder
OpenSSO Enterprise
SAMLv2
ID-FF1
WS-Federation
1. Employee clicks the logout link on the Stock Options administration page.
2. Stock Options Service Provider initiates a LogoutRequest to the WS-Federation endpoint in OpenSSO Enterprise.
3. OpenSSO Enterprise invokes multi-federation single logout SPI for WS-Federation.
* Single Logout is completed for WS-Federation.
* OpenSSO Enterprise sends Single Logout request to Retirement Plan Service Provider.
4. OpenSSO Enterprise invokes multi-federation single logout SPI for ID-FF.
* Single Logout is completed for ID-FF.
* OpenSSO Enterprise sends Single Logout request to Health Care Service Provider.
5. OpenSSO Enterprise invokes multi-federation single logout SPI for SAMLv2.
* Single Logout is completed for SAMLv2.
* OpenSSO Enterprise destroys session.
* OpenSSO Enterprise sends Single Logout response to Stock Options Service Provider
7. Stock Options Service Provider displays logout page to employee.

FIGURE 8–3 Process Flow for Single Logout Using the Multi-Federation Protocol Hub

Setting Up and Configuring a Multi-Federation Protocol Hub

The following information provides a high-level overview of setup and configuration instructions. Before you can begin, OpenSSO Enterprise must be deployed on a supported web container. You can configure the Multi-Federation Protocol Hub by importing the metadata using either the OpenSSO Enterprise administration console, or using the `ssoadmin` command-line interface. For detailed configuration steps, see the *Sun OpenSSO Enterprise 8.0 Administration Guide*. A code JSP file is contained in theopensso.war file. The sample JSP demonstrates how to configure a Multi-Federation Protocol Hub.

Using the Sample JSP

1. Install and deploy OpenSSO Enterprise instances on four separate host computers, one instance in each domain.
2. Locate the sample JSP on the Open SSO instance.

   `http://FQDN/opensso/samples/multiprotocol/index.html`
3. Configure OpenSSO Enterprise instance 1 as a SAMLv2 Service Provider named SP1.

   Run the sample JSP to create one hosted SAMLv2 Service Provider and one remote SAMLv2 Identity Provider in the same circle of trust.
4. Configure OpenSSO Enterprise instance 2 as an ID-FF Service Provider named Service Provider 2.

   Run the sample JSP to create one host ID-FF Service Provider and one remote ID-FF Identity Provider in the same circle of trust.
5. Configure OpenSSO Enterprise instance 3 as a WS-Federation Service Provider named Service Provider 3.

   Run the sample JSP to created one hosted WS-Federation Service Provider and one remote WS-Federation Identity Provider in one circle of trust.
6. Configure OpenSSO Enterprise instance 4 as an Identity Provider using the following protocols: IDP, referred as IDP1, IDP2 and IDP3 respectively.
   - SAMLv2 (Identity Provider 1)
   - ID-FF (Identity Provider 2)
   - WS-Federation (Identity Provider 3)

   Run the sample JSP to create three hosted Identity Providers (one each for SAMLv2, ID-FF and WS-Federation), and three remote Service Providers (one each for SAMLv2, ID-FF and WS-Federation) the same circle of trust.
7. Run single sign-on from Service Provider 1 to Identity Provider 2, then from Service Provider 2 to Identity Provider 2 without logging in again, then Service Provider 3 to Identity Provider 3 without logging in.
8. Run single logout from Service Provider 1.

   All sessions on Service Provider 2, Service Provider 3, and on all Identity Providers are destroyed.
9. Run single sign-on again, and then run single Logout from Identity Provider 1.

   All sessions on Service Provider 1, Service Provider 2, Service Provider 3, and on Identity Providers are destroyed.

Evaluating Benefits and Tradeoffs

The Multi-Federation Protocol Hub feature in OpenSSO Enterprise enables the Identity Provider to integrate with any existing or future service provider or service partner. The Multi-Federation Protocol Hub achieves single sign-on and single Logout regardless of which federation protocol the service provider or partner uses. Without this feature, the Identity Provider has to force the Service Providers to use a single federation protocol.

With the Multi-Federation Protocol Hub only one circle-of-trust is required when using heterogeneous Service Providers and Identity Providers in the same circle of trust. Without this feature, you must set up and configure multiple circles of trust, one for each federation protocol used. The Identity Provider could require multiple OpenSSO Enterprise instances. Each OpenSSO Enterprise would have to act as an Identity Provider, and each OpenSSO Enterprise instance would require a different protocol. To achieve single sign-on and single Logout, you would have to install some kind of intelligent proxy in front of the Identity Provider. The proxy would have to be able to recognize the incoming protocol from the Service Provider, and route the request to the correct Identity Provider instance accordingly.

The Multi-Federation Protocol Hub configuration steps are simple. The only configuration required is one extra metadata file for each protocol to be supported by the Identity Provider.

CHAPTER 9

Enabling Web Services Federation Between Active Directory Federation Service and OpenSSO Enterprise

Sun OpenSSO identity federation is based on the Liberty Alliance specification which includes uses the Identity Federation Framework (ID-FF) and SAMLv2 protocols. Microsoft Active Directory Federation Service (ADFS) is based on the Web Services Architecture specification which uses the Microsoft Web Browser Federated Sign-On (MS-MWBF) and Web Services Federation (WS-Federation) protocols.

OpenSSO Enterprise provides support for MS-MWBF so that single sign-on can work among OpenSSO and ADFS-based environments. This interoperability is achieved by creating trust relationships between different security realms, and exchanging security tokens using the Web Services Federation protocol.

This chapter provides information about enabling web services federation between ADFS-based and OpenSSO Enterprise. The following topics are contained in this chapter:

Analyzing the Deployment Architecture

This deployment consists of two different environments:

- An ADFS-based environment
- A load-balanced, multi-server OpenSSO Enterprise environment

This deployment illustrates the interoperability between both environments, and also illustrates the added constraints of a multi-server OpenSSO Enterprise solution.

The ADFS environment is derived entirely from `http://Step-by-Step Guide for Active Directory Federation Services`. In this deployment, a web browser (client) interacts with a web resource to request a security token from a requestor Identity Provider or Security Token Service. The request is communicated through a resource partner such as an Identity Provider or Security Token Service.

OpenSSO Enterprise can play the role of either resource (Service Provider) or requestor (Identity Provider). The following figure illustrates OpenSSO Enterprise acting as a Service Provider, known in the MS-MWBF specification as a Requestor Identity Provider/Security Token Service (Requestor IP/STS). The business use case for this architecture is described in "OpenSSO Enterprise Acts as Service Provider" on page 122.

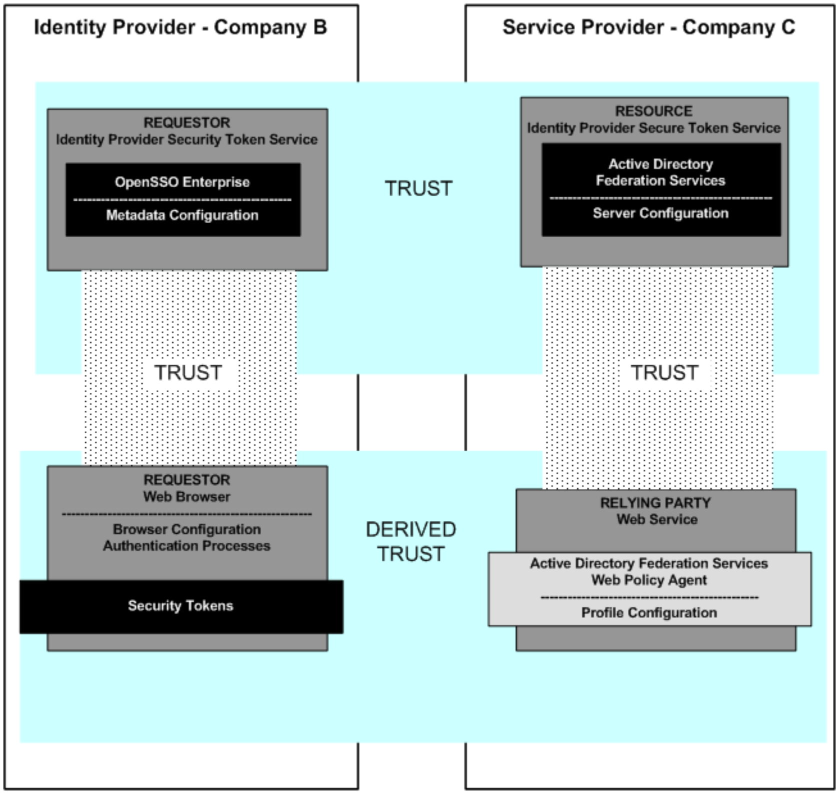


FIGURE 9–1 Deployment Architecture for ADFS Integration with OpenSSO Enterprise Acting as Service Provider

The following figure illustrates OpenSSO Enterprise acting as a Service Provider, known in the MS-MWBF specification as a Resource Identity Provider/Security Token Service (Resource IP/STS). The business use case for this architecture is described in "OpenSSO Enterprise Acts as Identity Provider" on page 123.

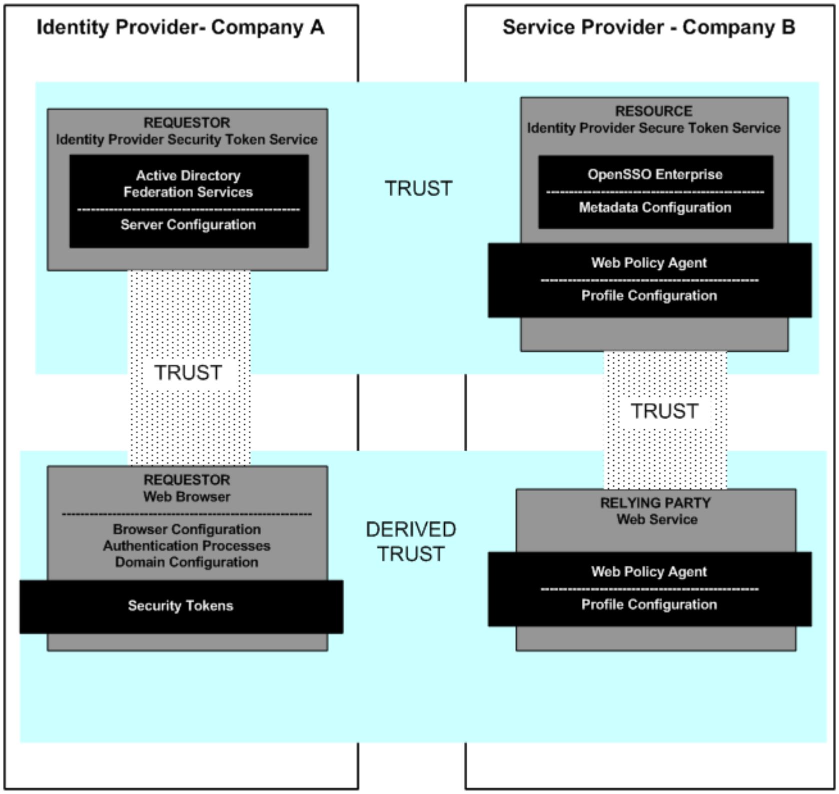


FIGURE 9–2 Deployment Architecture for ADFS Integration withOpenSSO Enterprise Acting as Service Provider

Considering Assumptions, Dependencies, and Constraints

The following are issues you must resolve before you can enable Web Service Federation among ADFS and OpenSSO Enterprise.

Assumptions and Dependencies

- The ADFS-based environment is already set up and running.

This chapter is based on the assumption that you are proficient in setting up and ADFS-based environment as described in

- OpenSSO Enterprise is already deployed.

  This is a prerequisite for configuring OpenSSO Enterprise to act as an Identity Provider or Service Provider in a circle of trust.

Constraints

- The configuration requirements documented in this chapter include tools and procedures that are not appropriate for a production deployment.

  Examples are: the use of self-signed certificates, the modification of host files, local time synchronization, and so forth. These are described for illustration purposes only. For production deployments, you must use different solutions suitable for your environment.
- OpenSSO Enterprise supports WS-Federation as it relates to its support within the ADFS boundaries. Parts of the WS-Federation specification not required by ADFS may not be supported in this release.

Understanding Typical Business Use Cases

This chapter describes two typical business use cases:

- "OpenSSO Enterprise Acts as Service Provider" on page 122
- "OpenSSO Enterprise Acts as Identity Provider" on page 123

OpenSSO Enterprise Acts as Service Provider

In this use case, Company A is acquired by Company B. The intranets for both companies have been merged, but much of the network infrastructure remains as though they were still two separate entities. Company A maintains an Active Directory domain, and Company B maintains an OpenSSO Enterprise single sign-on infrastructure in its own domain.

In order for Company A employees to access some internal applications available to Company B employees, a trust relationship is created between the Company A domain and the Company B domain. The trust relationship is created using the Web Services Federation protocol. Company A employees, signed on to their Microsoft Windows computers, can now navigate to the Company B paycheck application by using a Web Services Federation secure token.

OpenSSO Enterprise Acts as Identity Provider

In this use case, Company B wants to offer its employees a new online collaborative environment based on Microsoft SharePoint Services. The collaboration solutions is an outsourced model where Company C provides dedicated SharePoint Services to its customers. In order to provide single sign-on to the Company B employees, Company C leverages the federation services provided by ADFS. A trust relationship is created between created between the Company B OpenSSO Enterprise Identity Provider and the Company C Resource Identity Provider /Security Token Service.

Setting up and Configuring Single Sign-On Among OpenSSO Enterprise and ADFS Environments

This chapter provides high-level deployment information for setting up and configuring single sign-on among OpenSSO Enterprise and ADFS environments. For detailed information, see the *Microsoft Administering Active Directory Federation Services Guide*`http://technet.microsoft.com/en-us/library/cc736337.aspx` for instructions on configuring Resource and Account Partners. See the *Sun OpenSSO Enterprise 8.0 Administration Guide* for information about using the `ssoadmin` command or administration console to generate metadata, create a circle of trust and import entities.

Enabling WS-Federation between an ADFS environment and an OpenSSO Enterprise environment involves exchanging metadata to enable a trust relationship. Prior to this, the following requirements must be met:

- All communications between WS-Federation components must be made over SSL.

  ADFS does not perform an HTTP POST of a WS-Federation RSTR to a non-HTTPS URL.
- Name resolution is based on host files. Therefore, host files must be appropriately updated with host names and IP addresses.
- The ADFS environment can rely on DNS.
- All servers must be time-synchronized.

  This is essential to proper token validation.
- Token signing certificates must be created and imported for both ADFS and OpenSSO Enterprise endpoints.

  This process is automated in ADFS, but requires the use of the `keytool` command for OpenSSO Enterprise.

The creation of a trust relationship relies on the exchange of metadata between the parties involved. Importing this information is straightforward and can be done through the GUI on the ADFS side. On the OpenSSO Enterprise side, to import the information you can use the `ssoadmin` command-line utility or the `ssoadmin.jsp`.

Configuring OpenSSO Enterprise to Act as a Service Provider

This use case requires that the ADFS server in the Company B domain be configured to recognize the Company A OpenSSO Enterprise endpoint as a Resource Partner. The Company B ADFS server must be recognized as a valid Identity Provider in a circle of trust that includes the Company A OpenSSO Enterprise server as a Service Provider.

In the OpenSSO Enterprise environment:

Use the ADFS snap-in to create a new Resource Partner. The new Resource Partner must be defined using the proper name and endpoint URL.

In the ADFS-based environment:

- Create metatdata and extended metadata files to define the Company B ADFS server as the Identity Provider, and the Company A OpenSSO Enterprise server as the Service Provider in a WS-Federation protocol paradigm.
- Create a new circle of trust and import each Identity Provider and Service Provider to belong to this circle of trust.

This configuration currently works only if a user account with the same UPN is created in both the ADFS domain and the OpenSSO Enterprise server. This is a major constraint.

Configuring OpenSSO Enterprise to Act as an Identity Provider

This use case requires that the ADFS server in the Company C domain to be configured to recognize the Company A server as an Account Partner. The Company A server must be configured to recognize the Company C ADFS server as a Service Provider in a circle of trust.

In the OpenSSO Enterprise environment:

- Configure a new keystore for the token signing certificate, or leverage the one provided by the container.
- Create metadata and extended metadata files to define the Company A OpenSSO Enterprise server as the Identity Provider.
- Create metadata and extended metadata files to define the Company B ADFS server as the Identity Provider, and the Company C ADFS server as the Resource Provider in a WS-Federation protocol paradigm.
- Create a new circle of trust, and import each Identity Provider and Service Provider to belong to this new circle of trust.

In the ADFS environment:

- Create a new Account Partner using the ADFS snap-in.

  The proper name and endpoint URL must be defined.
- Import the OpenSSO Enterprise token signing certificate (DER format). For detailed information, see the .*Sun OpenSSO Enterprise 8.0 Administration Guide*

Evaluating Benefits and Tradeoffs

The following information helps you decide whether enabling Web Services Federation between ADFS and OpenSSO Enterprise is suitable for your needs.

Benefits

Using OpenSSO Enterprise as Service Provider

You are likely to leverage WS-Federation in a mixed environment involving Windows domains and heterogeneous web service environments. In such cases, using WS-Federation eliminates the need to complete the complicated setups involved with Desktop SSO (IWA, Kerberos, etc.). This simplifies the integration of web services in the ADFS-based environments.

Using OpenSSO Enterprise as Identity Provider

The immediate benefit is the single sign-on to SharePoint Services from non-ADFs environments. This can be extended to pure claims-based applications residing inside the Resource Partner's domain.

Tradeoffs

The main drawback to using WS-Federation is that currently only limited support or configuration help is offered for ADFS claims within OpenSSO Enterprise. For example, the *Microsoft Administering Active Directory Federation Services Guide*`http://technet.microsoft.com/en-us/library/cc736337.aspx` depicts the use of group claims and their mapping between realms. The use of group claims eliminates the need to map user principals information from one realm to the next in a federated environment. These claims, based on group memberships, have not been tested in this deployment example configuration.

Finding More Information

Specifications

- *Microsoft Web Browser Federated Sign-on Protocol Specification*

  `http://msdn.microsoft.com/en-us/library/cc236471.aspx`
- *Web Services Federation Language*

  `http://specs.xmlsoap.org/ws/2006/12/federation/wsfederation.pdf`

Guides and Overviews

- *Understanding WS-Federation*

  `http://msdn.microsoft.com/en-us/library/bb498017.aspx`
- *Overview of ADFS*

  `http://download.microsoft.com/download/d/8/2/d827389e-760a-40e5-a68a-4e75723998c5/ADFS_Overview.doc`
- *Microsoft Administering Active Directory Federation Services Guide*

  `http://technet.microsoft.com/en-us/library/cc736337.aspx`

Case Study

OpenSSO, WS-Federation & IBM DataPower by Joachim Andre

`https://opensso.dev.java.net/files/documents/3676/79106/OpenSSO-WS-Fed-DataPower-FederationPoC.pdf`

CHAPTER 10

Securing Web Services Using ID-WSF (Liberty Alliance Specifications)

This chapter provides information about developing identity-based web services using the Identity Web Services Framework (ID-WSF) developed in OpenSSO Enterprise. OpenSSO Enterprise provides an implementation for the Liberty Personal Profile Service and typically uses the default OpenSSO Enterprise configuration store for the Personal Profile Service configuration. However, this chapter describes a new use case. With OpenSSO Enterprise 8.0, you can develop a custom, identity-based web service using the OpenSSO Enterprise infrastructure instead of using the Personal Profile Service as a data service.

The following topics are contained in this chapter:

About the Identity Web Services Framework

Identity Web Services Framework (ID-WSF) is a set of specifications created by the Liberty Alliance to promote secure web services. ID-WSF is part of Liberty's Phase 2 specifications which augment the initial Identity Federation Framework (ID-FF) specifications. The ID-FF focuses on federating the user's authentication and single sign-on. The ID-WSF defines specifications for web services in a federated environment. The federated environment establishes a trust between all the participating entities without revealing the end user's identity. The following diagram illustrates the relationship between entities in such an environment, known as a circle of trust.

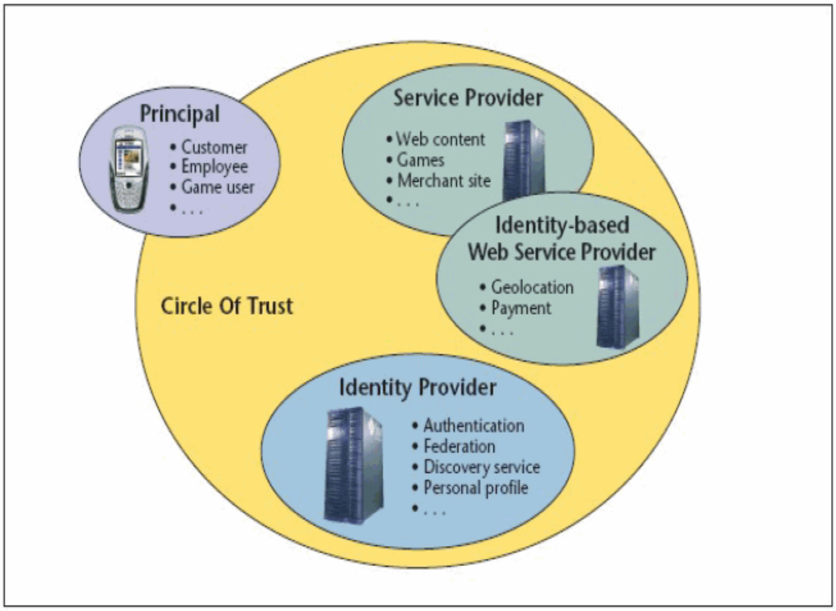


FIGURE 10–1 Major Components in a Circle of Trust

The ID-WSF defines specifications for the following Liberty components:

- Discovery Service
- SOAP Binding
- Authentication Service
- Security Mechanisms
- Interaction Service
- PAOS Binding
- Data Service Template

The ID-WSF defines a concrete trust authority called the Liberty Discovery Service. The framework is built around the Liberty Discovery Service. The Liberty Discovery Service not only enables a web service to register, but also helps in generating security credentials for web service clients that may be doing lookups for a specific service type.

The Liberty Interaction Service allows the user to interact during web services communication for any authorization. Liberty Authentication Service allows the web services clients to authenticate the principal in non-browsed based environments. As part of the Phase 2 specifications, Liberty Alliance also defined specifications for additional identity services such as Liberty Personal Profile Service, Liberty Employee Personal Profile Service and others. These Phase 2 specifications form the Liberty Service Instance Specifications (SIS) Specifications.

OpenSSO Enterprise fully implements ID-WSF1.x specifications and exposes the ID-WSF as a pluggable framework which the user can leverage for increased security. OpenSSO Enterprise is a self-contained, single WAR file built to industry standard specifications. The Identity Web Services Framework in OpenSSO Enterprise enables developers to focus on the business logic of their service while leaving the security aspect to OpenSSO Enterprise.

The following diagram represents the OpenSSO Enterprise ID-WSF framework from an implementation point of view.

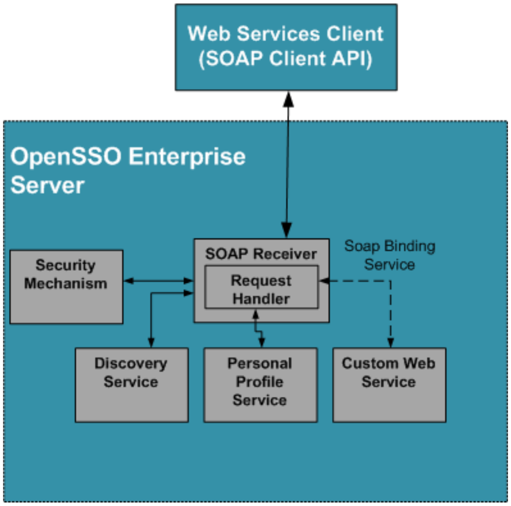


FIGURE 10–2 OpenSSO Enterprise Identity Web Service Framework Implementation

The OpenSSO Enterprise ID-WSF uses a simple servlet-based extension framework that any identity based web service can plug into. OpenSSO Enterprise provides tools and APIs for two distinct classes of developers: Identity Web Service Implementors (WSPs), and application developers (WSCs) who use these web services . OpenSSO Enterprise also provides the hooks required to quickly integrate the existing infrastructure with Liberty-enabled infrastructure.

Any custom web service that is developed using the OpenSSO Enterprise ID-WSF must register with the SOAP Binding Service. The SOAP Binding Service provides the validation of SOAP message with respect to security such as XML Digital Signature and Encryption and other Liberty processing rules. The Soap Binding Service then generates the OpenSSO Enterprise single sign-on token for further authorization of the client by the web service.

Analyzing the Deployments

The Identity Web Services Framework can be used in either browser-based mode or desktop mode. The browser-based client uses SAMLv2 or ID-FF protocols to find the Liberty Discovery Service Resource Offering of an authenticated user, or bootstrap, into the ID-WSF framework. The desktop clients can leverage the Liberty Authentication Service to bootstrap into the ID-WSF. OpenSSO Enterprise supports ID-WSF boot strapping through the SAMLv2, ID-FF, or ID-WSF -based authentication service, depending upon the client needs. This chapter describes deployments for both browser-based clients and desktop clients

Browser-based ID-WSF Deployment

The following diagram represents the deployment architecture for ID-WSF using OpenSSO Enterprise for browsed-based clients.

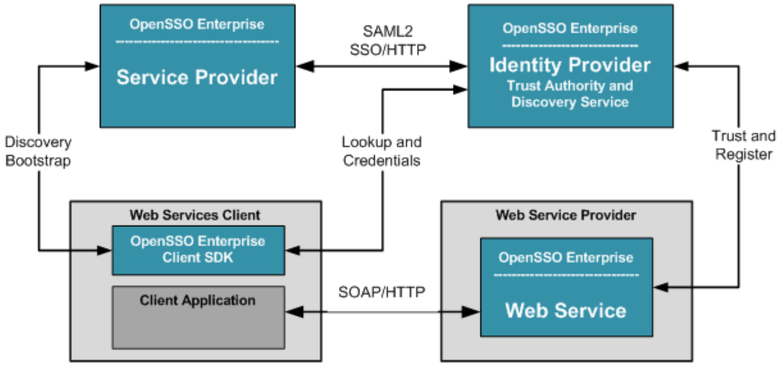


FIGURE 10–3 Deployment Architecture for Browser-based Identity Web Services

Both Service Provider and Identity Provider are used for authenticating the user's identity using SAMLv2 protocols. OpenSSO Enterprise can be an Identity Provider or a Service Provider or a hosting web service in this deployment. The Service Provider and Web Services Client are in the same domain in this deployment. The Web Service Provider registers its service resource offering with the Discovery Service before it offers services to various clients. The registration can be done through either the Discovery Service protocol or out of band. The OpenSSO Enterprise can be deployed in various roles for this deployment as illustrated in the deployment architecture diagram. The following figures shows the process flow among various entities in the browser-based identity web services deployment.

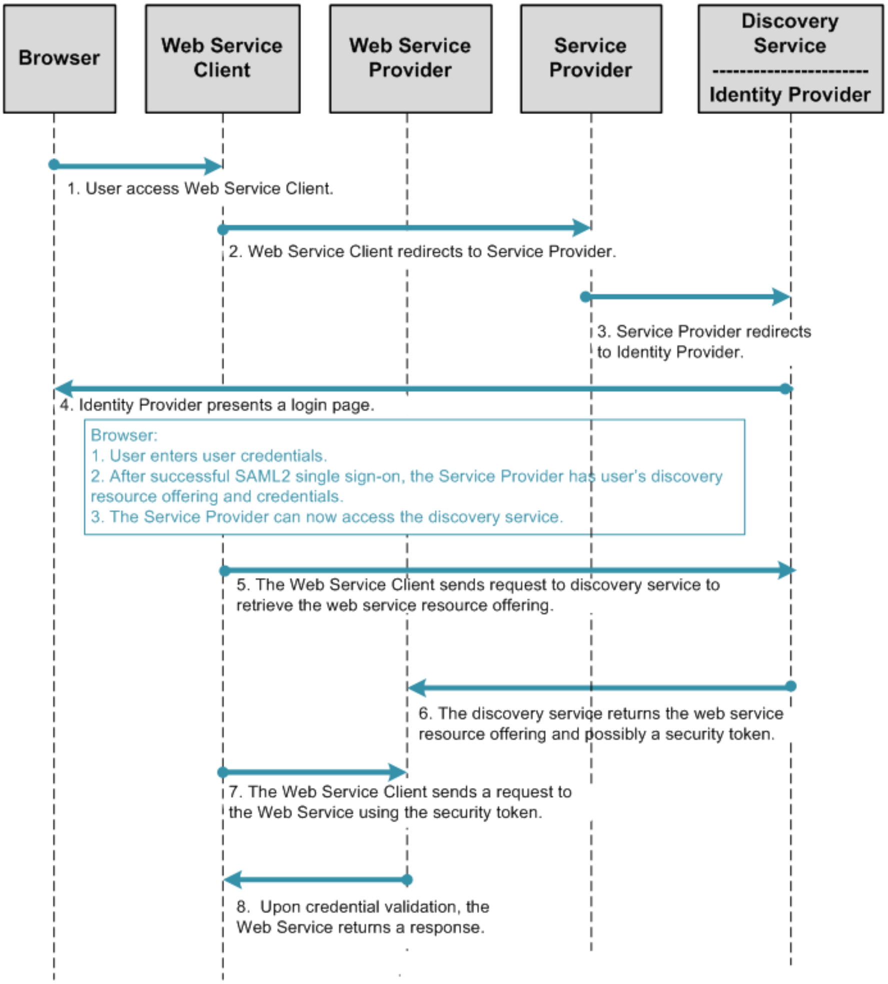

Browser
Web Service Client
Web Service Provider
Service Provider
Discovery Service
Identity Provider
1. User access Web Service Client.
2. Web Service Client redirects to Service Provider.
3. Service Provider redirects to Identity Provider.
4. Identity Provider presents a login page.
Browser:
1. User enters user credentials.
2. After successful SAML2 single sign-on, the Service Provider has user's discovery resource offering and credentials.
3. The Service Provider can now access the discovery service.
5. The Web Service Client sends request to discovery service to retrieve the web service resource offering.
6. The discovery service returns the web service resource offering and possibly a security token.
7. The Web Service Client sends a request to the Web Service using the security token.
8. Upon credential validation, the Web Service returns a response.

FIGURE 10–4 Process Flow for Browser-based Identity Web Services

Desktop ID-WSF Deployment

The desktop mode ID-WSF deployment supports desktop mode clients, so they do not require an application container. Desktop ID-WSF is useful for standalone Web Service Clients. The custom Web Service Clients can leverage the OpenSSO Enterprise Client SDK to enable a secure connection for a given payload. The Web Services Client bundles the OpenSSO Enterprise Client SDK that accesses the local OpenSSO Enterprise instance to secure web service requests.

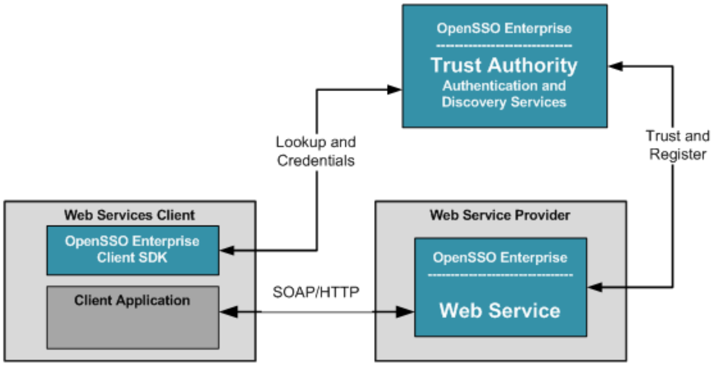


FIGURE 10–5 Desktop ID-WSF Deployment

The Web Services package is contained in the OpenSSO Enterprise WAR file and must be deployed along with the OpenSSO Enterprise server to leverage the ID-WSF security framework. The configuration and user data is not required to be same data store as that of OpenSSO Enterprise. The trust authority for ID-WSF is the Discovery Service. The Discovery Service end point is exposed the same way as any data web service, and can fully leverage OpenSSO Enterprise infrastructure components such as authentication, policies, and so forth to serve web service clients and as well as web services. OpenSSO Enterprise is independent in all these roles and can be deployed appropriately based on the customer or application requirements. The following figure illustrates the process flow for desktop ID-WSF.

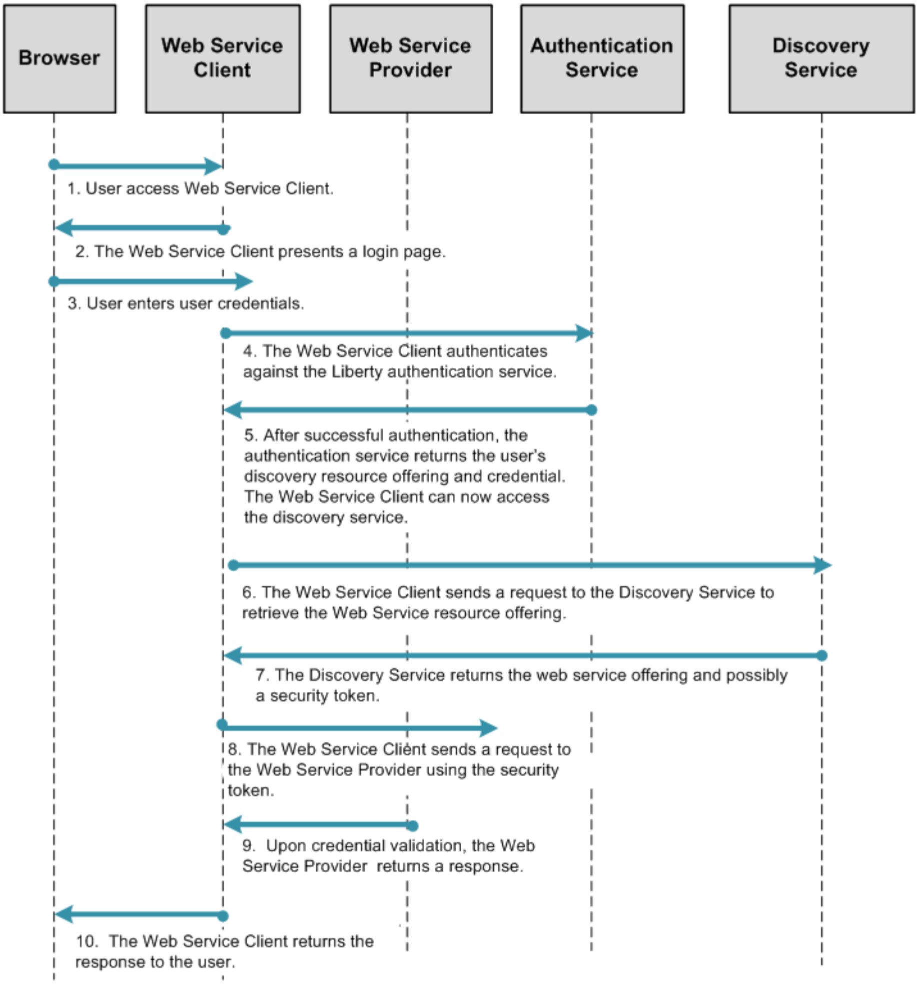

Browser
Web Service Client
Web Service Provider
Authentication Service
Discovery Service
1. User access Web Service Client.
2. The Web Service Client presents a login page.
3. User enters user credentials.
4. The Web Service Client authenticates against the Liberty authentication service.
5. After successful authentication, the authentication service returns the user's discovery resource offering and credential. The Web Service Client can now access the discovery service.
6. The Web Service Client sends a request to the Discovery Service to retrieve the Web Service resource offering.
7. The Discovery Service returns the web service offering and possibly a security token.
8. The Web Service Client sends a request to the Web Service Provider using the security token.
9. Upon credential validation, the Web Service Provider returns a response.
10. The Web Service Client returns the response to the user.

FIGURE 10–6 Process Flow for Desktop Identity Web Services

Considering Assumptions, Dependencies and Constraints

As you plan your deployment, consider the following assumptions, dependencies, and constraints to determine if your environment is appropriate for using the ID-WSF.

Assumptions and Dependencies

The fundamental difference between ID-WSF and generic web services is that the ID-WSF defines a security framework around user identity. The ID-WSF allows an end user to register his service offerings with their trusted trust authorities. Generic web services advertise their offerings through either a Web Service Description Language (WSDL) file or by the Enterprise Universal Description, Discovery, and Integration (UDDI) registry. These use cases are driven through Liberty ID-WSF and thus have a dependency on other Liberty protocols such as ID-FF and SAMLv2.

Constraints

The majority of identity web services are deployed in the Mobile Communications industry. For server-side web service providers OpenSSO Enterprise provides a comprehensive solution for ID-WSF. However, solutions for Liberty enabled clients do not provide the same degree of coverage. For example, the Client SDK is not J2ME-compatible and will not work with mobile devices that typically use Midlets to invoke Identity Web Services. Also, the OpenSSO Enterprise ID-WSF does not implement all profiles for the Liberty-enabled user agent or device (LUAD) clients.

Understanding Typical Business Use Cases

The adoption of identity web services is widespread especially in mobile communications-based businesses. Many of the telecommunication industries in Europe have invested heavily in ID-WSF based architectures, although the adoption in US is relatively smaller.

The following figure illustrates a simple E-commerce deployment using OpenSSO Enterprise.

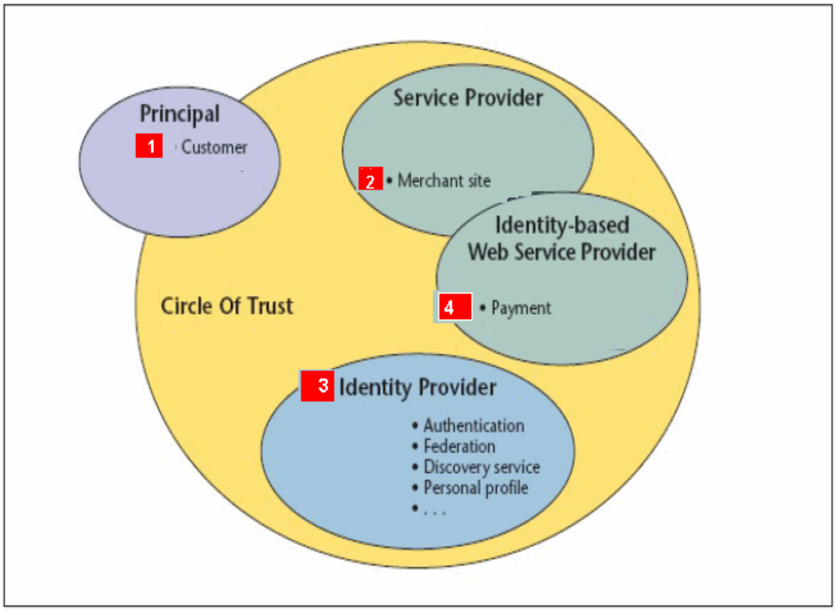


FIGURE 10–7 Identity Web Services Business Use Case

1. The customer is browsing the merchant site and initiates a purchase of some item.
2. The merchant who needs to authenticate the customer can request authentication through one of the trusted Identity Providers. The authentication happens here through ID-FF or SAMLv2, masking the real identity of the customer. This helps preserve the customer's privacy.
3. The merchant requests the payment service to guarantee the transaction. First the merchant site discovers the customer payment service through the Discovery Service. Then the merchant site requests the payment services on behalf of the principal. This step leverages the fact that the Liberty discovery mechanism is per principal-oriented, allowing merchants to request payments without having to know the real customer identity.
4. The payment service validates the transaction. Before charging the customer, the payment service may take a user consent. The payment service uses the Liberty Interaction Service for doing this. With successful purchase, the merchant returns the confirmation of purchase and delivers the service.

Setting Up and Configuring ID-WSF

For demonstration purposes, this section describes the high-level setup of a simple web-based service which was used as the basis for this chapter. OpenSSO Enterprise plays various roles in this environment depending:

- OpenSSO Enterprise is installed in the Web Services Client Domain.
- OpenSSO Enterprise is installed in the Web Services Provider Domain.
- OpenSSO Enterprise acts as a Trust Authority Service.

For browser-based deployment, the Liberty Personal Profile Web Service that is shipped with OpenSSO Enterprise is used. The user profile information is stored in the LDAP user data store.

For desktop based deployment, a simple weather service web service is developed to demonstrate the developer aspect.

Evaluating Benefits and Tradeoffs

The following lists are useful in helping you determine whether ID-WSF is suitable for your environment.

Benefits

- OpenSSO Enterprise provides an extensive, customizable framework for ID-WSF.
- ID-WSF consumers can leverage not only use Identity Web Services security, but can also leverage the OpenSSO Enterprise policy and access control features.
- Processing a custom web service payload can totally be independent of the OpenSSO Enterprise infrastructure.

Tradeoffs

- Developers must develop and deploy their web services along with OpenSSO Enterprise.
- If web services already exist in the environment, the developers must integrate them with OpenSSO Enterprise.

  OpenSSO Enterprise ID-WSF framework is tested in interoperable environments and certified by the Liberty Alliance. So the web services are almost certain to work in multi-vendor environments as long as the remote party is also a certified implementation.
- The lack of mobile client support with the OpenSSO Enterprise Client SDK may be a limitation. However, the primary use for OpenSSO Enterprise is in enterprise web service deployments.

Finding More Information

- Chapter 10, “Federation Management with OpenSSO Enterprise,” in *Sun OpenSSO Enterprise 8.0 Technical Overview*
- Liberty ID-WSF Specifications

  `http://www.projectliberty.org/resource_center/specifications/liberty-alliance_id_wsf_1_1_specifications`

CHAPTER 11

Securing Web Services Using Security Token Service (WS-* Specifications)

A web service is an application that exposes some type of business or infrastructure functionality though a callable interface that is both language-neutral and platform-independent. A company's web-based phonebook is an example of such an application. This document provides information about how you can use OpenSSO Enterprise to protect your web-base applications and services from unauthorized use or attack.

The following topics are included in this chapter:

About Web Services Security Models

A web service exposes its functionality using the Web Services Framework (WSF). The Web Services Framework defines its interface using Web Service Description Language (WSDL), and communicates using Simple Object Access Protocol (SOAP) and Extensible Markup Language (XML) messages. Although web services enable open, flexible, and adaptive interfaces, this openness create security risks. Without proper security measures in place, a web service can expose vulnerabilities that could allow unauthorized entities access to the enterprise. You can ensure the integrity, confidentiality and security of web services by using a comprehensive security model. In a good security model, web services are secured either point-to-point as provided by SSL/TLS, or end-to-end as specified by the Web Services Security (WS-Security) Framework.

The WS-Security Framework was developed by the OASIS Security committee along with other WS-* specifications such as WS-Trust and WSPolicy. Transport-layer or point-to-point transport mechanisms transmit information over the wire between clients and providers.

Transport-layer security relies on secure HTTP transport (HTTPS) using Secure Sockets Layer (SSL). Transport security can be used for authentication, message integrity, and confidentiality. When running over an SSL-protected session, the server and client can authenticate one another and negotiate an encryption algorithm and cryptographic keys before the application protocol transmits or receives its first byte of data. Security is enabled from the time data leaves the consumer until the data arrives at the provider, or from the time the data leaves the provider until the data arrives at the consumer. Sometimes security data transfer can transpire even across intermediaries.

The following figure illustrates a security model that uses point-to-point security.

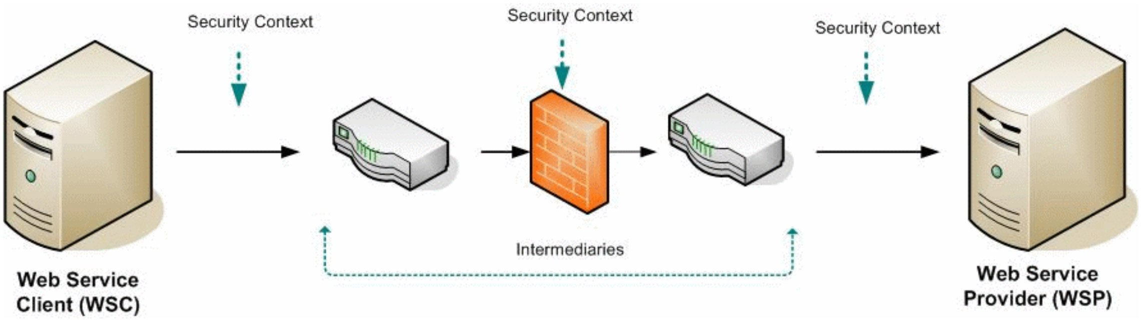


FIGURE 11–1 Secure Communication Channel Providing Point-to-Point Security

A drawback to using point-to-point security is that the message is not protected once it gets to its destination. One solution is to encrypt the message before sending using application security.

Using application-layer or end-to-end security, the message is secure even when the message is not in transit. Additionally, in application-layer security, the security information is contained within the SOAP message and the message attachment. This allows security information to travel along with the message or attachment. For example, a portion of the message may be signed by a sender and encrypted for a particular receiver. When the message is sent from the initial sender, it may pass through intermediate nodes before reaching its intended receiver. When this happens, the encrypted portions continue to be opaque to any intermediate nodes, and can only be decrypted by the intended receiver. Message security can be used to decouple message protection from message transport so that the message remains protected after transmission. For this reason, application-layer security is also sometimes referred to as end-to-end security .

The following figure illustrates a security model that uses end-to-end security.

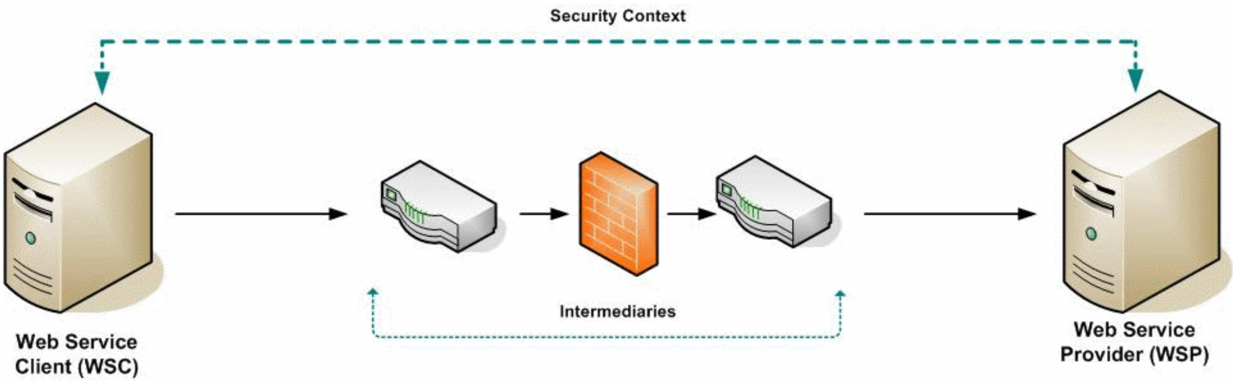


FIGURE 11–2 Secure Communication Channel Providing End-to-End Security

Application-layer security provides the following: Confidentiality, by encrypting message parts, integrity, by digital signatures , and authentication, by requiring username or X.509 tokens.

About OpenSSO Enterprise Web Services Security

You can configure OpenSSO Enterprise to act as a security token service, or as a web service security provider. When you use OpenSSO Enterprise to act as a web service security provider, you must configure both the web service client and at the web service provider.

Security Token Service

When configured as a security token service, OpenSSO Enterprise acts as a generic web service that does the following:

- Issues, renews, cancels, and validates security tokens.
- Enables customers to write custom security token providers by extending the framework.
- Provides standards-based APIs for clients and applications to access the security token service.
- Provides mechanisms to support additional security token types such as Kerberos and others.

Web Services Security Provider

OpenSSO Enterprise 8.0 provides web services security support for client applications which are based on Java API for XML Web Services (JAX-WS) or SOAP with Attachments API for Java (SAAJ). For JAX-WS based clients, web services security can be enforced at either the web or JavaEE container level using container-provided security authentication and authorization plug-ins, or using JAX-WS Handlers. The JSR 196 specification is one of the well known authentication and authorization security SPIs, currently supported by the Sun Application Server. Handlers are interceptors that can be easily plugged into the Java API for XML-Based Web Services (JAX-WS) 2.0 runtime environment to do additional processing of inbound and outbound messages.

For non-JAX-WS based client applications such as SAAJ-based, you can use the OpenSSO Enterprise client SDK can to programmatically, explicitly secure and validate both outbound and inbound messages between the web service client and web service provider.

Analyzing the Deployment Architecture

In this deployment example, messages are exchanged using the SOAP protocol to transfer security tokens between the communicating web service client and web service provider entities. The web service security providers can work independently of the OpenSSO Enterprise instance which is deployed as security token service. Web service security providers can secure the SOAP message by obtaining the security tokens from a vendor-neutral security token service.

The following are the major components in this deployment example:

- OpenSSO Enterprise configured as a security token service
- OpenSSO Enterprise configured as a web service security provider on a web service client
- OpenSSO Enterprise configured as a web service security provider on a web service provider
- Browser

The following figure illustrates the deployment architecture for using OpenSSO Enterprise to secure a web-based calendar service.

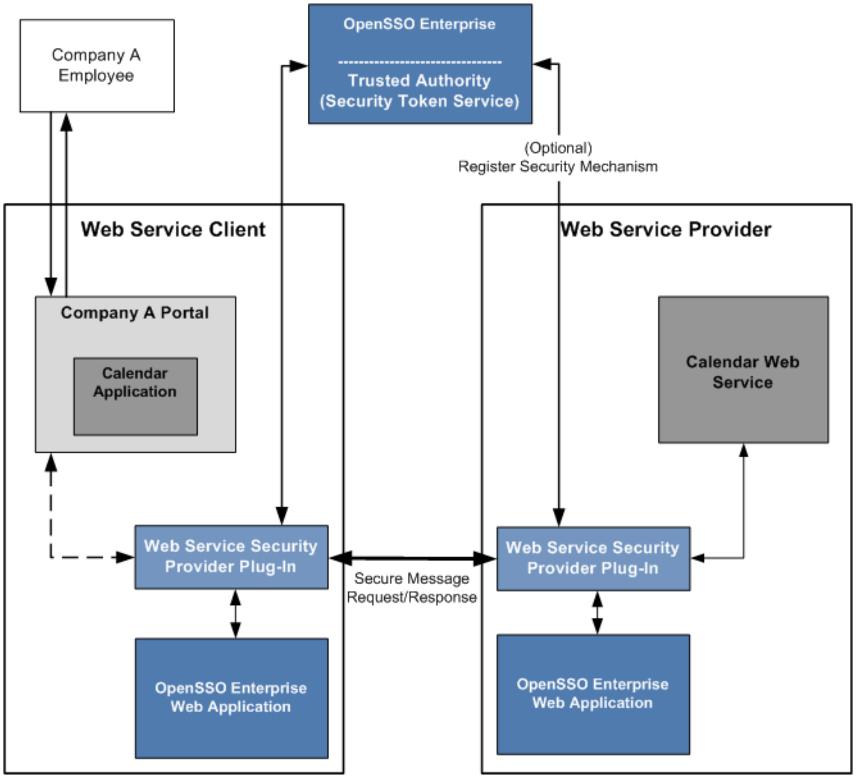


FIGURE 11–3 Deployment Architecture for Web Security Service Using Secure Token Service

In this deployment example, a company employee has a user account in the Example Company identity system. The employee wants to access an internal calendar application to view a calendar service. The calendar application is part of the Example Company portal. All Example Company employees are required to authenticate themselves before accessing this internal portal. Additionally, the individual employee's credentials, such as role or group membership, must be validated before the employee can access the calendar application service.

The calendar application, on the employee's behalf, securely supplies the employee's credentials to the remote calendar web service.

The following two figures illustrate the process flow for this business use case.

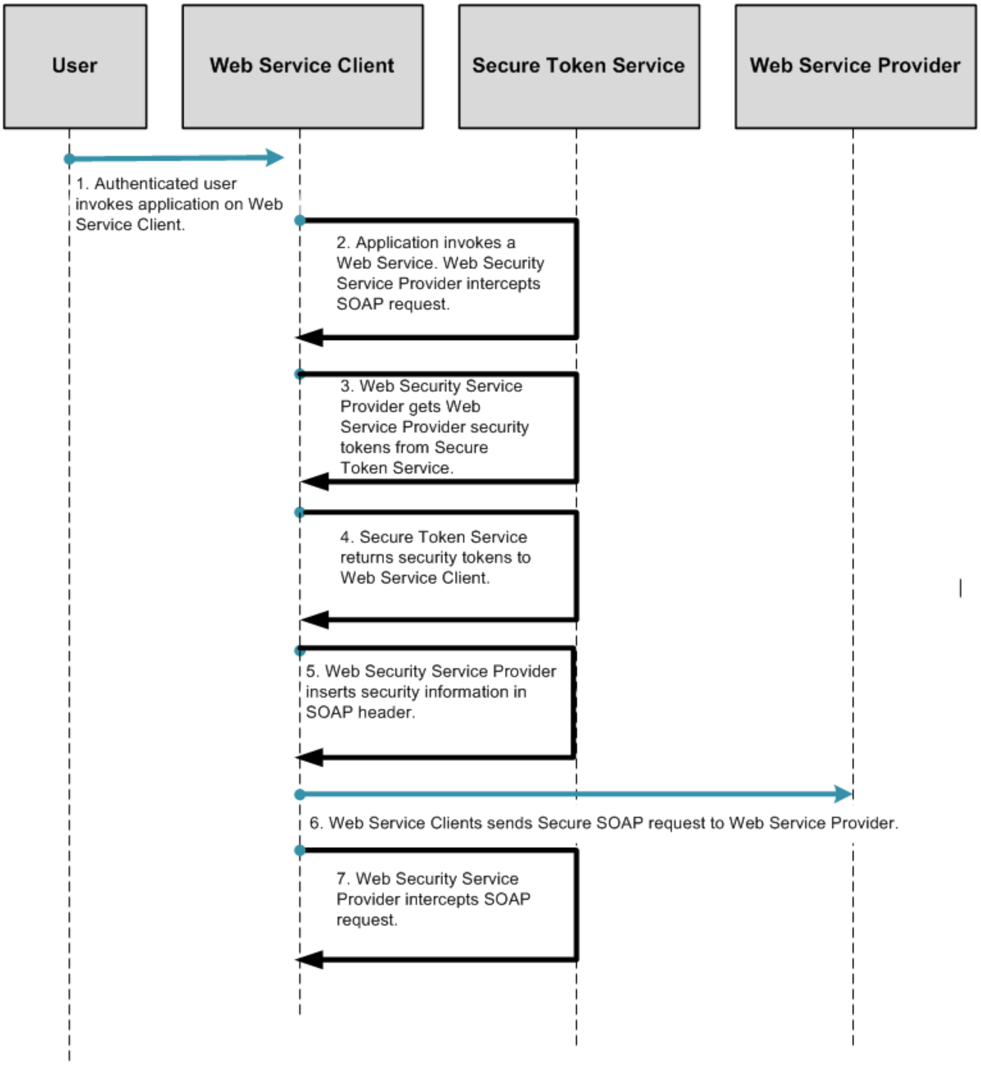
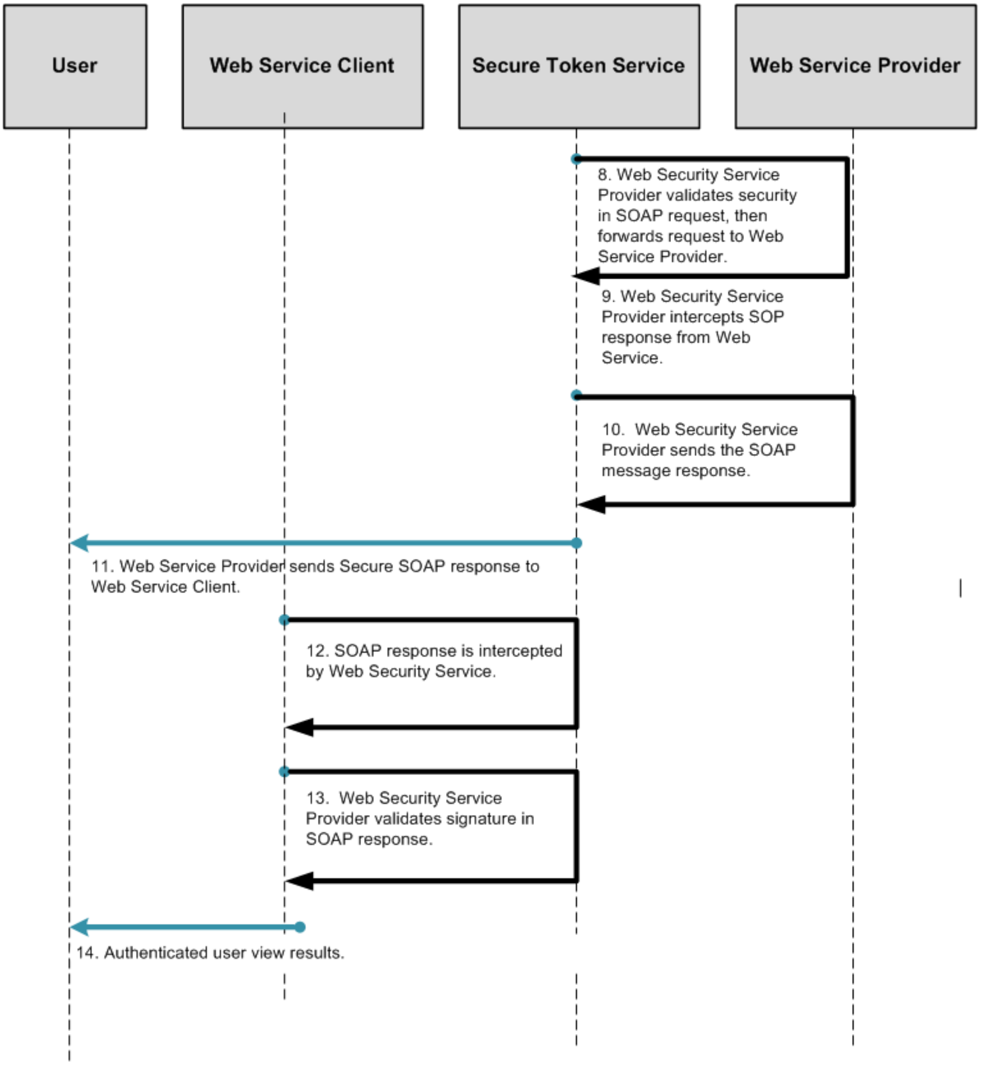

User
Web Service Client
Secure Token Service
Web Service Provider
1. Authenticated user invokes application on Web Service Client.
2. Application invokes a Web Service. Web Security Service Provider intercepts SOAP request.
3. Web Security Service Provider gets Web Service Provider security tokens from Secure Token Service.
4. Secure Token Service returns security tokens to Web Service Client.
5. Web Security Service Provider inserts security information in SOAP header.
6. Web Service Clients sends Secure SOAP request to Web Service Provider.
7. Web Security Service Provider intercepts SOAP request.

FIGURE 11–4 Process Flow for Web Security Service Using Secure Token Service (Continued on next page)



User
Web Service Client
Secure Token Service
Web Service Provider
8. Web Security Service Provider validates security in SOAP request, then forwards request to Web Service Provider.
9. Web Security Service Provider intercepts SOP response from Web Service.
10. Web Security Service Provider sends the SOAP message response.
11. Web Service Provider sends Secure SOAP response to Web Service Client.
12. SOAP response is intercepted by Web Security Service.
13. Web Security Service Provider validates signature in SOAP response.
14. Authenticated user view results.

FIGURE 11–5 Process Flow for Web Security Service Using Secure Token Service (Continued)

Understanding Typical Business Use Cases

The following are the types of users involved in transactions using Web Services Security and Secure Token Service:

- Developers

Developers are typically application owners who use the OpenSSO Enterprise Client SDK APIs to communicate with a local security token service instance, or with web service security providers, to secure the applications making web services calls.

- System administrators

  Administrators are responsible for the configuring the OpenSSO Enterprise secure token service.

- End users

  End users such as company employees are exposed to OpenSSO Enterprise when they access the published web services.

Use Case 1: Kerberos Security Token

The following figure illustrates the process flow for a secured stock quotes web service using a Kerberos security token.

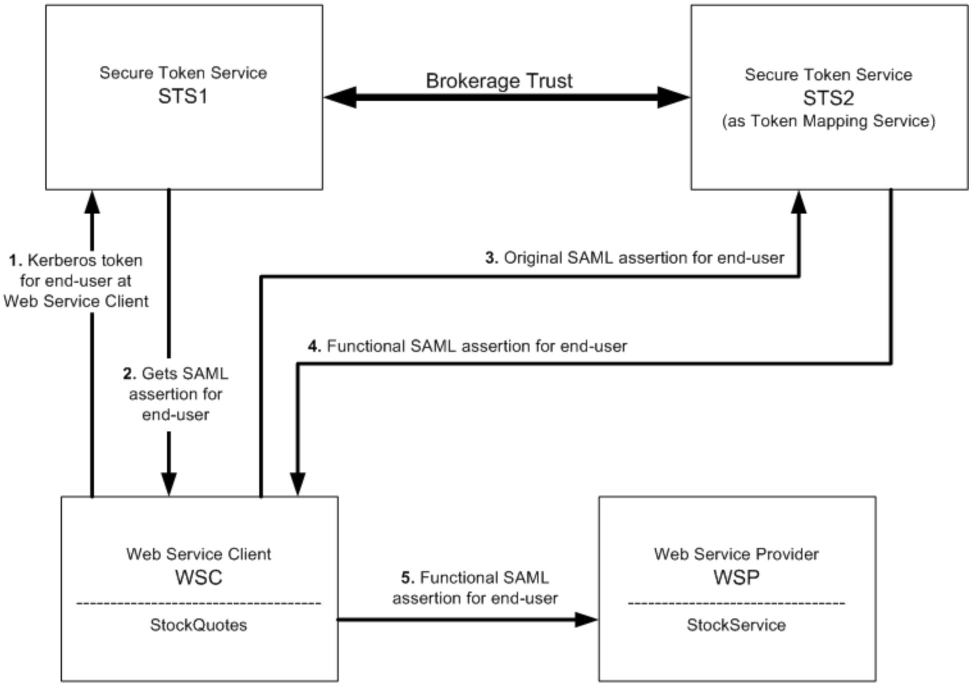

Secure Token Service
STS1
Brokerage Trust
Secure Token Service
STS2
(as Token Mapping Service)
1. Kerberos token for end-user at Web Service Client
2. Gets SAML assertion for end-user
3. Original SAML assertion for end-user
4. Functional SAML assertion for end-user
Web Service Client
WSC
StockQuotes
5. Functional SAML assertion for end-user
Web Service Provider
WSP
StockService

FIGURE 11–6 Process Flow for a Stock Quote Web Service Using Kerberos Security Token

1. The Web Service Client authenticates to STS-1 instance with the end user's Kerberos token .

The end user logs in to the Desktop at the Web Service Client. This can be viewed as a Kerberos token for the Web Service Client, too.

2. The Web Service Client gets the SAML token for the end user (Web Service Client).
3. The Web Service Client then talks to the STS2 (Token Mapping Service) .

   The Web Service Client converts the end user's (Web Service Client) SAML token to a functional SAML token.

   This is called an organizational SAML token, and used as an authentication token of the Web Service Client to STS2. Here the functional SAML token has the same identity or owner as the original SAML token, but with more attributes and privileges.
4. The Web Service Client then secures the web services request to the Web Service Provider with the functional SAML token.

The following are configuration suggestions for this use case:

1. STS client agent - profile name is STS1

Security Mechanism	Kerberos
STS End Point	of STS1 service
STS Mex Ed Point	of STS1 service

2. STS client agent - profile name is STS2

Security Mechanism	STSSecurity
STS config	STS1
STS End Point	of STS2 service
STS Mex Ed Point	of STS2 service

3. WSC agent - profile name is StockService or wsc

Security Mechanism	STSSecurity
STS config	STS2
WSP End Point	default

Use Case 2: SAML1 Security Token

The following figure illustrates the process flow for a bank loan web service using a SAML 1 security token.

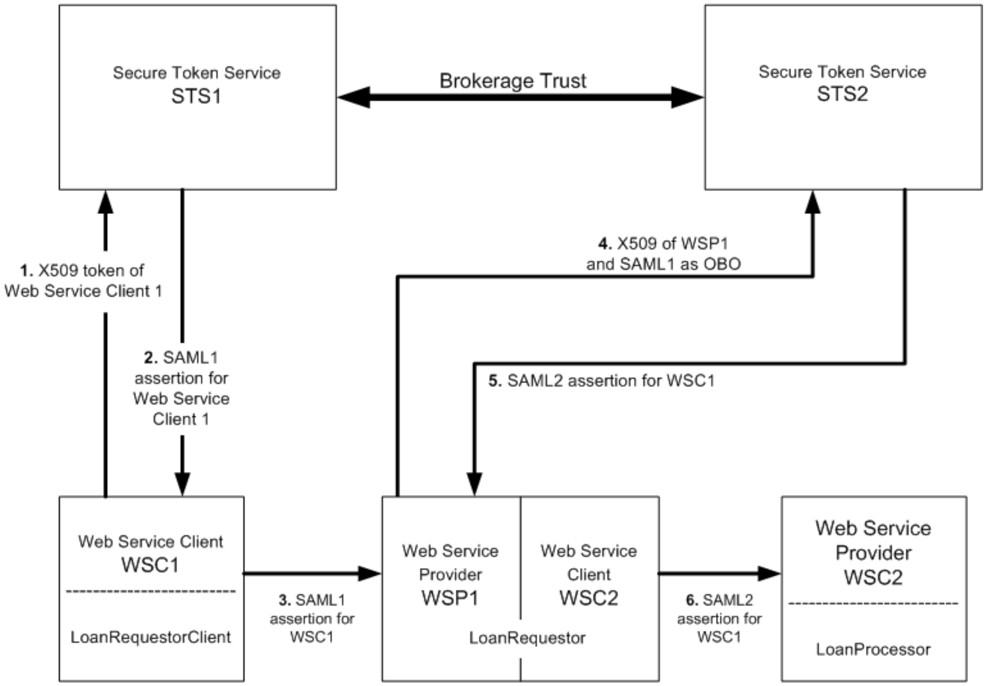


FIGURE 11–7 Process Flow for a Bank Loan Web Service Using SAML1 Security Token

1. WSC1 authenticates to STS1 with its X509 token and gets to SAML1 token (owner is WSC1).
2. WSC1 secures web service to WSP1 with its SAML1 token.
3. WSP1 then authenticates to STS2 with its X509 token, and sends the SAML1 token of WSC1.

   The SAML1 token is sent on behalf of the X509 token in order to convert it to SAML2 token for WSC1.
4. WSC2 just passes through this SAML2 token of WSC1 to WSP2.

   WSC2 secures the web service to WSP2 with the SAML2 token of WSC1.

The following are configuration suggestions for the Bank Loan use case:

1. WSC agent - profile name is LoanRequestorService for WSC1

Security Mechanism	STSSecurity
STS config	SecurityTokenService

2. WSP agent - profile name is wsp for WSP1

WSP End Point	default

Authentication Chain	ldapService
Token Conversion Type	SAML2 token

3. WSC agent - profile name is LoanProcessorService for WSC2

Use Pass Through Security Token	checked

Use Case 3: X509 Security Token

The following figure illustrates the process flow for a bank loan web service using a X509 security token.

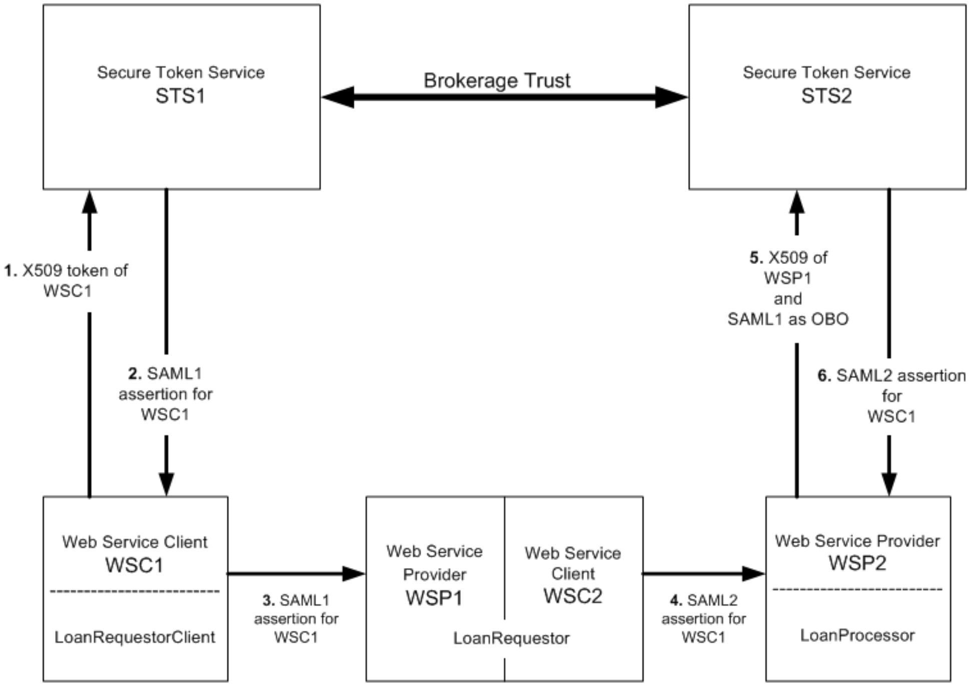


FIGURE 11–8 Process Flow for a Bank Loan Web Service Using an X509 Security Token

1. WSC1 authenticates to STS1 with its X509 token and gets the SAML1 token (owner is WSC1).
2. WSC1 secures web service to WSP1 with its SAML1 token.
3. WSP1/WSC2 just passes through this SAML1 token of WSC1 to WSP2

   Secures web service to WSP2 with SAML1 token of WSC1.

4. WSP2 then authenticates to STS2 with its X509 token.

   Sends SAML1 token of WSC1 as On Behalf Of token in order to convert it to SAML2 token for WSC1.

The following are suggested configurations:

1. Web Service Client agent - profile name is LoanRequestorService for WSC1

Security Mechanism	STSSecurity
STS Configuration	SecurityTokenService

2. Web Service Provider agent - profile name is wsp for WSP2

Web Service Provider End Point	default
Authentication Chain	ldapService
Token Conversion Type	SAML2 token

3. WSC agent - profile name is LoanProcessorService for WSC2

Use Pass Through Security Token	checked

Considering Assumptions, Dependencies, and Constraints

Before using OpenSSO Enterprise to secure web services, you must resolve the following issues:

Assumptions and Dependencies

- Metadata exchange (MEX) between individual components has already been completed.
- Users are authenticated to an Identity Provider.

Constraints

- The scope of the current Web Service Security provider plug-ins is limited to JSR 196 SPI implementation, and is only supported on Sun Application Server version 9.0 and above.
- Clients using JAX-WS based applications on web or J2EE containers that do not support JSR 196 specifications must use handlers.
- Clients using SAAJ based applications need to secure the messages programmatically using the OpenSSO Enterprise Client SDK.

Setting Up and Configuring Web Services Security Using Security Token Service

OpenSSO Enterprise ships with the `StockQuoteClient` and `StockService` sample applications. These sample applications show you how the Web Service Client, Web Service Provider, and Secure Token Service interact together in a demonstration environment. The sample applications are available in the `wssagents/openssowssproviders.zip` on the OpenSSO Enterprise download site.

To configure and deploy the sample applications, see the README files in the zipped archive. The following steps describe the high-level tasks for setting up the deployment illustrated in section “Use Case 1: Kerberos Security Token” on page 146. This deployment uses the `StockQuoteClient` (Web Service Client) and `StockService` (Web Service Provider) applications, from the OpenSSO Enterprise samples.

1. Create and configure a Secure Token Service instance, STS-1.
    a. Install the STS-1 instance.
    b. Configure a policy agent profile for the Web Service Provider.
    c. Select security mechanisms.
2. Create and configure a second Secure Token Service instance, STS-2 instance.
    a. Install the STS-2 instance.
    b. Configure an policy agent profile for the STS-1 instance.
3. Create and configure the Configuration Instance for the Web Service Client and Web Service Provider.
    a. Install the WSC-WSP Configuration Instance.
    b. Create And Configure a policy agent profile for the STS-2 instance.
    c. Configure a policy agent profile for the STS-1 instance.
    d. Configure a policy agent profile for the Web Service Client.
    e. Configure a policy agent profile for the Web Service Provider.
4. Create and configure the Web Service Client instance.
    a. Install the Web Service Client Instance.
    b. Configure the Web Service Client as an OpenSSO Enterprise client.
    c. Configure the Web Service Client Glassfish instance.
        i. Update the Glassfish classpath.
        ii. Configure for end-user authentication.
5. Create and configure the Web Service Provider instance.
    a. Install the Web Service Provider instance.
    b. The Web Service Provider as an OpenSSO Enterprise client.
    c. Configure the Web Service Provider Glassfish instance.

6. Build and deploy the Web Service Client application.
7. Build and deploy the Web Service Provider application.
8. Test to verify that the Web Service Security works as designed.

Evaluating Benefits and Tradeoffs

The following lists are designed to help you determine whether using OpenSSO Enterprise to secure web services is suitable in your environment.

Benefits

- Based on standards specification as developed by OASIS, Liberty Alliance Project, Web Services Interoperability Organization, World Wide Web Consortium.
- Secures the message over all hops and after the message arrives at its destination.
- Security is fine-grained and can selectively be applied to different portions of a message (and to attachments if using XWSS).
- Can be used in conjunction with intermediaries over multiple hops.
- Is independent of the application environment or transport protocol.
- Securing web services interactions is transparent to the client applications when web service security providers are configured in web or J2EE containers.

Tradeoff

The drawback to using message-level security is that it is relatively complex and adds some overhead to processing.

CHAPTER 12

Enabling Single Sign-On Between Sun Identity Manager and OpenSSO Enterprise

This chapter provides information about integrating Sun OpenSSO Enterprise 8.0 with Sun Identity Manager 8.0. This information is useful when you want to enable single sign-on between the two products, or when you want to use Identity Manager to provision users to OpenSSO.

The following topics are contained in this chapter:

About Sun Identity Manager

Sun Identity Manager enables you to securely and efficiently manage and audit access to accounts and resources, and to distribute access management overhead. By mapping Identity Manager objects to the entities you manage such as users and resources, you significantly increase the efficiency of your operations. The Identity Manager solution enables you to:

- Manage account access to a large variety of systems and resources.
- Securely manage dynamic account information for each user's array of accounts.
- Set up delegated rights to create and manage user account data.
- Handle large numbers of enterprise resources, as well as an increasingly large number of extranet customers and partners.
- Securely authorize user access to enterprise information systems.

  Grant, manage, and revoke access privileges across internal and external organizations.

- Keep data in sync by not keeping data.

Analyzing the Deployment Architecture

This deployment requires an OpenSSO Enterprise server, an Identity Manager server, and a Sun Policy Agent installed on the Identity Manager web container. The OpenSSO Enterprise server is configured with two data stores: the OpenSSO configuration data store, and the Sun Directory Server user data store. The user data store is configured in the OpenSSO Enterprise subrealm. The Identity Manager server is configured to use a MySQL server for both Identity Manager configuration and Identity Manager user data.

The following figure illustrates the main components of the deployment.

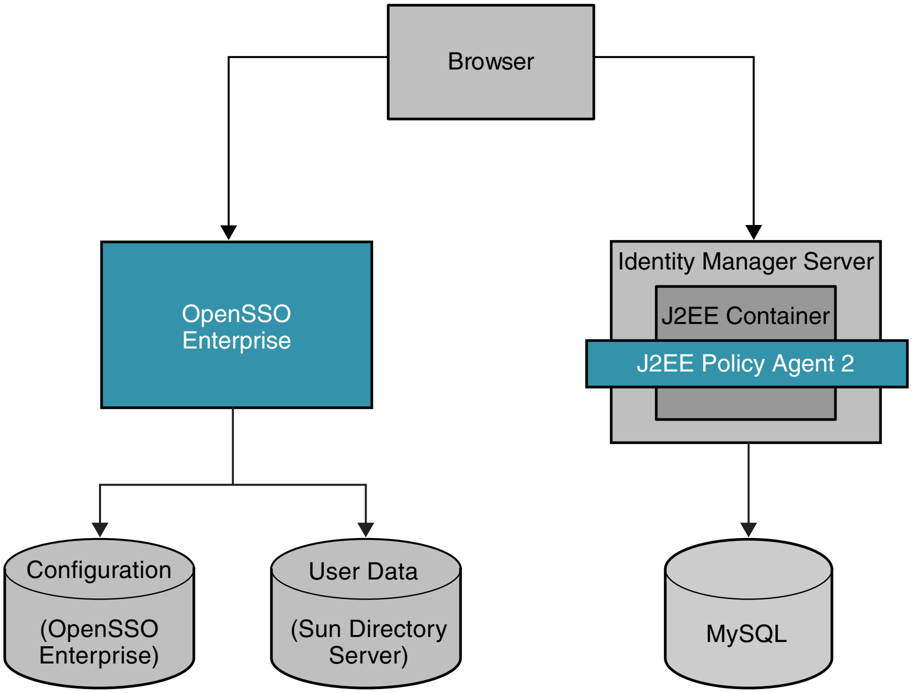


FIGURE 12–1 Deployment Architecture for Enabling Single Sign-On Between OpenSSO Enterprise and Identity Manager

The Sun Policy Agent plays an important role in the single sign-on between OpenSSO Enterprise and Identity Manager. In addition to protecting the Identity Manager content pages, it helps map the OpenSSO Enterprise user ID to the Identity Manager user ID.

The following two figures illustrate a typical process flow.

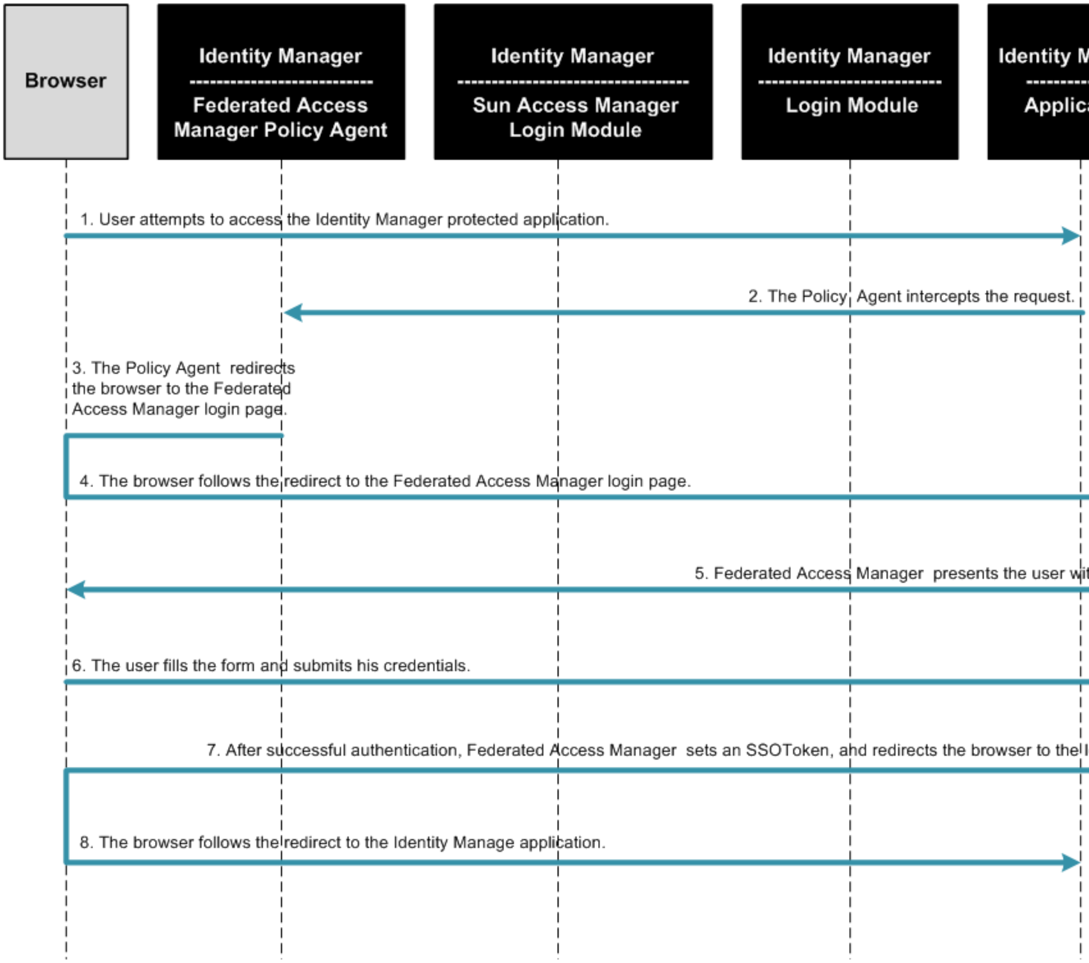

Browser
Identity Manager
Federated Access Manager Policy Agent
Identity Manager
Sun Access Manager Login Module
Identity Manager
Login Module
Identity M
Applic
1. User attempts to access the Identity Manager protected application.
2. The Policy Agent intercepts the request.
3. The Policy Agent redirects the browser to the Federated Access Manager login page.
4. The browser follows the redirect to the Federated Access Manager login page.
5. Federated Access Manager presents the user wi
6. The user fills the form and submits his credentials.
7. After successful authentication, Federated Access Manager sets an SSOToken, and redirects the browser to the I
8. The browser follows the redirect to the Identity Manage application.

FIGURE 12–2 Process Flow for Single Sign-On Between OpenSSO Enterprise and Identity Manager (Continued on next page)

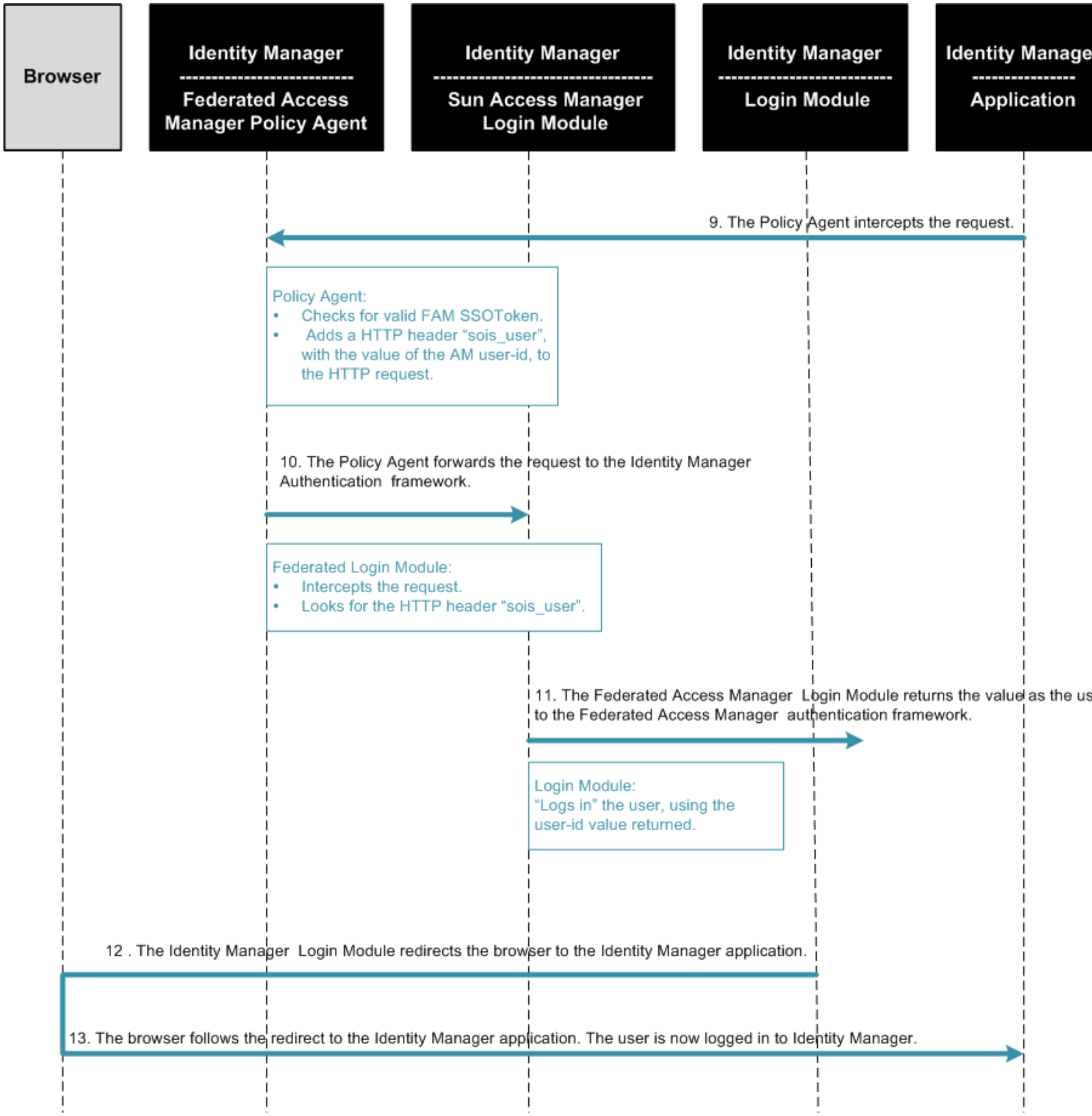

Browser
Identity Manager
Federated Access Manager Policy Agent
Identity Manager
Sun Access Manager Login Module
Identity Manager
Login Module
Identity Manage
Application
9. The Policy Agent intercepts the request.
Policy Agent:
• Checks for valid FAM SSOToken.
• Adds a HTTP header "sois_user", with the value of the AM user-id, to the HTTP request.
10. The Policy Agent forwards the request to the Identity Manager Authentication framework.
Federated Login Module:
• Intercepts the request.
• Looks for the HTTP header "sois_user".
11. The Federated Access Manager Login Module returns the value as the us
to the Federated Access Manager authentication framework.
Login Module:
"Logs in" the user, using the user-id value returned.
12 . The Identity Manager Login Module redirects the browser to the Identity Manager application.
13. The browser follows the redirect to the Identity Manager application. The user is now logged in to Identity Manager.

FIGURE 12–3 Process Flow for Single Sign-On Between OpenSSO Enterprise and Identity Manager (Continued)

The following UML use case diagram illustrates the provisioning and retrieval of objects in Identity Manager.

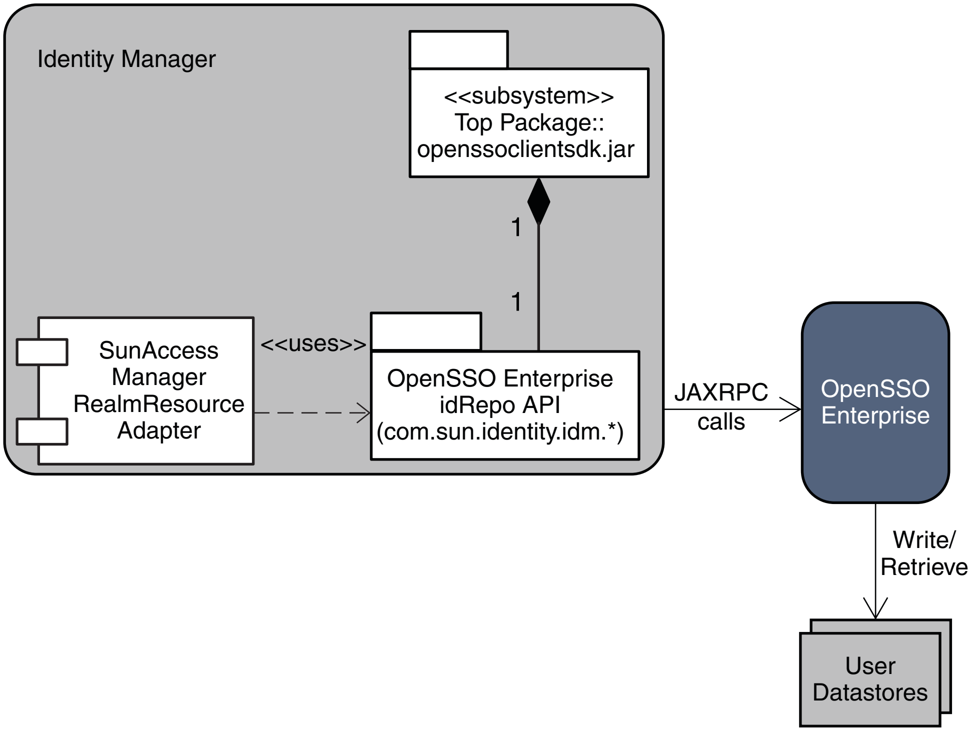


FIGURE 12–4 Provisioning and Retrieving Objects in Identity Manager

Considering the Deployment Assumptions, Dependencies, and Constraints

Before you can enable single sign-on between OpenSSO Enterprise and Identity Manager, you must resolve the following issues.

Assumptions

- OpenSSO Enterprise must already be installed.
- OpenSSO Enterprise must be installed in the Realm mode of operation.
- OpenSSO Enterprise must be configured with Sun OpenDS as the embedded configuration data store.
- OpenSSO Enterprise must contain a sub-realm named `idm` configured with Sun Directory Server user data store.

- The Directory Server user data store must also have the OpenSSO Enterprise schema loaded in it.

Dependencies

- If you require roles to be provisioned on Identity Manager to OpenSSO Enterprise, and you are using the Sun Access Manager Resource Adapter, then the OpenSSO Enterprise user data store must have the OpenSSO Enterprise schema loaded in it.

  If the OpenSSO Enterprise data store plug-in for a generic LDAPv3 data store has no OpenSSO Enterprise schema in it, the data store plug-in does not support the management of either managed roles or filtered roles through it. The OpenSSO Enterprise data store plug-in is designed to work this way. It is important to note here that you do not have to provision roles in order to achieve single sign-on.
- Because OpenSSO Enterprise is installed in the Realm mode of operation, the Identity Manager resource adapter for Realm mode, `SunAccessManagerRealmResourceAdapter`, must be configured on Identity Manager.

  In earlier versions of OpenSSO Enterprise, previously known as Access Manager, the product was installed in the Legacy mode of operation. In Legacy mode, a different Identity Manager resource adapter, `SunAccessManagerResourceAdapter`, must be configured on Identity Manager. Both types of adapters have the same functionality with one difference. The `SunAccessManagerResourceAdapter` uses the legacy Access Manager `AMSDK` API, while the `SunAccessManagerRealmResourceAdapter` uses the OpenSSO Enterprise `idRepo` API. The `idrepo` APIs are the next-generation OpenSSO client APIs, and will eventually replace the legacy `AMSDK`API.

Constraints

When testing the Sun Access Manager Resource Adapter, before the Policy Agent has been installed, the client-side `AMConfig.properties` file must be configured with `amadmin` or a user that has privileges to read the OpenSSO Enterprise configuration data, for the property `com.sun.identity.agents.app.username`. If a different type of user is used in this configuration, the configuration of the Sun Access Manager Resource Adapter fails. This change is required only until the Policy Agent is installed. After the Policy Agent has been installed, the `AMConfig.properties` file is not required and can be deleted from the filesystem.

Although this document describes the use case where Identity Manager and OpenSSO Enterprise are configured for both single sign-on and provisioning, it is possible to configure the deployment for either single sign-on or provisioning only. If single sign-on between OpenSSO Enterprise and Identity Manager is not required, then the OpenSSO Enterprise Policy Agent does not need to be installed and configured. You can ignore the steps that involve configuring the OpenSSO Enterprise Policy Agent.

Understanding Typical Business Use Cases

The most common use case for this deployment is when a company uses OpenSSO Enterprise with OpenSSO Enterprise Policy Agents to protect business applications, including Identity Manager applications. The main objective is to streamline the login process for end-users who are already logged in to OpenSSO Enterprise. For example, if a user is already logged in to OpenSSO Enterprise, the user should experience a seamless transition into any Identity Manager application without having to log in to Identity Manager. A secondary objective is to provide a measure of controlled access to all Identity Manager applications.

Another typical use case for this deployment is provisioning. The company uses Identity Manager to provision users into various business systems such as human resources or accounting systems. This can also include provisioning users into the company's business systems that are protected by OpenSSO Enterprise.

Setting Up and Configuring Single Sign-On Between Identity Manager and OpenSSO Enterprise

The following components are used in this deployment:

- Sun Solaris Operating System 10
- Sun Federated Access Manager 8.0 / OpenSSO 8.0
- Sun Java Identity Manager 8.0
- Sun Java Web Server 7.0
- Sun Java Application Server 9.1
- Sun Java Directory Server 6.1
- MySQL 5.1
- MySQL Connector 5.0
- Sun Java AM Policy Agent 3.0 (for Sun Java Application Server 9.1) (Optional)
- Netbeans IDE 6.0.1

See the Chapter 1, "Integrating Sun Identity Manager ," in *Sun OpenSSO Enterprise 8.0 Integration Guide* for detailed installation steps. The following is a summary of high-level procedures you must complete to enable single sign-on between OpenSSO Enterprise and Identity Manager:

1. Installing And Configuring MYSQL
   - Install MySQL.
   - Complete post-installation tasks.
2. Installing And Configuring Identity Manager Application Server
   - Install Application Server.
   - Install Identity Manager on Application Server.
   - Complete post-installation configuration on Application Server.

3. Create an OpenSSO Enterprise Realm Administrator in OpenSSO Enterprise.
4. Create an OpenSSO Enterprise Realm Resource Object in Identity Manager.
5. Provision identities from Identity Manager to OpenSSO Enterprise.
    - Provision a test user from Identity Manager into OpenSSO Enterprise.
    - Provision a test role from Identity Manager into OpenSSO Enterprise.
    - Provision an Admin-User from Identity Manager into OpenSSO Enterprise
    - Provision an Admin-Role from Identity Manager into OpenSSO Enterprise.
6. Install and Configure the OpenSSO Enterprise Policy Agent on Identity Manager.
    - Complete pre-installation tasks for the OpenSSO Enterprise Policy Agent.
    - Install the OpenSSO Enterprise Policy Agent on the Identity Manager server.
    - Configure the OpenSSO Enterprise Policy Agent on the OpenSSO Enterprise Server.
    - Disable OpenSSO Enterprise Policy Agent protection of the Identity Manager server.
    - Configure the OpenSSO Enterprise Policy Agent on the Identity Manager server.
7. Configure Identity Manager for single sign-on.
    - Configure Identity Manager Login Module Groups.
    - Configure the Identity Manager User Login Interface.
    - Configure the Identity Manager Admin Login Interface.
8. Test single sign-on from OpenSSO Enterprise to Identity Manager.
    - Re-Enable OpenSSO Enterprise Policy Agent protection of the Identity Manager server.
    - Test end-user single sign-on between OpenSSO Enterprise and Identity Manager.
    - Test Admin-User single sign between OpenSSO Enterprise and Identity Manager.

Evaluating Benefits and Tradeoffs

As you design your deployment architecture, be sure to consider the benefits, tradeoffs. The following lists may help you determine if enabling single sign-on between Identity Manager and OpenSSO Enterprise is appropriate to meet your business needs.

Benefits

- If you use an OpenSSO Enterprise user store that has the OpenSSO Enterprise schema loaded into it, you can provision managed-roles or filtered-roles into OpenSSO Enterprise.
- If you configure OpenSSO Enterprise with a generic LDAP user datastore that does not have the OpenSSO Enterprise schema loaded into it, then you could configure an LDAP Resource Adapter instance instead of the Sun Access Manager Resource Adapter on Identity Manager. The LDAP Resource Adapter on Identity Manager is a generic adapter that can be

used to provision or manage objects in any LDAP resource. This would potentially reduce the number of different types of Resource Adapters, that an Identity Manager administrator would have to maintain.

Tradeoffs

If the OpenSSO Enterprise user store does not have the OpenSSO Enterprise schema loaded into it, OpenSSO Enterprise would use the generic LDAPv3 plug-in for this datastore. The creation of managed-roles and filtered-roles is not supported on such a datastore.

Finding More Information

- Sun Identity Manager 8.0 product documentation

  `http://docs.sun.com/app/docs/coll/1514.5`

CHAPTER 13

Enabling Single Sign-On Using CA SiteMinder and OpenSSO Enterprise

This chapter describes options for co-locating CA SiteMinder with Sun OpenSSO Enterprise in the same environment. For more detailed information about configuring end-to-end SiteMinder single sign-on using OpenSSO, see the *Sun OpenSSO Enterprise 8.0 Integration Guide*.

The following topics are contained in this chapter:

About CA SiteMinder

Computer Associates (CA) SiteMinder, formerly Netegrity SiteMinder, is an enterprise infrastructure product that enables centralized, secure Web access management. Its features include user authentication and single sign-on, policy-based authorization, and identity federation. One of the first single sign-on products to arrive on the market, legacy SiteMinder installations still exist to protect enterprise applications in many company networks.

Analyzing the Deployment Architecture Options

This chapter describes single sign-On between Federation Access Manager and SiteMinder in both intranet and federated extranet environments. The examples in this chapter describe single sign-on, but do not include authorization.

SiteMinder and Federated Access Manager typically co-exist in the following use cases:

- Simple Single Sign-On

  Major components are OpenSSO Enterprise, an OpenSSO Enterprise Policy Agent, a custom OpenSSO Enterprise authentication module, SiteMinder, and a SiteMinder Policy Agent.
- Federated Single Sign-On in an Identity Provider Environment

  Major components are OpenSSO Enterprise, an OpenSSO Enterprise Policy Agent, a custom OpenSSO Enterprise authentication module, SiteMinder, and a SiteMinder Policy Agent.
- Federated Single Sign-On in a Service Provider Environment

  Major components are OpenSSO Enterprise, a custom OpenSSO Enterprise authentication module, SiteMinder, a custom SiteMinder plug-in, and SiteMinder Policy Agent.

Single logout for any these of these use cases can be implemented in many ways.

Logical architecture diagrams and process flow diagrams for these deployment options are described in the following section "Understanding the Business Use Cases."

Considering Assumptions, Dependencies, and Constraints

This chapter describes the conceptual integration between the two access management products. However, in real deployments the use cases will vary. In all the deployment architecture examples, the common data store is shared between two products when they are co-located. This document focuses on mutual validation of user sessions. However, mutual validation can be extended to attributes and other state information. The sessions are managed independently, and managing session timeouts are outside the scope of this document. Also, this document assumes the logout is relatively simple and involves invalidating both sessions as POST Logout process. For federation single sign-on, this document assumes SAMLv2 protocols. However, similar functionality can be achieved using other federation protocols such as ID-FF, WS-Federation, and SAML1.

Understanding Typical Business Use Cases

The following use cases focus on single sign-on enablement and do not describe authorization options:

- "Simple Single Sign-On " on page 165
- "Federated Single Sign-On" on page 167

Simple Single Sign-On

In a simple single sign-on example, the SiteMinder instance is already deployed and configured to protect some of the enterprise applications in a company intranet. In the architecture figure below, the legacy application is contained in the Protected Resource . The company wants to continue leveraging the legacy SiteMinder deployment as the authentication authority. The company also wants to add OpenSSO Enterprise to the environment to leverage its advanced features such as identity federation, XACML policies, web services, and so on. A Federated Access Manager policy agent protects the Protected Resource, while Federated Access Manager itself is protected by a SiteMinder policy agent. The following figure illustrates the deployment architecture for single sign-on using both SiteMinder and Federated Access Manager.

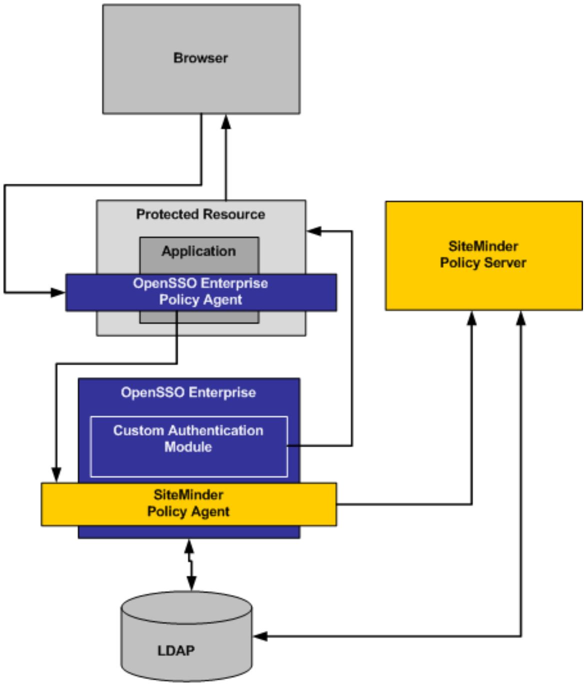


FIGURE 13–1 Deployment Architecture for Simple Single Sign-On with SiteMinder

The following figure illustrates the process flow in this deployment.

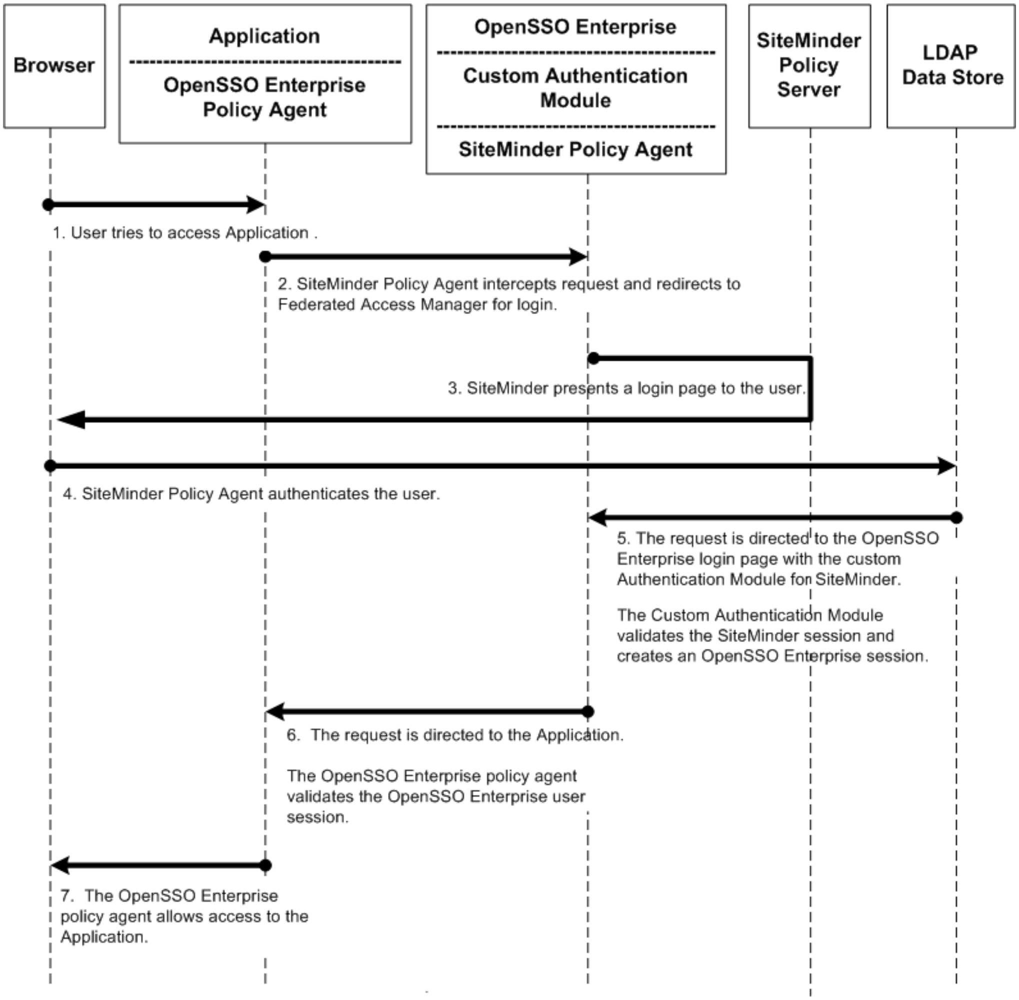

Browser
Application
OpenSSO Enterprise Policy Agent
OpenSSO Enterprise
Custom Authentication Module
SiteMinder Policy Agent
SiteMinder Policy Server
LDAP Data Store
1. User tries to access Application .
2. SiteMinder Policy Agent intercepts request and redirects to Federated Access Manager for login.
3. SiteMinder presents a login page to the user.
4. SiteMinder Policy Agent authenticates the user.
5. The request is directed to the OpenSSO Enterprise login page with the custom Authentication Module for SiteMinder.
The Custom Authentication Module validates the SiteMinder session and creates an OpenSSO Enterprise session.
6. The request is directed to the Application.
The OpenSSO Enterprise policy agent validates the OpenSSO Enterprise user session.
7. The OpenSSO Enterprise policy agent allows access to the Application.

FIGURE 13–2 Process Flow for Simple Single Sign-On with SiteMinder

Federated Single Sign-On

The SAML, ID-FF, and WS-Federation protocols provide cross-domain single sign-on among multiple trusted business entities. These protocols are also used in Identity Federation. Identity Federation involves an Identity Provider, also known as an authentication provider, and a Service Provider where the user authentication session at the Identity provider is consumed. The following are common use cases in which SiteMinder is enabled for federation protocols:

- Enabling SiteMinder for federation protocols in a Service Provider environment
- Enabling SiteMinder for federation protocols in an Identity Provider environment

Federated Single Sign-On in an Identity Provider Environment

This is the most common of the deployments. This is a good approach when you want to use Federation Access Manager for establishing partner relations and still leverage the SiteMinder authentication framework.

For example, as a company partners with external companies, the company deploys OpenSSO in the Service Provider environment to leverage the SAMLv2 Federation protocols. The following figure illustrates how SiteMinder can be enabled in an Identity Provider environment using Federated Access Manager for federation protocols.

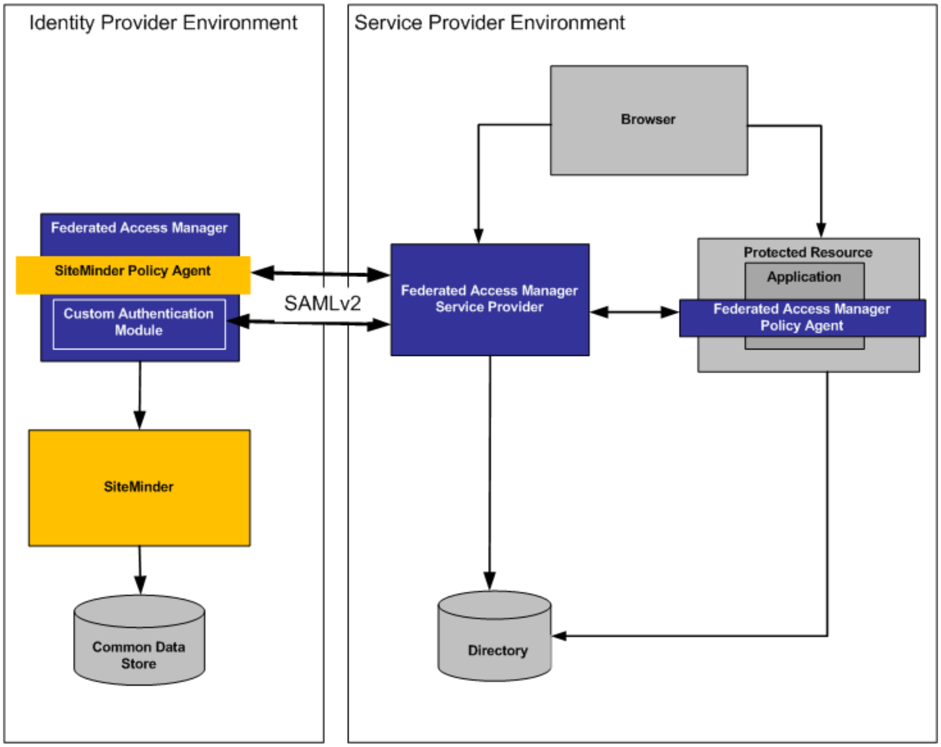

Identity Provider Environment
Service Provider Environment
Browser
Federated Access Manager
SiteMinder Policy Agent
Custom Authentication Module
SAMLv2
Federated Access Manager Service Provider
Protected Resource
Application
Federated Access Manager Policy Agent
SiteMinder
Common Data Store
Directory

FIGURE 13–3 Deployment Architecture for Federated Single Sign-On in an Identity Provider Environment

In this example, Federated Access Manager provides federated single sign-on among enterprise applications in partner environments, while SiteMinder continues to provide authentication. The following two figures illustrates a typical transaction flow.

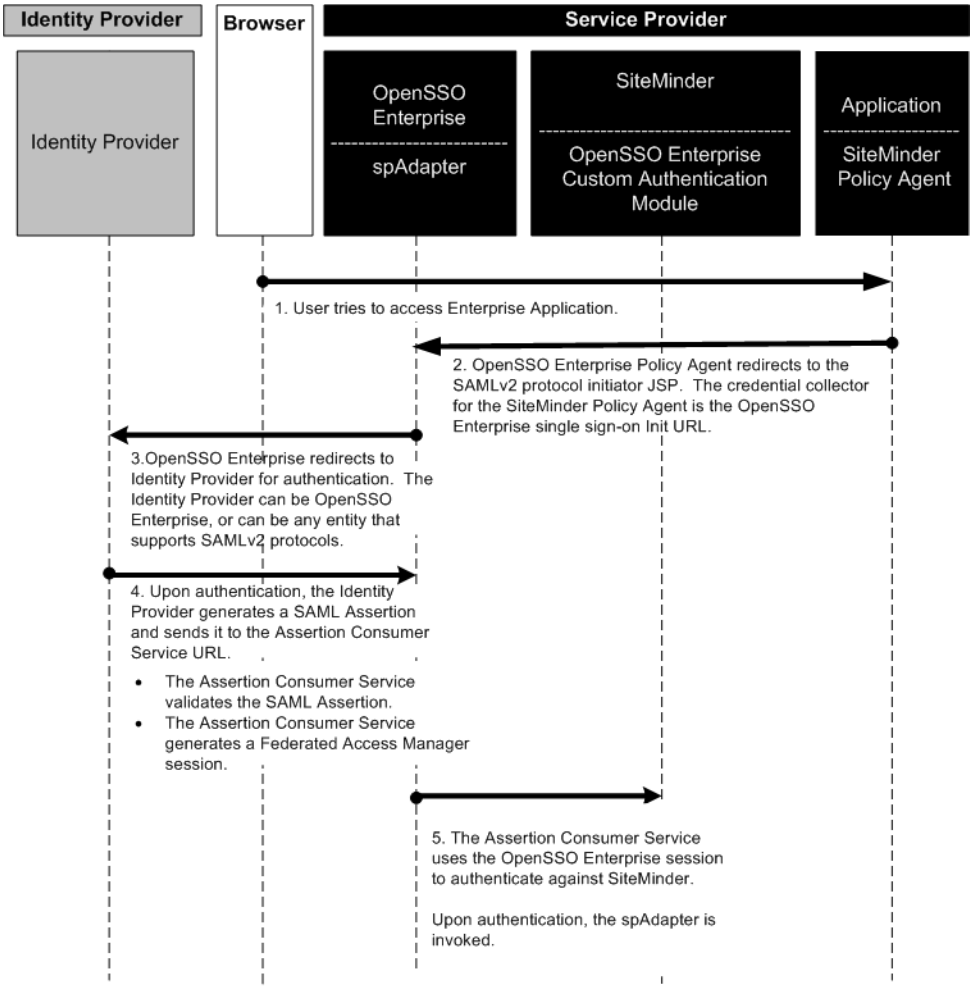

Identity Provider
Browser
Service Provider
Identity Provider
OpenSSO Enterprise
spAdapter
SiteMinder
OpenSSO Enterprise Custom Authentication Module
Application
SiteMinder Policy Agent
1. User tries to access Enterprise Application.
2. OpenSSO Enterprise Policy Agent redirects to the SAMLv2 protocol initiator JSP. The credential collector for the SiteMinder Policy Agent is the OpenSSO Enterprise single sign-on Init URL.
3.OpenSSO Enterprise redirects to Identity Provider for authentication. The Identity Provider can be OpenSSO Enterprise, or can be any entity that supports SAMLv2 protocols.
4. Upon authentication, the Identity Provider generates a SAML Assertion and sends it to the Assertion Consumer Service URL.
• The Assertion Consumer Service validates the SAML Assertion.
• The Assertion Consumer Service generates a Federated Access Manager session.
5. The Assertion Consumer Service uses the OpenSSO Enterprise session to authenticate against SiteMinder.
Upon authentication, the spAdapter is invoked.

FIGURE 13–4 Process Flow for Federated Single Sign-On in an Identity Provider Environment

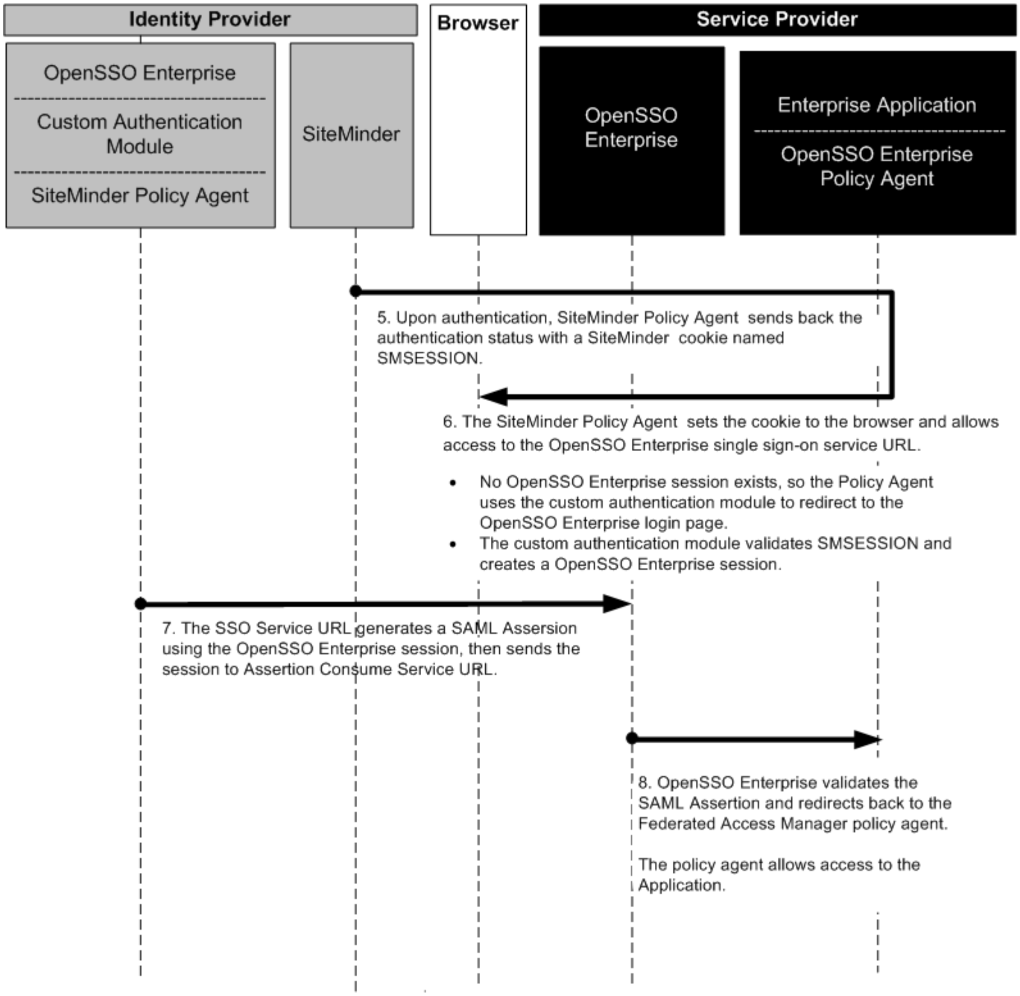

Identity Provider
Browser
Service Provider
OpenSSO Enterprise
Custom Authentication Module
SiteMinder Policy Agent
SiteMinder
OpenSSO Enterprise
Enterprise Application
OpenSSO Enterprise Policy Agent
5. Upon authentication, SiteMinder Policy Agent sends back the authentication status with a SiteMinder cookie named SMSESSION.
6. The SiteMinder Policy Agent sets the cookie to the browser and allows access to the OpenSSO Enterprise single sign-on service URL.
• No OpenSSO Enterprise session exists, so the Policy Agent uses the custom authentication module to redirect to the OpenSSO Enterprise login page.
• The custom authentication module validates SMSESSION and creates a OpenSSO Enterprise session.
7. The SSO Service URL generates a SAML Assersion using the OpenSSO Enterprise session, then sends the session to Assertion Consume Service URL.
8. OpenSSO Enterprise validates the SAML Assertion and redirects back to the Federated Access Manager policy agent.
The policy agent allows access to the Application.

FIGURE 13–5 Process Flow for Federated Single Sign-On in an Identity Provider Environment (Continued)

Federated Single Sign-On Use Case in the Service Provider Environment

In this example, the company uses SiteMinder in the Service Provider environment to protect legacy applications. OpenSSO Enterprise is installed solely to invoke Federation protocols. This deployment quickly enables partners (Service Providers) to establish federation environments with their trusted Identity Providers where the authenticates must be delegated.

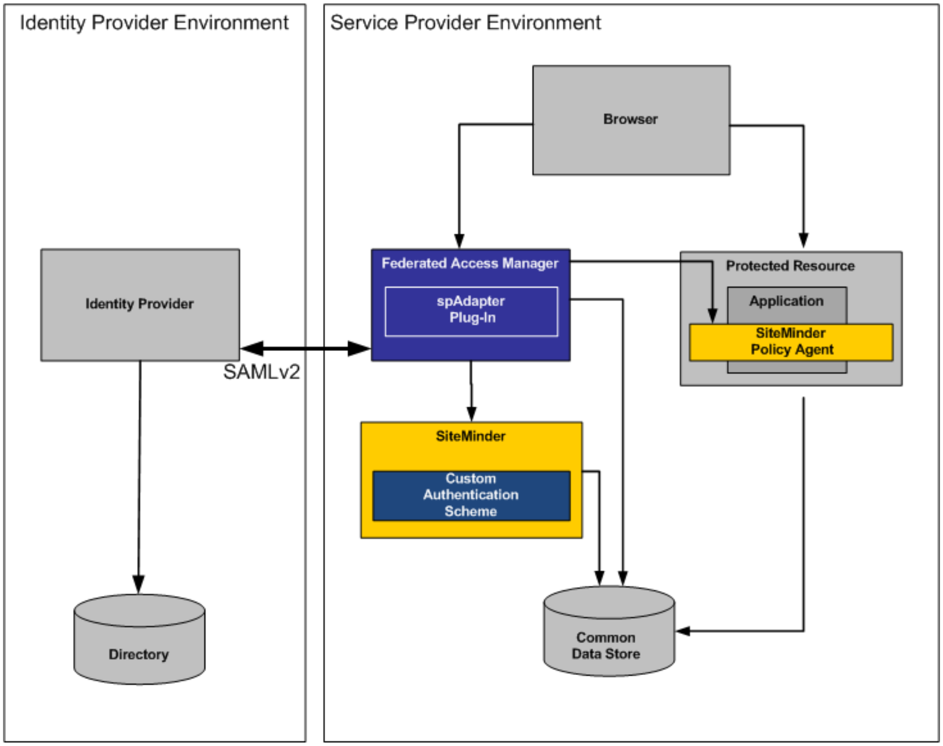


FIGURE 13–6 Deployment Architecture for Federated Single Sign-On In the Service Provider Environment

The following two figures illustrate the steps in the single sign-on flow:

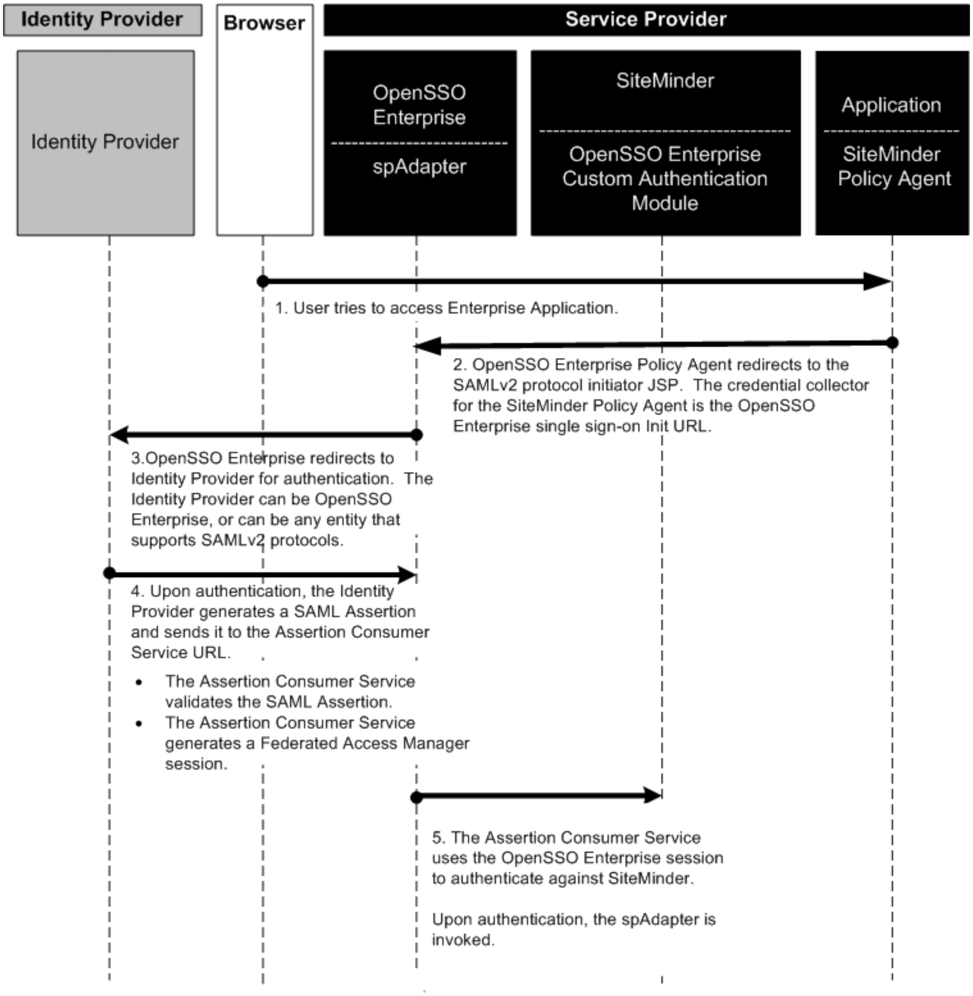

Identity Provider
Browser
Service Provider
Identity Provider
OpenSSO Enterprise
spAdapter
SiteMinder
OpenSSO Enterprise Custom Authentication Module
Application
SiteMinder Policy Agent
1. User tries to access Enterprise Application.
2. OpenSSO Enterprise Policy Agent redirects to the SAMLv2 protocol initiator JSP. The credential collector for the SiteMinder Policy Agent is the OpenSSO Enterprise single sign-on Init URL.
3.OpenSSO Enterprise redirects to Identity Provider for authentication. The Identity Provider can be OpenSSO Enterprise, or can be any entity that supports SAMLv2 protocols.
4. Upon authentication, the Identity Provider generates a SAML Assertion and sends it to the Assertion Consumer Service URL.
• The Assertion Consumer Service validates the SAML Assertion.
• The Assertion Consumer Service generates a Federated Access Manager session.
5. The Assertion Consumer Service uses the OpenSSO Enterprise session to authenticate against SiteMinder.
Upon authentication, the spAdapter is invoked.

FIGURE 13–7 Process Flow for SiteMinder Federation in the Service Provider Environment

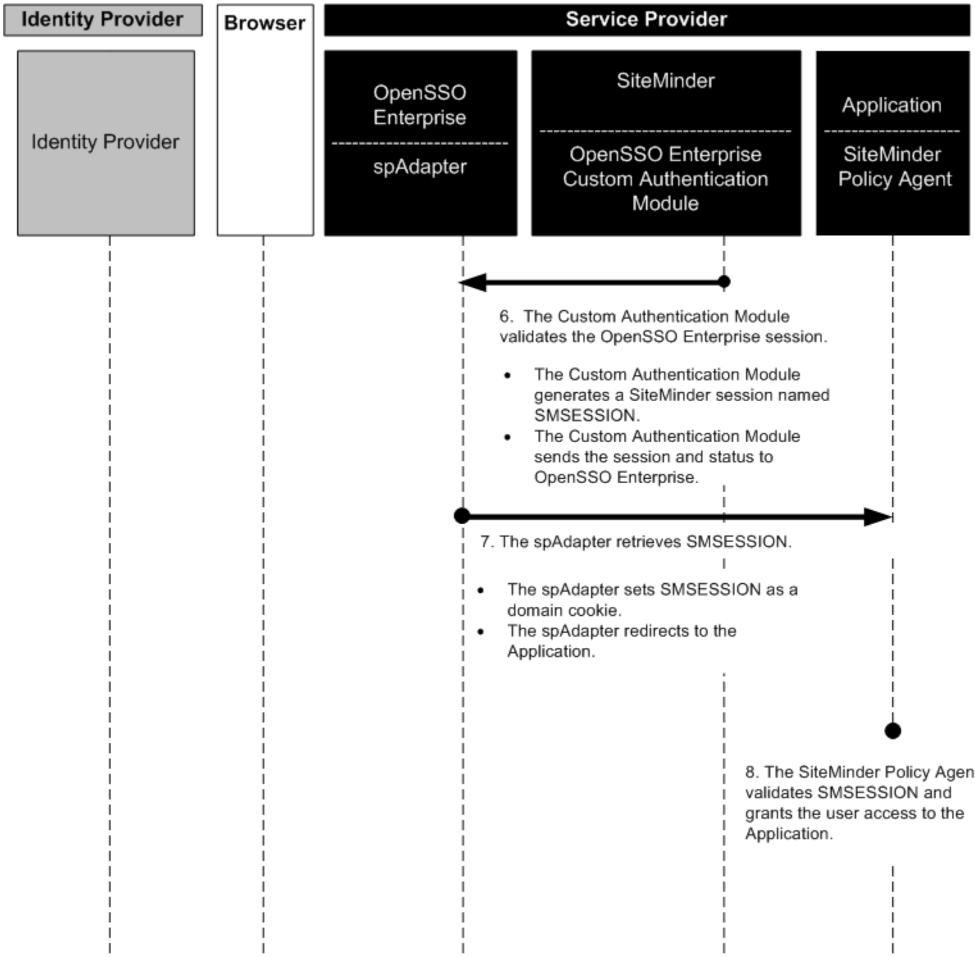

Identity Provider
Browser
Service Provider
Identity Provider
OpenSSO Enterprise
spAdapter
SiteMinder
OpenSSO Enterprise Custom Authentication Module
Application
SiteMinder Policy Agent
6. The Custom Authentication Module validates the OpenSSO Enterprise session.
• The Custom Authentication Module generates a SiteMinder session named SMSESSION.
• The Custom Authentication Module sends the session and status to OpenSSO Enterprise.
7. The spAdapter retrieves SMSESSION.
• The spAdapter sets SMSESSION as a domain cookie.
• The spAdapter redirects to the Application.
8. The SiteMinder Policy Agent validates SMSESSION and grants the user access to the Application.

FIGURE 13–8 Process Flow for SiteMinder Federation in the Service Provider Environment (continued)

Setting Up and Configuring Single Sign-On with SiteMinder and OpenSSO Enterprise

To co-locate both SiteMinder and OpenSSO Enterprise in the same federation environment, you must install the OpenSSO Enterprise server and OpenSSO Enterprise policy agents. The setup requires OpenSSO Enterprise 8.0 and the corresponding Policy Agents. Federation Access Manager is supported on various containers, however, you must choose a container where both Federation Access Manager and SiteMinder Policy Agents are supported.

The SiteMinder software is not available online, and you must have an account with Computer Associates to obtain the software. To validate this document, the following components were deployed in a lab environment:

- Sun's Federation Access Manager 8.0 (Early Access version)
- Sun Web Server 6.1 SP5
- Sun Directory Server 5.2 SP2
- CA SiteMinder Server 6.0 SP5
- CA SiteMinder Agent 6.0 for iPlanet Web Server 6.1
- CA SiteMinder SDK 6.0 SP5
- Custom codes (Bundled in the OpenSSO Enterprise zip)

The OpenSSO Enterprise bundle ships integration bits contained in the OpenSSO Enterprise WAR file. Instructions for configuring the authentication modules are described in corresponding README files. For more detailed steps for complete integration see the Chapter 2, "Integrating CA SiteMinder," in *Sun OpenSSO Enterprise 8.0 Integration Guide*.

Evaluating Benefits and Tradeoffs

As you design your deployment architecture, be sure to consider the benefits, tradeoffs. The following lists may help you determine if enabling federation using SiteMinder and OpenSSO Enterprise is appropriate to meet your business needs.

Benefits

- OpenSSO Enterprise allows you to continue using an existing SiteMinder deployment for authentication while leveraging the more advanced features of Federation Access Manager.
- OpenSSO Enterprise quickly enables federation protocols for SiteMinder without few changes to the existing infrastructure.
- OpenSSO Enterprise supports a variety of industry standard protocols such as SAMLv2 , ID-FF, ID-WSF, WS-Federation, XACML, WS-*, and others.
- OpenSSO Enterprise supports any generic LDAP repository for users, and can work with the existing SiteMinder database.
- OpenSSO Enterprise leverages its own configuration data store, which minimizes the need to migrate data from a different data store.

Tradeoffs

In general, when integrating any two access management products, you must consider the increased costs in resources and maintenance.

- When co-locating SiteMinder and OpenSSO, session management for both the products must be synchronized.
- Full integration requires you to set up session synchronization, possibly by using notification mechanisms effectively
- Administrators must be trained and proficient in the use of both products.

Finding More Information

See the Chapter 2, "Integrating CA SiteMinder," in *Sun OpenSSO Enterprise 8.0 Integration Guide* for detailed information about implementing single sign-on using CA SiteMinder and OpenSSO Enterprise.

CHAPTER 14

Enabling Single Sign-On Using Oracle Access Manager and OpenSSO Enterprise

This chapter describes options for co-locating Oracle Access Manager with Sun OpenSSO Enterprise in the same environment. For more detailed information about configuring end-to-end Oracle Access Manager single sign-on using OpenSSO, see the *Sun OpenSSO Enterprise 8.0 Integration Guide*.

The following topics are contained in this chapter:

About Oracle Access Manager

Oracle Access Manager (previously known as Oblix NetPoint and Oracle COREid) is an enterprise single sign-on product with many of the same features as Sun OpenSSO Enterprise and CA SiteMinder (previously known as Netegrity SiteMinder). Many companies have Oracle Access Manager already deployed and want to keep existing functionality even after installing OpenSSO Enterprise.

Analyzing the Deployment Architecture Options

Oracle has two solutions for web-based single sign-on. One solution is to use the legacy Oracle Access Manager single sign-on product, previously known as Oblix Access, which is integrated in the Oracle Application Server. This chapter focuses on this first solution.

Another solution is to use the Oracle Access Manager product with OpenSSO Enterprise. Oracle Access Manager is usually used for both single sign-on and delegated administration. This second solution is out of the scope of this document.

Oracle Access Manager and OpenSSO Enterprise typically co-exist in the following use cases:

- Simple Single Sign-On

  Major components are OpenSSO Enterprise, an OpenSSO Enterprise Policy Agent, a custom OpenSSO Enterprise authentication module, Oracle Access Manager, and Oracle WebGate.
- Federated Single Sign-On in an Identity Provider Environment

  Major components are OpenSSO Enterprise, an OpenSSO Enterprise Policy Agent, a custom OpenSSO Enterprise authentication module, Oracle Access Manager, and Oracle WebGate.
- Federated Single Sign-On in a Service Provider Environment

  Major components are OpenSSO Enterprise, a custom OpenSSO Enterprise authentication module, Oracle Access Manager, a custom Oblix plug-in, and Oracle WebGate.

Single logout for any these of these use cases can be implemented in many ways.

Logical architecture diagrams and process flow diagrams for these deployment options are described in the following section “Understanding the Business Use Cases.”

Considering Assumptions, Dependencies, and Constraints

This chapter describes the conceptual integration between the two products, OpenSSO Enterprise and Oracle Access Manager. In real deployments the use cases vary widely. In the deployment architecture diagrams above, the common data store is used between two products when they are co-located. The examples in this chapter focus primarily on mutual validation of user sessions. However, the same model can be extended to attribute exchange and other state information. For example, sessions can be managed independently. But and managing session timeouts are outside the scope of this document.

In the deployment examples in this chapter, the logout is assumed to be relatively simple and involves validating both OpenSSO Enterprise and Oracle Access Manager sessions as POST Logout processes.

For federated single sign-on, the examples in this chapter use SAMLv2 protocols. Similar functionality can be achieved using other federation protocols such as ID-FF, WS-Federation, SAML1 and so forth.

Understanding Typical Business Use Cases

The following use cases focus on single sign-on enablement and do not describe authorization options:

- "Simple Single Sign-On Use Case" on page 179
- "Federated Single Sign-On Use Cases" on page 181

Simple Single Sign-On Use Case

Simple single sign-on integration is useful when an Oracle Access Manager instance is already deployed and configured to protect intranet enterprise applications. Additionally, Federation Access Manager is deployed to protect the same intranet applications by honoring the user session obtained by Oracle Access Manager. In the following illustration, both Federated Access Manager and Oracle Access Manager share the same user repository for user profile verification. Federated Access Manager can also be configured to use the Ignore Profile option if it relies on the Oracle Access Manager session for attributes.

The following figure illustrates architecture in the simple single sign-on use case.

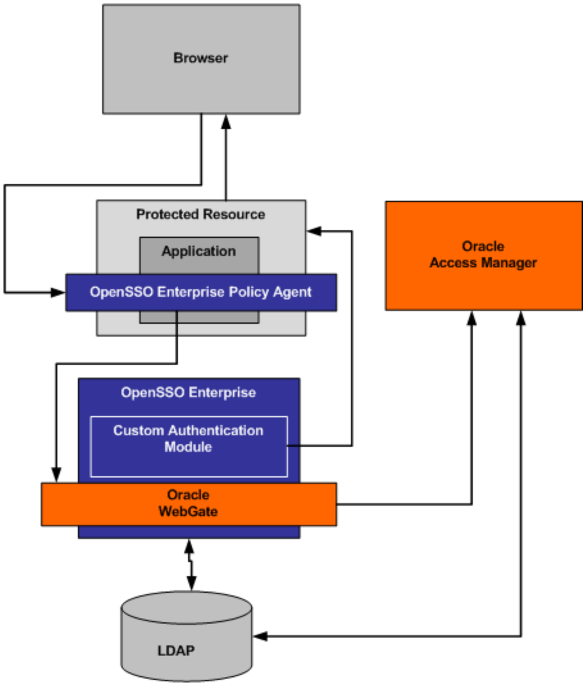


FIGURE 14–1 Simple Oracle Access Manager Single Sign-On

The following figure illustrates the process flow among components in the Identity Provider environment and Service Provider environment.

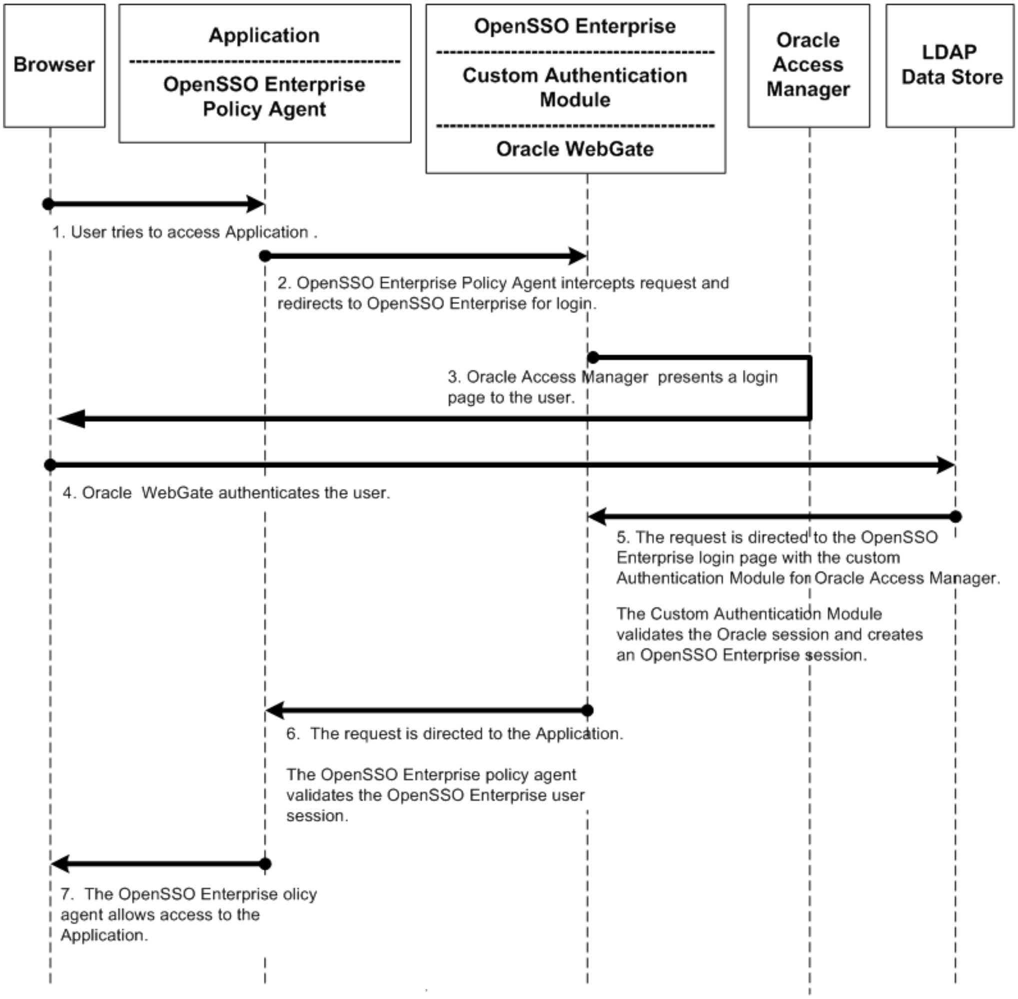

Browser
Application
OpenSSO Enterprise Policy Agent
OpenSSO Enterprise
Custom Authentication Module
Oracle WebGate
Oracle Access Manager
LDAP Data Store
1. User tries to access Application .
2. OpenSSO Enterprise Policy Agent intercepts request and redirects to OpenSSO Enterprise for login.
3. Oracle Access Manager presents a login page to the user.
4. Oracle WebGate authenticates the user.
5. The request is directed to the OpenSSO Enterprise login page with the custom Authentication Module for Oracle Access Manager.
The Custom Authentication Module validates the Oracle session and creates an OpenSSO Enterprise session.
6. The request is directed to the Application.
The OpenSSO Enterprise policy agent validates the OpenSSO Enterprise user session.
7. The OpenSSO Enterprise olicy agent allows access to the Application.

FIGURE 14–2 Process Flow for Simple Oracle Access Manager Single Sign-On

Federated Single Sign-On Use Cases

The SAML, ID-FF, and WS-Federation protocols provide cross-domain single sign-on among multiple trusted business entities. These protocols are also used in Identity Federation. Identity Federation involves an Identity Provider, also known as an authentication provider, and a Service Provider where the user authentication session at the Identity Provider is consumed. The following are common use cases in which Oracle Access Manager is enabled for federation protocols:

- Enabling Oracle Access Manager for federation protocols in a Service Provider environment

- Enabling Oracle Access Manager for federation protocols in an Identity Provider environment

Using Federated Access Manager to Enable Oracle Federation in an Identity Provider Environment

In this example, Oracle Access Manager is the authentication provider in an Identity Provider environment and protects some of the intranet applications. The Federation Access Manager in this deployment resolves the single sign-on issues among enterprise applications in partner environments while Oracle Access Manager provides authentication.

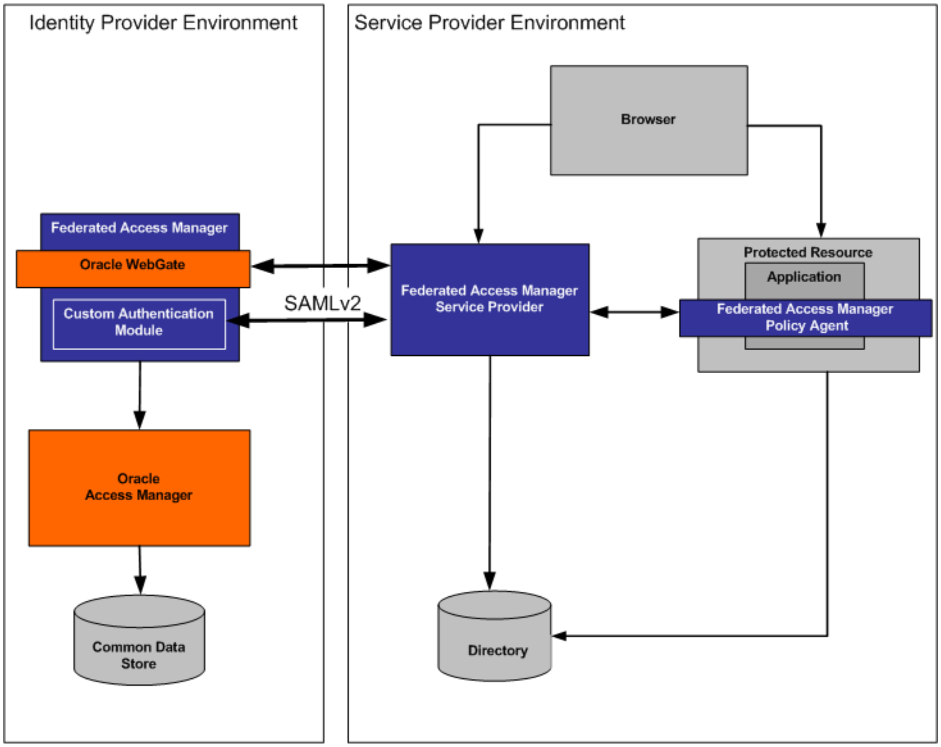

Identity Provider Environment
Service Provider Environment
Browser
Federated Access Manager
Oracle WebGate
Custom Authentication Module
SAMLv2
Federated Access Manager Service Provider
Protected Resource
Application
Federated Access Manager Policy Agent
Oracle Access Manager
Common Data Store
Directory

FIGURE 14–3 Oracle Access Manager Federation in an Identity Provider Environment

The following two figures illustrate the process flow among components in the Identity Provider environment and Service Provider environment.

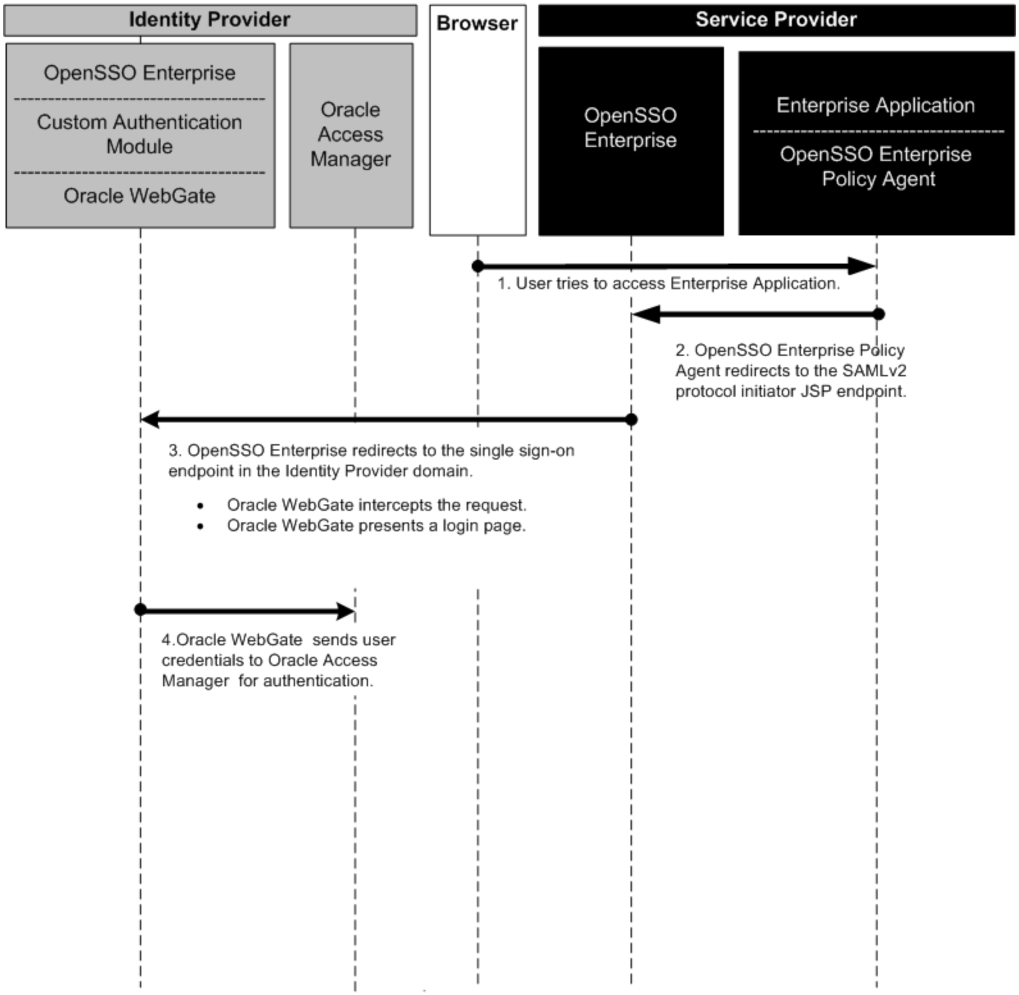

Identity Provider
OpenSSO Enterprise
Custom Authentication Module
Oracle WebGate
Oracle Access Manager
Browser
Service Provider
OpenSSO Enterprise
Enterprise Application
OpenSSO Enterprise Policy Agent
1. User tries to access Enterprise Application.
2. OpenSSO Enterprise Policy Agent redirects to the SAMLv2 protocol initiator JSP endpoint.
3. OpenSSO Enterprise redirects to the single sign-on endpoint in the Identity Provider domain.
• Oracle WebGate intercepts the request.
• Oracle WebGate presents a login page.
4.Oracle WebGate sends user credentials to Oracle Access Manager for authentication.

FIGURE 14–4 Process flow for Oracle Access Manager Federation in an Identity Provider Environment

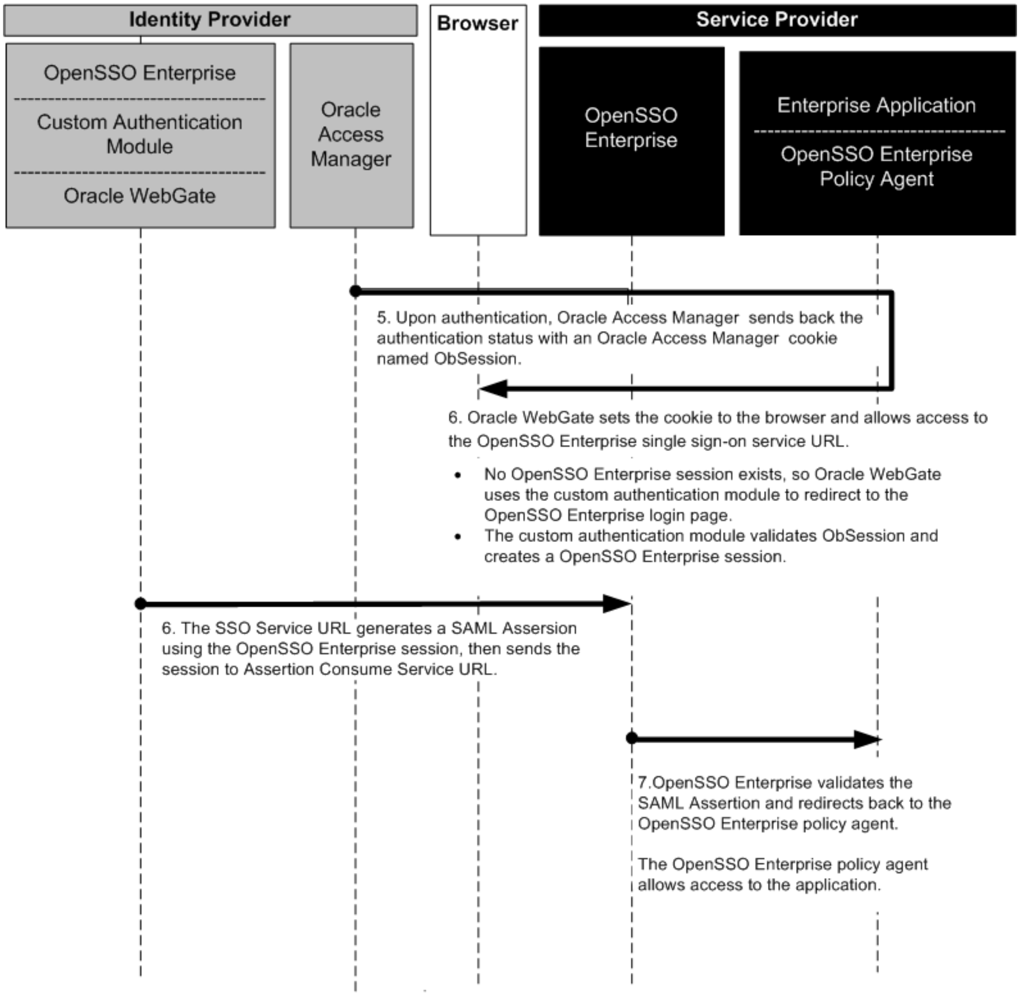

Identity Provider
OpenSSO Enterprise
Custom Authentication Module
Oracle WebGate
Oracle Access Manager
Browser
Service Provider
OpenSSO Enterprise
Enterprise Application
OpenSSO Enterprise Policy Agent
5. Upon authentication, Oracle Access Manager sends back the authentication status with an Oracle Access Manager cookie named ObSession.
6. Oracle WebGate sets the cookie to the browser and allows access to the OpenSSO Enterprise single sign-on service URL.
• No OpenSSO Enterprise session exists, so Oracle WebGate uses the custom authentication module to redirect to the OpenSSO Enterprise login page.
• The custom authentication module validates ObSession and creates a OpenSSO Enterprise session.
6. The SSO Service URL generates a SAML Assersion using the OpenSSO Enterprise session, then sends the session to Assertion Consume Service URL.
7.OpenSSO Enterprise validates the SAML Assertion and redirects back to the OpenSSO Enterprise policy agent.
The OpenSSO Enterprise policy agent allows access to the application.

FIGURE 14–5 Process flow for Oracle Access Manager Federation in an Identity Provider Environment (continued)

Using Federated Access Manager to Enable Oracle Federation in a Service Provider Environment

In this deployment, Oracle Access Manager is installed and configured in Service Provider Environment to protect legacy applications.

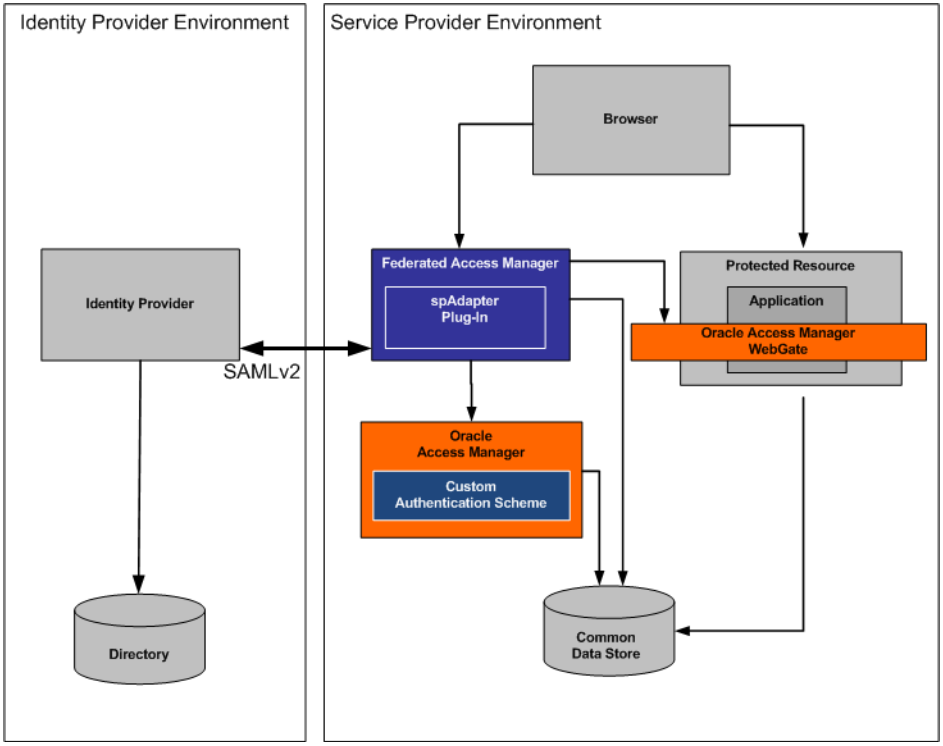


FIGURE 14–6 Oracle Access Manager Federation in a Service Provider Environment

The following two figures illustrate the process flow among components in the Identity Provider environment and Service Provider environment.

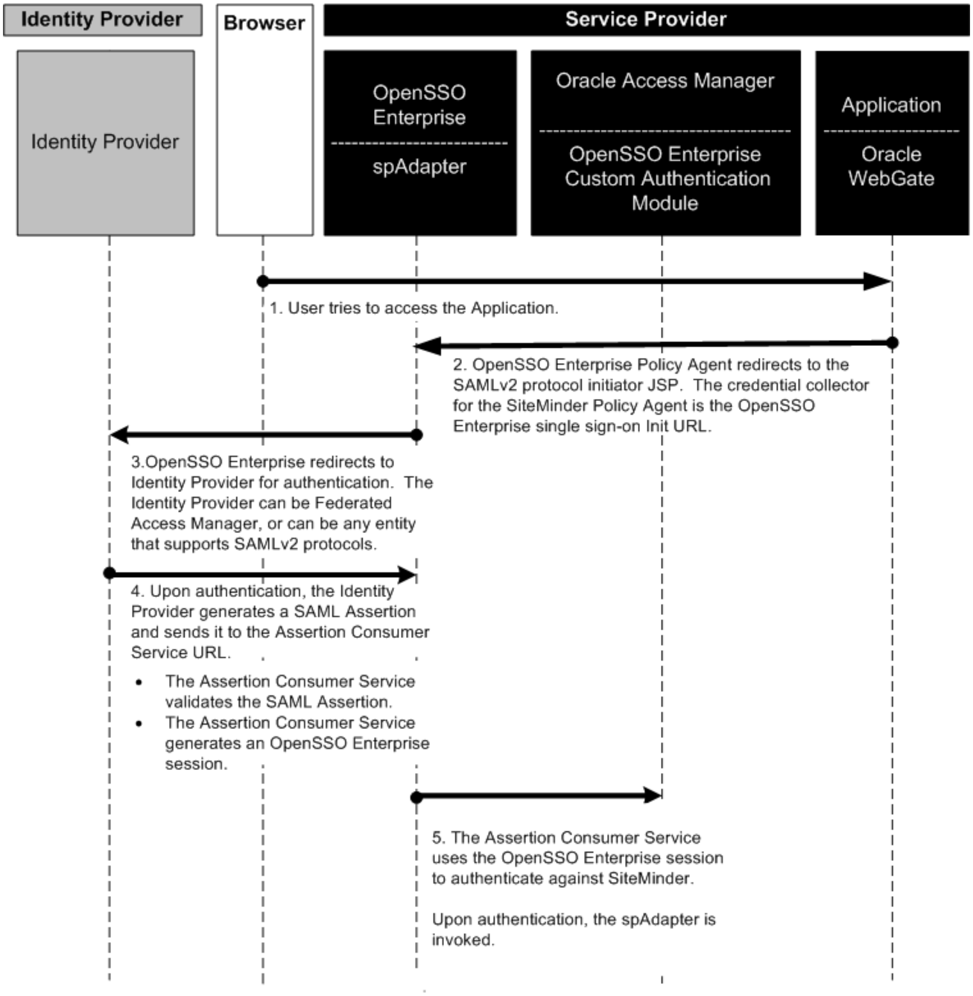

Identity Provider
Browser
Service Provider
Identity Provider
OpenSSO Enterprise
spAdapter
Oracle Access Manager
OpenSSO Enterprise Custom Authentication Module
Application
Oracle WebGate
1. User tries to access the Application.
2. OpenSSO Enterprise Policy Agent redirects to the SAMLv2 protocol initiator JSP. The credential collector for the SiteMinder Policy Agent is the OpenSSO Enterprise single sign-on Init URL.
3.OpenSSO Enterprise redirects to Identity Provider for authentication. The Identity Provider can be Federated Access Manager, or can be any entity that supports SAMLv2 protocols.
4. Upon authentication, the Identity Provider generates a SAML Assertion and sends it to the Assertion Consumer Service URL.
• The Assertion Consumer Service validates the SAML Assertion.
• The Assertion Consumer Service generates an OpenSSO Enterprise session.
5. The Assertion Consumer Service uses the OpenSSO Enterprise session to authenticate against SiteMinder.
Upon authentication, the spAdapter is invoked.

FIGURE 14–7 Process Flow for Oracle Access Manager Federation in a Service Provider Environment

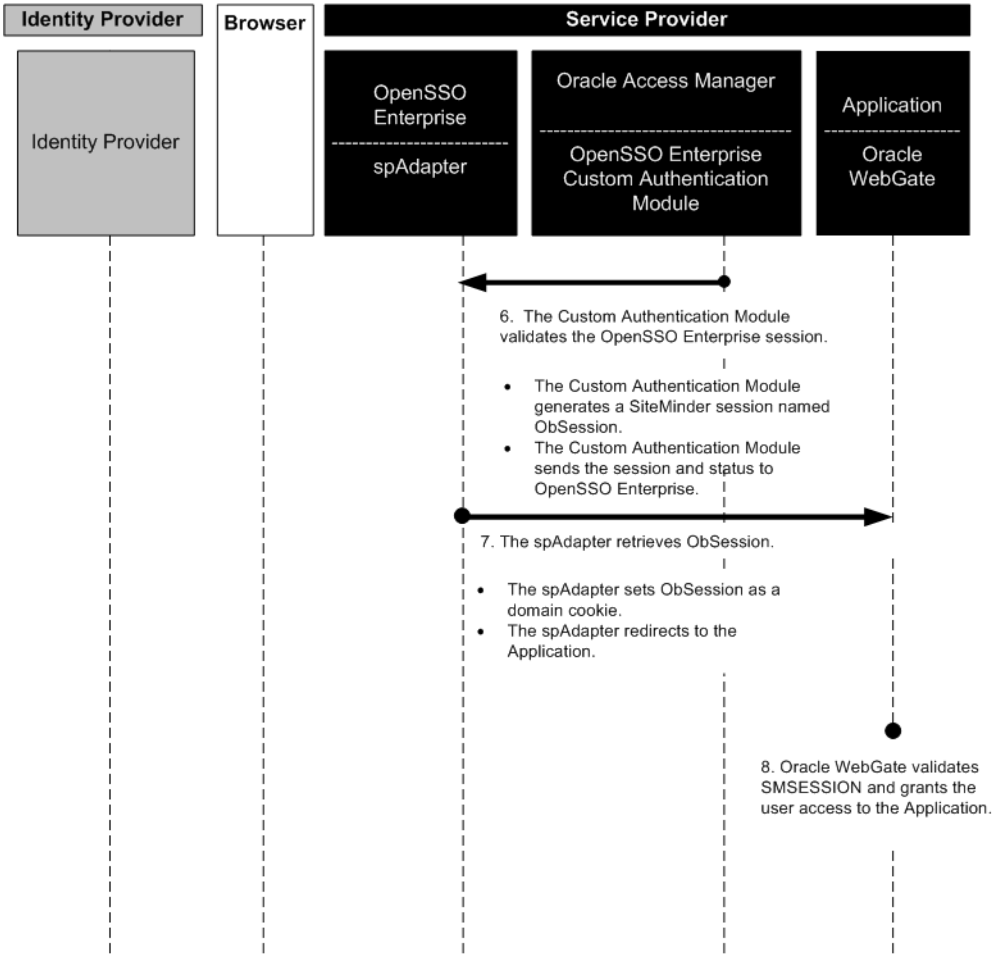

Identity Provider
Browser
Service Provider
Identity Provider
OpenSSO Enterprise
spAdapter
Oracle Access Manager
OpenSSO Enterprise Custom Authentication Module
Application
Oracle WebGate
6. The Custom Authentication Module validates the OpenSSO Enterprise session.
• The Custom Authentication Module generates a SiteMinder session named ObSession.
• The Custom Authentication Module sends the session and status to OpenSSO Enterprise.
7. The spAdapter retrieves ObSession.
• The spAdapter sets ObSession as a domain cookie.
• The spAdapter redirects to the Application.
8. Oracle WebGate validates SMSESSION and grants the user access to the Application.

FIGURE 14–8 Process Flow for Oracle Access Manager Federation in a Service Provider Environment (continued)

Setting Up and Configuring Single Sign-On Using Oracle Access Manager and OpenSSO Enterprise

The setup requires Federation Access Manager 8.0 and the corresponding Policy Agents. Federation Access Manager is supported on various containers, however, we have to choose a container where both Federation Access Manager and Oracle Access Manager Web Gate are supported. The Oracle Access Manager Software is available Online for temporary evaluation. For validation, this document used following software:

- Sun's Federation Access Manager 8.0 (Early Access version)
- Sun Web Server 6.1 SP5 Sun Directory Server 5.2 SP2
- Oracle Access Manager 10g (10.1.4.0.1)
- Oracle Access Manager Agents (Web Gate) 10g (10.1.4.0.1)
- Oracle Access Manager SDK 10g (10.1.4.0.1)
- Custom codes (Bundled in the OpenSSO Enterprise zip)

The Federation Access Manager bundle ships integration bits along with Federation Manager war file. The instructions on configuring the authentication modules corresponding README files. However, more detailed steps for complete described in Federation Access Manager Integration Document Guide.

Evaluating Benefits and Tradeoffs

As you design your deployment architecture, be sure to consider the benefits, tradeoffs. The following lists may help you determine if enabling federation using Oracle Access Manager and OpenSSO Enterprise is appropriate to meet your business needs.

Benefits

- OpenSSO Enterprise allows you to continue using an existing Oracle Access Manager deployment for authentication while leveraging the more advanced features of OpenSSO Enterprise.
- OpenSSO Enterprise quickly enables federation protocols for Oracle Access Manager with few changes to the existing infrastructure.
- OpenSSO Enterprise supports a variety of industry standard protocols such as SAMLv2 , ID-FF, ID-WSF, WS-Federation, XACML, WS-*, and others.
- OpenSSO Enterprise supports any generic LDAP repository for users, and can work with the existing Oracle Access Manager database.
- OpenSSO Enterprise leverages its own configuration data store, which minimizes the need to migrate data from a different data store.

Tradeoffs

In general, when integrating any two access management products, you must consider the increased costs in resources and maintenance.

- When co-locating Oracle Access Manager and OpenSSO, session management for both the products must be synchronized.
- Full integration requires you to set up session synchronization, possibly by using notification mechanisms effectively
- Administrators must be trained and proficient in the use of both products.

CHAPTER 15

Using the Embedded Configuration Data Store for OpenSSO Enterprise

This chapter is under development.

This chapter is under development.

This chapter is under development.