



FEDERATED IDENTITY MANAGEMENT

IDENTITY FEDERATION CONCEPTS WHITE PAPER



BUSINESS SOLUTIONS
TECHNOLOGY
OUTSOURCING

Dr. Alan Nagelberg
July 2010

ABSTRACT

Effective and secure management of authentication and identity information in a business enterprise is threatened by numerous forces. These include:

- Proliferation of applications being accessed daily
- Integration of Web services and technologies
- Interaction and use of data from multiple applications that may exist across networks
- Access to internal applications by business partners
- Access by employees to external applications, and the increased distribution of services among multiple contractors for ongoing maintenance
- Support of the applications and services

For users, management of multiple usernames and passwords has not only become a burden but has resulted in use of weak passwords, password reuse, and frequently, insecure storage of username/password pairs. Organizations face the additional burden of managing non-employee accounts in the absence of robust sources of identity information.

Identity federation is a standardized method for dealing with these issues. Several general approaches for establishing federation between entities have been developed: public key infrastructure (PKI), Security Assertion Markup Language (SAML), WS-Federation, Liberty Alliance, and various user-centric approaches (e.g., OpenID).

This document gives an overview of “provider-centric” identity federation used by organizations internally and for business-to-business (B2B) interactions; specifically, the use of the SAML, WS-Federation, and Liberty Alliance approaches.

The topics covered include:

- Describing general identity federation concepts
- Reviewing existing identity federation standards and specifications
- Describing the profiles and use cases that need to be handled under the SAML framework

Although “user-centric” federation concepts will be briefly introduced for purposes of comparison, a detailed business and technical discussion of that topic will be part of a subsequent white paper.

This paper is based on an internal white paper by Soren Thygesen Gjesse.

DOCUMENT DETAILS

Authors: Dr. Alan Nagelberg
Rebecca L. Daniels (ed.)
Scott L. Lewis (ed.)
Melvin Vaughan (ed.)

TABLE OF CONTENTS

1	Introduction.....	7
2	Identity Management Concepts	10
2.1	Identity	10
2.2	Identity Management.....	11
2.3	User Repositories.....	11
2.4	The Login Challenge	12
2.5	Adding Web Services and SOA.....	14
3	Identity Federation.....	15
3.1	Trust Between Parties	16
3.2	Single Sign-On in an Identity Federation.....	17
3.3	Identity Provider Discovery.....	18
3.4	Assertion Query	19
3.5	Attribute Definitions	20
3.6	Privacy and Pseudonyms.....	20
3.7	Name Mapping.....	21
3.8	Name Linking	22
3.9	Federation Termination	22
3.10	Single Logoff	23
3.11	Communication Channels and Artifact Resolution.....	24
3.12	Adding Web Services and SOA.....	26
3.13	Federation SSO vs. Traditional SSO.....	28
3.14	Federation Scenario.....	28
3.15	Adopting a Federation Standard.....	29
4	Summary	35

TABLE OF FIGURES AND TABLES

Figure 2.2-1. Identity Management Life Cycle	11
Figure 3.2-1. Service Provider-Initiated Access to an Application in a Federation	17
Figure 3.2-2. Identity Provider-Initiated Access to an Application in a Federation	18
Figure 3.3-1. Identity Provider Discovery	19
Figure 3.4-1. Assertion Query Service.....	19
Figure 3.6-1. Protecting Privacy Using Persistent Pseudonyms in Assertions.....	21
Figure 3.8-1. Name Linking	22
Figure 3.9-1. Federation Termination	23
Figure 3.10-1. Single Logoff in a Federation.....	24
Figure 3.11-1. SSO Utilizing Front-Channel and Back-Channel Communication.....	25
Figure 3.11-2. Federation Using an Enhanced Client or Proxy (ECP) Device	26
Figure 3.12-1. Relationships Among an STS, WSC, and WSP	27
Figure 3.12-2. Chaining	27
Figure 3.13-1. Traditional SSO: Using Multiple SSOs to Manage Application Access	28
Figure 3.14-1. Typical Federation SSO	29
Figure 3.15-1. Federation Standard Evolution	30
Table 3.15-2. SAML Specification vs. WS-Federation Specification	31
Figure 3.15-3. OpenID Sign-On Process.....	32
Figure 3.15-4. CardSpace Sign-On Process	33
Table 3.15-5. Comparison of Federation Protocols	34

BIBLIOGRAPHY

- 1) Nick Ragouzis *et al.* *Security Assertion Markup Language (SAML) V2.0 Technical Overview*. OASIS SSTC, March 2008. Document ID sstc-saml-tech-overview-2.0-cd-02. See <http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0-cd-02.pdf>
- 2) Eve Maler, *Web Services and Identity, 2.2 Federated Identity Technologies*, XML Summer School 2007. See: <http://www.xmlgrl.com/publications/fed-id-tech-27jul2007.pdf>
- 3) J. Kemp *et al.* *Authentication Context for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-authncontext- 2.0-os. See <http://docs.oasis-open.org/security/saml/v2.0/saml-authncontext-2.0-os.pdf>.
- 4) S. Cantor *et al.* *Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-bindings-2.0-os. See <http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf>.
- 5) P. Mishra *et al.* *Conformance Requirements for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID samlconformance-2.0-os. See <http://docs.oasis-open.org/security/saml/v2.0/samlconformance- 2.0-os.pdf>.
- 6) S. Cantor *et al.* *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-core-2.0-os. See <http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf>.
- 7) J. Moreh. *Errata for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, May, 2006. Document ID sstc-saml-errata-2.0-draft-nn. See <http://www.oasis-open.org/committees/security/>.
- 8) P. Madsen, *et al.* *SAML V2.0 Executive Overview*. OASIS SSTC, April, 2005. Document ID sstc-saml-exec-overview-2.0-cd-01. See <http://www.oasisopen.org/committees/security/>.
- 9) J. Hodges *et al.* *Glossary for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-glossary-2.0-os. See <http://docs.oasis-open.org/security/saml/v2.0/saml-glossary-2.0-os.pdf>.
- 10) T. Scavo, *et al.* *SAML Metadata Extension for Query Requesters*. OASIS SSTC, March 2006. Document ID sstc-saml-metadata-ext-query-cd-01. See <http://www.oasis-open.org/committees/security/>.
- 11) G. Whitehead *et al.* *Metadata Profile for the OASIS Security Assertion Markup Language (SAML) V1.x*. OASIS SSTC, March 2005. Document ID sstc-saml1xmetadata- cd-01. See <http://www.oasis-open.org/committees/security/>.
- 12) S. Cantor *et al.* *Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-metadata-2.0-os. See <http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf>.
- 13) S. Cantor *et al.* *Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-profiles-2.0-os. See <http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf>.
- 14) S. Cantor. *SAML Protocol Extension for Third-Party Requests*. OASIS SSTC, March 2006. Document ID sstc-saml-protocol-ext-thirdparty-cd-01. See <http://www.oasisopen.org/committees/security/>.
- 15) F. Hirsch *et al.* *Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-seconconsider-2.0-os. See <http://docs.oasis-open.org/security/saml/v2.0/saml-seconconsider-2.0-os.pdf>.
- 16) OASIS Security Services Technical Committee Web site, <http://www.oasisopen.org/committees/security>.
- 17) J. Hodges. *Show to Study and Learn SAML*. OASIS SSTC SAML Whitepaper, November 2006. See <http://identitymeme.org/doc/draft-hodges-learning-saml-00.html>

- 18) J. Hodges. *Technical Comparison: OpenID and SAML Draft 06*. OASIS SSTC SAML Whitepaper, January 2008. See <http://identitymeme.org/doc/draft-hodges-saml-openid-compare-05.html>
- 19) Ping Identity. *The Primer: Nuts and Bolts of Federated Identity Management*. 2008
- 20) R. Randall et al. *SAML Attribute Sharing Profile for X.509 Authentication-Based Systems*. OASIS SSTC, March 2006. Document ID sstc-saml-x509-authn-attrbprofile-cd-02. See <http://www.oasis-open.org/committees/security/>.
- 21) C. Morris et al. *SAML XPath Attribute Profile*. OASIS SSTC, August, 2005. Document ID sstc-saml-xpath-attribute-profile-cd-01. See <http://www.oasisopen.org/committees/security/>.
- 22) S. Carmody. *Shibboleth Overview and Requirements*. Shibboleth project of Internet2. See <http://shibboleth.Internet2.edu/docs/draft-Internet2-shibbolethrequirements-01.html>.
- 23) A. Nadalin et al. *Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)*. OASIS WSS-TC, February 2006. Document ID wss-v1.1-spec-os-SOAPMessageSecurity. See <http://www.oasis-open.org/committees/wss/>.
- 24) R. Monzillo et al. *Web Services Security: SAML Token Profile 1.1*. OASIS WSS-TC, February 2006. Document ID wss-v1.1-spec-os-SAMLSecurityProfile. See <http://www.oasis-open.org/committees/wss/>.
- 25) T. Moses, et al. *OASIS eXtensible Access Control Markup Language (XACML) Version 2.0*. OASIS XACML-TC, February 2005. Document ID oasis-access_controlxacml-2.0-core-spec-os. See <http://www.oasis-open.org/committees/xacml>.
- 26) D. Eastlake et al. *XML Encryption Syntax and Processing*. World Wide Web Consortium. See <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>.
- 27) D. Eastlake et al. *XML-Signature Syntax and Processing*. World Wide Web Consortium. See <http://www.w3.org/TR/xmlsig-core/>.
- 28) J. Sermersheim, L. Poitou. *Password Policy for LDAP Directories*, Network Working Group. Internet Draft, July 2005. See <http://www.faqs.org/ftp/Internet-drafts/draft-beheraldap-password-policy-09.txt>.

1 INTRODUCTION

Individual management of access to required applications and associated user data is cumbersome, time consuming, and expensive for both individuals and the enterprises supporting the applications. In most current IT enterprises, access to multiple required applications — internal and external to the enterprise — is typically managed individually for each application. This approach requires that application support personnel and the user themselves manage not only the individual application accounts, but also the corresponding access rights, passwords, and profile information. As a result, much of the data managed is duplicated and redundant.

In day-to-day work and personal interactions, a user's management of accounts, passwords, and profile data becomes cumbersome quickly. On any given day a user authenticates many times, most likely with different usernames and passwords for various applications. In addition, the personally identifiable information (PII) stored in each application can rapidly become dated, since changes made to the information in one application repository are not generally replicated and updated in the user's other repositories.

Scenario 1A highlights the challenges a typical business-to-business (B2B) user faces in day-to-day access of multiple applications.

Scenario 1A: Accessing Multiple Applications — Typical Challenges

Joe works as an order fulfillment representative in an organization that utilizes several applications to review and process customer orders. The workflow system assigns customer orders to the various representatives. When an order is entered, the workflow system generates and sends a summary of the order to the corresponding representative. To process his assigned orders and see more data associated with a specific order, Joe uses the workflow application. While the workflow system manages the work, it does not manage the inventory or allow new inventory to be ordered.

As a result, Joe typically starts his day by logging into three internal applications: the corporate e-mail system, workflow application/system, and order processing system. He selects an order from the workflow system, opens it, and begins to process it. As he processes the first order, he must use the order processing system to allocate stock to an order and manage the inventory. While allocating stock, he realizes that some items ordered are out of stock. To order the needed stock, Joe uses an external application on the supplier's Web site. Joe accesses the site with his account name and password, which are managed by the supplier. After he places the order, the supplier's application generates an e-mail for Joe, noting that one item is on back order. Unfortunately, as Joe's e-mail address has changed since his last visit to the supplier Web site, he does not receive the back order notice.

In this simple example, Joe was required to login to four separate applications to support one customer. In addition, because Joe uses the supplier site infrequently, he doesn't remember to update his e-mail address on the site. How would it be for our user, Joe, to work in a world where these different accounts are reduced into a few, or perhaps just one? In a federated scenario using the techniques described in this white paper, the process could flow much more smoothly.

Scenario 1B: Identity Federation

Joe starts his day by opening his workflow application to find his assigned orders. Joe is required to login to the application. When Joe is prompted to login, he is met with an enterprise login screen. After login to the enterprise, he can access his e-mail system, the workflow application, and order processing system — without having to separately authenticate to each one. Because he has authenticated within the enterprise, and the applications he needs to access all accept his enterprise login credentials, Joe continues working without interruption. And because Joe's company has federated — pre-established a trust relationship — with the supplier, Joe does not have to login to place an order on the supplier's site. Finally, when the supplier's site needs to send Joe the back order notice, it automatically sends an attribute query to Joe's home system so it will provide his current email address.

In Scenario 1B, the trust relationships established between the enterprise and the applications, either internal or external to the enterprise, establish a basic principle of identity federation: single sign-on (SSO). The primary difference between traditional SSO and federated SSO is in the last step, with Joe's company login being valid on the supplier's (external) Web site. The company's use of international standards to communicate identity and authentication information to the supplier's Web site makes federation with SSO possible.

In Scenario 1B, the different applications are part of an **Identity Federation**, the term used when organizations form trust relationships where identities or assertion of an identity can be shared by all applications within the federation. It is critical that business partners involved in a federation build a trust relationship with one another. This trust relationship, defined by business, technical, and legal agreements, describes the applications that will be involved, the user profile information that will be shared, and the responsibilities of all parties to manage and protect user identities.

In the two scenarios, some of the benefits of federation are highlighted. Forming federations provides benefits to enterprises as well as individual users.

Reducing the number of identities in use has several benefits:

- Enhanced user experience via elimination of requirement to login multiple times during performance of a single workflow task
- Decreased number of user accounts and passwords that need to be remembered by the user and fewer user accounts that need to be managed by the organizations and enterprises participating in the federation
- Increased security, as fewer accounts are managed by a user. A strong password, biometric attribute, token, or one-time password (OTP) can be chosen and remembered (not written down)
- Decreased helpdesk costs related to forgotten password resets
- Reduced need to manage profile information in multiple locations and associated profile issues, as a result of sharing of personal data among applications. Federation minimizes the proliferation of personal identity data by providing assurance of a user's identity to a federation partner, as opposed to the user's having to provide and manage identity data for each application

Forming federations also enhances opportunities for businesses:

- New application and partner integration is more easily accomplished with shared identities managed in one place
- Sharing information across alliances can facilitate improved customer service

Identity federation is applicable in situations beyond the normal B2B relationships where enterprises have been most concerned with restricting user access to ensure data security. When considering relationships between individuals and business-to-consumer (B2C) relationships, additional federation scenarios become apparent. Scenario 2 illustrates federated identity concepts involving consumers using an online social network application and retailers, to highlight the additional benefits for B2C relationships.

Scenario 2: Federated Single Sign-On (SSO)

Mike frequently accesses a social network to interact, keep in touch, and share information with his friends and colleagues. This social network uses an Internet identity verification service to authorize access. Mike logs in to the Internet identity service to gain access to the social network. The social network reviews his credentials, accepts the login, and grants access. One of Mike's friends has posted a message recommending some new music that he may find entertaining.

Mike listens to the music clip and decides he'd like to purchase it through one particular online music retailer that he frequently uses. To increase business, the retailer has offered incentive discounts for members of Mike's identity service. The last time Mike used the retailer's site, he updated his profile on it and included his identity service information. Because Mike purchases music through an online retailer that uses the same Internet identity service as the social network, he is able to complete his purchase transaction on the site without additional authentication. And because Mike had previously updated his profile on the site, when he re-

entered the site this time, his identity provider status information was automatically collected — making it possible for him to receive a bonus: a discount on his purchase.

In Scenario 2, the federation's use of Federated SSO eliminated additional sign-on requirements. The user logged in using a service that then presented the user's login credentials to the different applications. However, going beyond basic SSO, the federated applications not only use information from the login credentials, but they also exchange information about the user that is not known by the user's identity service. The exchange of this additional information is possible because of the trust relationship pre-established between the applications.

This scenario illustrates a "user-centric" approach to federation, where the individual plays an active role in defining the trust relationship between the identity service and the application. In Scenario 2 — in contrast to Scenario 1B — the individual specifies the applications with which they want to federate and what private information can be shared with each application. Permitting a user-defined trust relationship is a significant "leap of faith" for many enterprises — the underlying processes by the identity provider to verify, proof, and maintain identities is a significant hurdle.

Discussion of these detailed actions related to "user-centric" federation and privacy will be the topic of a future white paper.

Implementing a federation not only improves user experience, but it also provides additional benefits for an organization:

- Reduces personnel requirements and costs associated with solution deployment and maintenance for account and access management
- Improves efficiency by reducing or eliminating time required to log into individual applications
- Increases competitive advantage through the enhanced communication and information exchange between business partners, suppliers, and customers
- Improves security and privacy and decreases the risk of identity theft by reducing the propagation of personally identifiable information
- Improves security and compliance management by demonstrating secure systems access control through a standardized approach to online identities between partners or applications

This white paper is an overview of identity federation from a general perspective of a B2B relationship where the trust relationships are determined by the businesses. It focuses on emerging approaches, use cases, and international standards. By implementing the standards, it is possible for an organization to dramatically decrease the number of identities they manage and for the user to decrease the number they must maintain and remember.

2 IDENTITY MANAGEMENT CONCEPTS

This section presents an overview the basic concepts of identity and identity management (IdM).

2.1 IDENTITY

An *identity* is established by a set of information (*identity attributes*) by which an individual is definitively distinguished within a context, such as an application. Identity attributes comprise physiological attributes, biographical information, issued credentials, and “secret” information, such as historical knowledge. Because an individual is initially registered with an application as a *user*, the application administrator will associate some of these identity attributes with the application *username*, to be subsequently used as authentication methods. On an ongoing basis, the user will be asked to authenticate prior to a transaction, such as log-on, single sign-on (SSO), or initiation or acknowledgement of a one-time transaction such as a financial trade. Once the individual has authenticated as the valid user, the application will determine whether the user is *authorized* to complete the transaction, based on their application rights and privileges.

Some examples of identity attributes:

- Physiological attributes
 - Fingerprint, voiceprint, retinal pattern, or facial features
- Biographical information
 - Name, birth date, place of birth, address
- Issued credentials
 - Passport, driver’s license, health card
- “Secret” information
 - Password, mother’s maiden name, favorite place

A composite of these identity attributes constitutes an identity, with each specific piece providing some level of a distinguishing characteristic. Over time, additional information further defines different aspects of an individual's identity.

As discussed earlier, a user will be requested to authenticate that they are who they claim to be, and if they are valid, the application will assess whether they are authorized to access the system. Authorization is based on different aspects of the user's identity such as group memberships, organization, and specific granted application rights

2.2 IDENTITY MANAGEMENT

To authenticate an individual user's identity, an application requires access to a secure and trusted repository where identity attributes are stored. The repository can be locally established within the application, or preferably, the application may read this information from a shared repository. Managing a portfolio of applications that use a variety of user repositories can quickly become a user management nightmare due to the effort required to synchronize all of the repositories when user attribute data change. Establishing IdM tools reduces the complexity of managing users by providing central administration of identities across multiple applications. **Figure 2.2-1** illustrates the basic identity management life cycle.

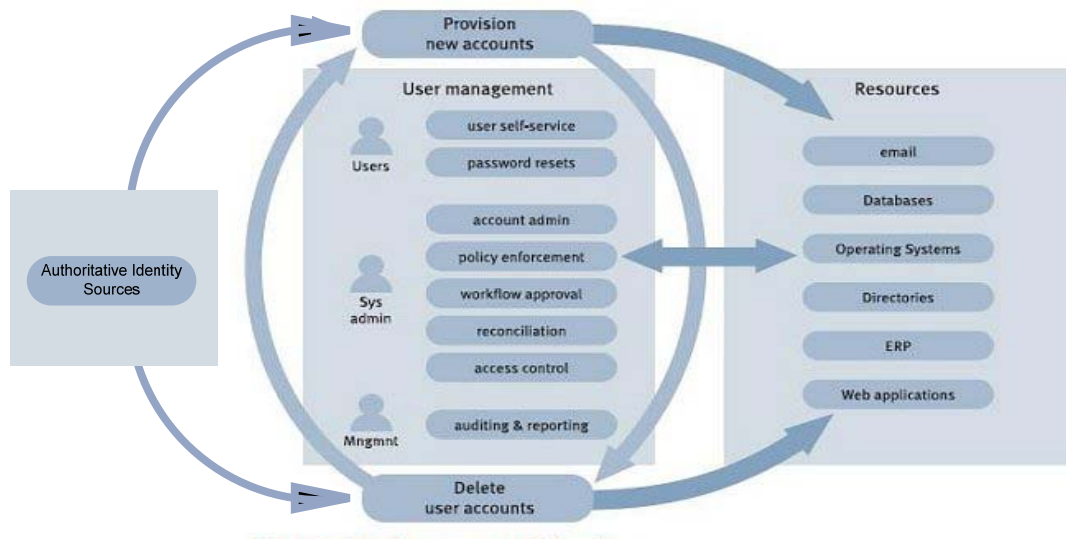


Figure 2.2-1. Identity Management Life Cycle

User management activity occurs in the Identity Management System (IdMS), depicted in the center of the figure. The resources accessed by the individual user are illustrated on the right side of the figure. When a user's information changes in an authoritative source or directly in the IdMS, the IdMS updates the identity repositories. To ensure the most efficient user management process, data should only be changed in the authoritative source(s) and then propagated to the resources by the IdMS, as needed. A resource should store only the minimal necessary information for a particular user. However, in reality, some applications manage and allow changes of user data directly in their own local identity repositories; therefore, they implement processes to synchronize these changes back to the authoritative stores. This "back-propagation" of changes can also be managed by the IdMS.

2.3 USER REPOSITORIES

The master authoritative repositories — with all the identity information used to update individual resource identity information — are a key component of an IdMS. The primary repository that contains information on all users within a business organization is typically either an HR system database or an enterprise directory. Using an LDAP directory is a preferred solution, given its designed optimization for a large volume of read requests. In many cases, a moderate level of user verification and proofing is required before an employee can be entered into the authoritative HR database or enterprise directory. Other types of users, such as contractors, may also be maintained in these repositories. The level of rigor for verifying and proofing these users should be considered when first designing the trust relationship.

With an enterprise-wide directory in place, access to all applications as a shared repository is possible. Applications can then obtain identity information from a central directory. Parts (or all) of the application-specific identity repositories can be replaced, and the applications can start to rely on information from the directory. This will simplify the IdMS' work, as the focus moves from keeping

individual repositories updated to managing the central directory. Although LDAP is a standard, this is not an easy process: LDAP defines only how to communicate with the directory and how to define the schema for the directory information, *not* the exact structure of the information in the directory. The structure is defined partly by what the chosen directory product mandates and partly by how the organization chooses to organize the directory. Applications that support identity information in a directory cannot just plug into any directory; additional configuration or customization will be needed for each application to use the directory.

The introduction of IdMSs and central directory services is a first step in moving applications toward a centrally managed, common repository. This will not be pure plug-and-play, as the information models in directories may differ between implementations. However, as more applications start to rely on the directory, the identity management process will grow simpler and the content of the local repositories significantly reduced.

2.4 THE LOGIN CHALLENGE

One of the primary goals of federated identity management — but also one of the more difficult technical issues — is ensuring synchronization of the user's password. Because passwords must be securely managed and difficult to “steal,” they are typically hashed or one-way encrypted. Distribution and exposure of a password across systems is intentionally made difficult.

The initial concept of login is simple. The application displays a user interface, requesting the user to enter username and token. The token can be simply a password or other methods such as a one-time password (OTP) from a physical device, a biometric attribute, PKI certificate, or combination of methods. If the username and token combination matches with the values securely stored by the application (or matches with the content of the directory), the user is authenticated and allowed access. In the remainder of this white paper, the phrase “username and password” will be used to refer to the login process.

Password Policies

To increase security, *password policies* are usually applied when a user establishes a password; these increase the complexity of system login processes. Following are a few typical password policy rules:

- **Temporary Password/Forced Change Upon Login:** If a temporary password is given to a user, either as an initial password or after administrator reset, a user may be forced to change the password upon their next login to the system.
- **Password Maximum Age/Password Expiration:** If a system has a maximum age parameter for a password established, a password will expire beyond the expiration date/age, forcing a user to change the password at next login.
- **Password Minimum Age:** A password may have a “minimum age”, disallowing a user to change a password too frequently.
- **Password Complexity/Strength:** To make it difficult for hackers to break a password, a password may be required to have complex quality/strength constraints.
- **Password History:** A system may store previously used passwords, disallowing a user from reusing a password.

With each password policy check, logging into a system becomes more complex. When a user logs into an application, the system checks the password policies to determine if a change is required. If so, before the new password is updated, the system must make further checks to ensure the password complies with all established policies. If a history policy is established following the change, the password history is automatically updated.

Login is further complicated when additional levels of authentication e.g., two- or three-factor authentication is employed using certificates, tokens, biometrics, etc. To effectively leverage two-or three-factor authentication, each application must be equipped to support the additional authentication complexity. Therefore, for a single username and password to work effectively when login is distributed across applications, all applications sharing the password will need to manage passwords according to the same set of policies. In a best-case scenario, all the password policy information is

available from a single directory, and all applications have been designed or modified to use this authentication system.

Consistent use across platforms and systems implies uniformly established practices and standards. Unfortunately, there are no current standards for storing password policy information in a directory. The documentation that came closest to a standard is an expired Internet Draft²⁸ that has been implemented by some directory products. Therefore, ensuring that all applications adhere to the same password policies is a complicated task: It requires that the exact same login logic be in place in all applications and that the policy information be shared.

Because the login process is not as simple as it might seem, at first, handing the login process off to a single component shared by applications makes good business sense. The login component becomes its own system, where it becomes responsible for supporting the different authentication forms and for enforcing the required policies. When an unauthenticated user tries to access an application, instead of managing the login, the application will instead hand login over to the control of the login component. The login component performs the verification and hands control back to the application, together with information on the authenticated user.

Achieving SSO

A process where a single system or service handles user authentication, not the application, is a process that supports achieving the single sign-on (SSO) goal. There are two types of approaches for achieving SSO: enterprise SSO (eSSO) and WebSSO. eSSO focuses on enabling SSO within an enterprise, including both Web and legacy applications. eSSO establishes a 'password safe' for each user to store all their username/password combinations. The password store can be unlocked and accessed using a single authentication. The username/password combinations are used to auto-populate the application login page transparently when accessed.

WebSSO is supported using Access Management applications by centrally managing user sessions and authentication status. WebSSO solutions focus on URL-based application access but different vendors support other applications, for example UNIX, to different extents. Prior to the development of federated identity management (FIM) standards, WebSSO session status was managed by proprietary session identifiers unique to each vendor's implementation. Thus, the user's session state established by one enterprise was not easily transferable to another enterprise or even to other internal organizations. Complete SSO is not achieved until a mechanism is developed to securely manage user sessions and securely distribute assertions to each application, organization, and enterprise.

SSO is available in many forms, both as commercial products and open source projects. All major vendors of security and IdM products have SSO in their portfolio. With an IdM and SSO solution from the same vendor, the SSO component has direct integration into the directory used by the IdM component — thereby providing integrated management of identity information.

For Web applications, SSO is especially attractive. It is technically easy for an SSO solution to intervene between the browser and the Web application. Whenever login is needed, the SSO can take over and request authentication from the user. The SSO stores the user's login and will use it for all Web applications managed by the SSO component.

For workstation client applications, many SSO solutions are based on support from the operating system. When a user is logged into a workstation, all client applications are running on the same operating system and within the security context of the active user. Using operating system services, the client application can retrieve information on the active user. It only becomes difficult for an application to participate in an SSO schema when a client application is not able to utilize the operating system services. In the case where the client application is unable to use SSO, the client application will prompt the user for username and password.

Today, many organizations provide access to their applications for internal users as well as for their business partners and customers. Typically, access for external business partners and customers is provided through extranet or portal solutions. Effectively managing the growing number of users from external organizations is an additional oversight burden for an IdM process.

2.5 ADDING WEB SERVICES AND SOA

With the introduction of service-oriented architecture (SOA), applications are getting more fragmented and dependent upon one another. Historically, most applications were designed to independently manage user authentication and all required business and security logic. With data transmission and integrated systems on the rise, applications are now interdependent, with more systems using shared business logic or data. Web services are now leveraged to expose business logic for consumption by and between applications. The communication across applications can be made directly from one application to another or via an intermediate system such as enterprise service bus (ESB), business process manager (BPM), or another Web service.

With the use of Web services, user actions are no longer confined to operations within a single application. Without additional individual logins, direct interaction, or knowledge that information is exchanging between systems, a system can use Web services to expose data and business logic for a user (or application) when needed. To ensure a secure exchange, the applications accessed through services will need to securely know both the identity of the actual user requesting the service and the identity of the user on behalf of whom the request is being made. An important part of any SOA architecture is defining the security architecture for passing user information through the various levels of service invocations.

3 IDENTITY FEDERATION

Managing and handling identities and the flow of information securely in a typical enterprise model become more complex with the following:

- Providing availability to more applications for more users
- Granting and managing access for external users
- Interfacing and exchanging information across applications within a single operating system
- Interacting and exchanging information and data between internal and external systems and across operating systems
- Increasing dependency upon Web services and SOA

Identity federation introduces strategies for effectively and securely handling identity information and application data in an enterprise-wide model. The term *federation* means “united in an alliance”. Thus an “identity federation” is a group of members who form a union to collaborate on identity information.

There are two distinguishable roles in a federation: The *identity provider* (IdP) and the *service provider* (SP)¹. The IdP manages the identity and associated information of their employees and “close” contacts, while the SP is responsible for accepting the authentication assertions and managing the accessibility and security requirements for a needed service. In a federation, the IdP provides the identity, user authentication, and authentication information or assertion to another organization (the SP) where the service is needed. The IdP is also commonly referred to as an *asserting or assuring party* because it asserts identity information about a user. The asserting party validates the identity through the authentication process and packages the identity information required for the federation from the available authoritative data sources. Once the user is authenticated within their IdP organization/network, that user can access the SP’s resource(s) without re-authentication. The SP is also often referred to as a *relying party* — because the SP accepts and trusts the identity information provided and allows access to the needed resource(s).

Identity Provider (IdP) — aka Asserting/Assuring Party
Service Provider (SP) — aka Relying Party

There are two major approaches to establishing federation; which approach is used depends on who has ownership of specifying the trust model between entities. This white paper focuses on “provider-centric” federation relationships used for a majority of business to business relationships.

- **Provider-centric defined federation:** This approach focuses on the interactions between different organizational entities — federation begins with a circle of trust, a group of IdPs and SPs who contractually agree to processes for an exchange of authentication information. [See Section 3.1 of this white paper for a detailed discussion of the topic.] Each circle of trust must include at least one IdP that manages identity data and provides authentication services and an SP that supplies resources to the users managed by the IdP. The “provider-centric” approach is considered more secure, since consistent policies can be enforced across the entire user community and policy enhancements easily implemented.
- **User-centric federation:** The rapidly expanding use of the Internet for individual- (user-) driven demands requires a brief overview of alternate federation approaches designed to address the associated “user-centric federations.” In this type of federation, the user can specify who they are willing to federate identities with and authorize the specific PII data to be shared with the federated organizations. Because the Internet is now one of the primary vehicles for the types of interactions represented by identity-defining information, people are creating online identities specific to the businesses with which they interact. By creating a user account with an identifier and password, an e-mail address, personal preferences (such as style of music, or opt-in/opt-out marketing decisions), and other information specific to the particular business (e.g., a bank

¹ Alternate names are used by different vendors, environments, or protocols. The identity provider role is also known as the asserting or assuring party and the service provider role as a relying party. The Microsoft ADFS system using WS-Federation uses the terms account provider and resource provider.

account number or ship-to address), a user is able to distinguish their account from others who also use the enterprise's services.

This distinguishing information is referred to as a *local identity* because it is specific to the SP for which it has been defined. Sending and receiving e-mail, checking bank balances, finalizing travel arrangements, accessing utility accounts, and shopping are just a few online services for which a user might define a local identity. If a user accesses all of these services, many different local identities have been configured.

Given the number of service providers for which a user can define a local identity, accessing each one can be time-consuming and frustrating. It would be advantageous to interconnect the identities across different entities since most local identities are independently configured (and fragmented across the Internet). For example: For easier online payments, a user's local identity with a bank and that with a utility company could be securely connected. This virtual phenomenon offers an opportunity for a system in which users can *federate* these local identities. *Identity federation* allows the user to link, connect, or bind the local identities that have been created for each service provider. The linked local identities, referred to as a *federated identity*, allow the user to login to one service provider site and click through to an affiliated service provider without having to re-authenticate or re-establish identity; this is, in effect, single sign-on (SSO). As discussed previously, *user-centric* federation and trust relationships add additional policy and administrative challenges above those required for *provider-centric* federation. User-centric relationships are currently in development and an active discussion topic within the identity community².

3.1 TRUST BETWEEN PARTIES

Typically, the relationship between IdPs and SPs can be considered a "many-to-many" relationship, i.e., an IdP can have a relationship with multiple SPs, and an SP can provide services to more than one IdP. The relationship that exists between the IdP and the SP is built on a documented and formalized trust model forming the federation. It becomes extremely important that the appropriate security technology support the trust established within the federation, so that only members of the identified federation have access to the data and information provided by the SP application.

The SP requires a technology architecture and business process that supports the SP in trusting the IdP's timely management and accuracy of the identity information provided. Without a proper trust model, the SP cannot rely on the information from the IdP and would require internal security checks to prohibit imposters from gaining access. Additional security checks from the SP decrease the overall efficiency gained by federating. Therefore, in the trust model, it is extremely important that the IdP establish effective identity information life cycle processes.

- The IdP needs a technology architecture and business process that supports the IdP in trusting the SP's use of the IdP information provided, as well as management and security of the data and information used within the SP's application. Because the identity information provided by the IdP may be sensitive, the SP must put safeguards in place to maintain the security of the information, using the information only as required and not maintaining or storing identity information.

Federation Protocols

When forming a federation, a key agreement is the *federation protocol* that will be used. This is the "common language" that will be used between the various applications in the federation to establish the standard architecture for supporting authentication and features available to the federation. Such protocols typically use XML and the XML Signature [27] standard as well as public key infrastructure (PKI). With PKI technology, each party uses a public key or certificate to prove ownership of a security key. When an unauthenticated user attempts access to an SP application, the federation protocol routes the user to the IdP for authentication. After authentication, the federation protocol builds the *assertions* containing the identity information and sends the assertions back to the SP, permitting the SP to make application access decisions regarding the user.

² See work of the Kantara Initiative (<http://kantarainitiative.org/>) and Project Concordia (<http://projectconcordia.org>)

Several protocols can be used to establish a trusted federation between parties. The Security Assertion Markup Language 2.0 (SAML 2.0) protocol and use cases are the most mature and represent the most widely adopted protocol for a provider-based federation. SAML 2.0 integrates key features of several earlier standards, e.g., Liberty Alliance and Shibboleth. The following sections are focused on the SAML 2.0 use cases. Starting in Section 3.13 are comparisons to traditional SSO and the alternate federation approaches.

3.2 SINGLE SIGN-ON IN AN IDENTITY FEDERATION

A primary goal of most federations is to establish single sign-on (SSO) ability to all systems in the federation. In a federation, first-time access to an SP application by a user is different than in a non-federation application. To a user in the federation, the login process may not appear any different; it appears to be done directly by the SP — the only observable difference may be the 'look and feel' of the authentication page presented by the IdP. However, it is the IdP that will actually have handled the authentication. **Figure 3.2-1** illustrates the process used for authentication in a federation.

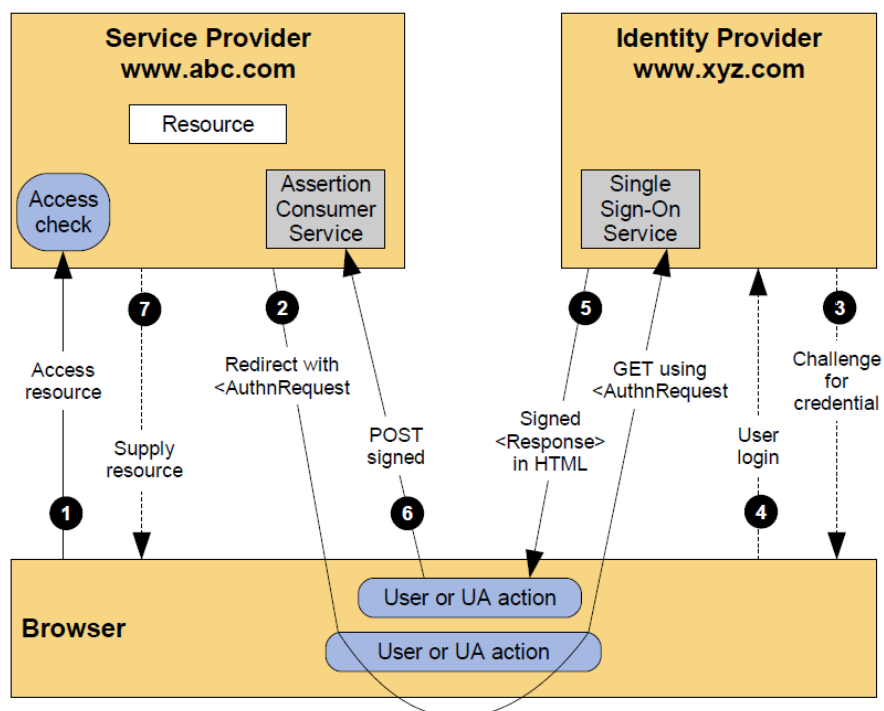


Figure 3.2-1. Service Provider-Initiated Access to an Application in a Federation

[Nick Ragouzis et al. Security Assertion Markup Language (SAML) V2.0 Technical Overview. OASIS SSTC, March 2008.]

1. When the user attempts access to an SP for the first time, there is no current session or security context established between the SP and the user. In order to establish a session, the SP needs to authenticate and confirm the identity of the user. The SP does not authenticate the user.
2. Instead, the user is re-directed (HTTP redirect or HTTP POST) to the IdP for authentication.
- 3./4. The IdP checks the credentials (i.e., password, token, biometric) of the user, and if they do not have a valid local security context, authenticates the user. When authentication is completed, the IdP collects information on the user's identity as *assertions*.
- 5./6. The user, including the user's assertions, is then redirected back to the SP using a HTTP POST.
7. The SP uses these assertions to establish a session for the user.

It is also possible to have identity provider-initiated SSO, as shown in **Figure 3.2-2**. In the IdP-initiated SSO, the IdP pushes the assertions to an SP. Using the IdP-initiated SSO is common in cases where a user accesses a common environment, such as a portal in their company intranet, to find and access external applications.

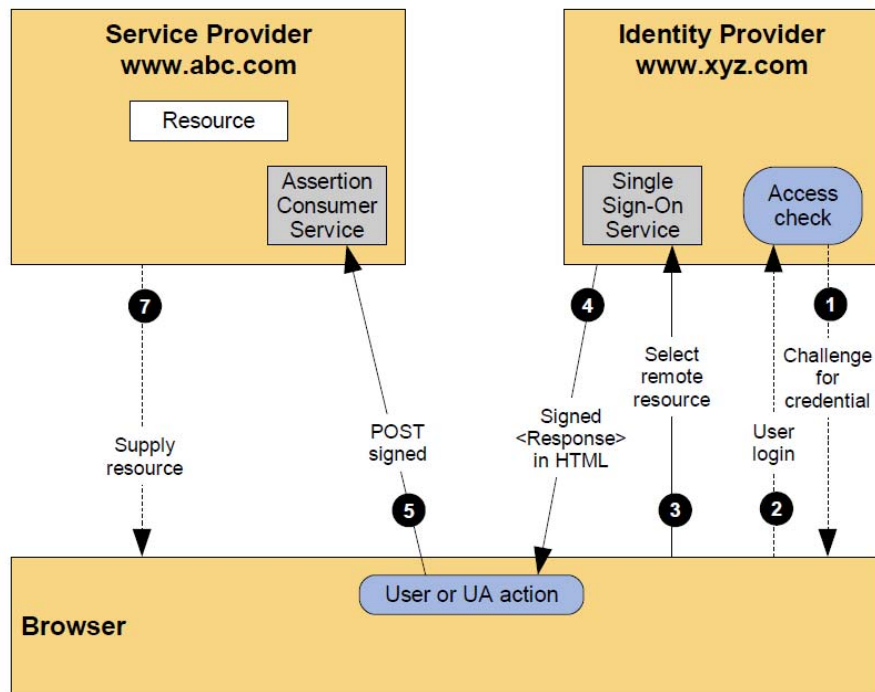


Figure 3.2-2. Identity Provider-Initiated Access to an Application in a Federation

[Nick Ragouzis et al. Security Assertion Markup Language (SAML) V2.0 Technical Overview. OASIS SSTC, March 2008.]

3.3 IDENTITY PROVIDER DISCOVERY

When the user requests application access, the SP directs the user to the preferred IdP for authentication. In a small federation with only one IdP, redirecting the user to the IdP is not difficult. But if a federation consists of multiple service providers and multiple identity providers, it might not be obvious which identity provider to use. The selection of the identity provider is handled by a discovery process referred to as a "where are you from" (WAYF) service. The discovery process can be either automatic (based on a local user profile, user's browser IP address, URL link used, or persistent user cookie), or it can be manual (user must select preferred identity provider from a list).

Besides implementing whatever the federation standard specifies, a discovery service will have additional components.

Federation discovery: Federation standards include profiles that establish processes for the identity provider to automatically discover the user's preferred identity provider.

Web WAYF: As the selection of IdP typically includes some kind of user interaction, a Web interface for selecting the IdP will be a part the discovery service. This interface will provide the user with a choice of IdP the first time the discovery service is used and provide the ability to store the user's selection so that they need only make it once.

IdP registry: Contains all IdPs in the circle of trust. One use of the registry is to build the selection list of possible IdPs mentioned above.

IdP selection rules: If there is some kind of automated selection logic, this is handled by the IdP selection rules. This component can work together with the federation discovery component but can also be based on specific rules for selection of the IdP. For example, rules of this type might be based on the source IP address or the referrer in a HTTP redirect.

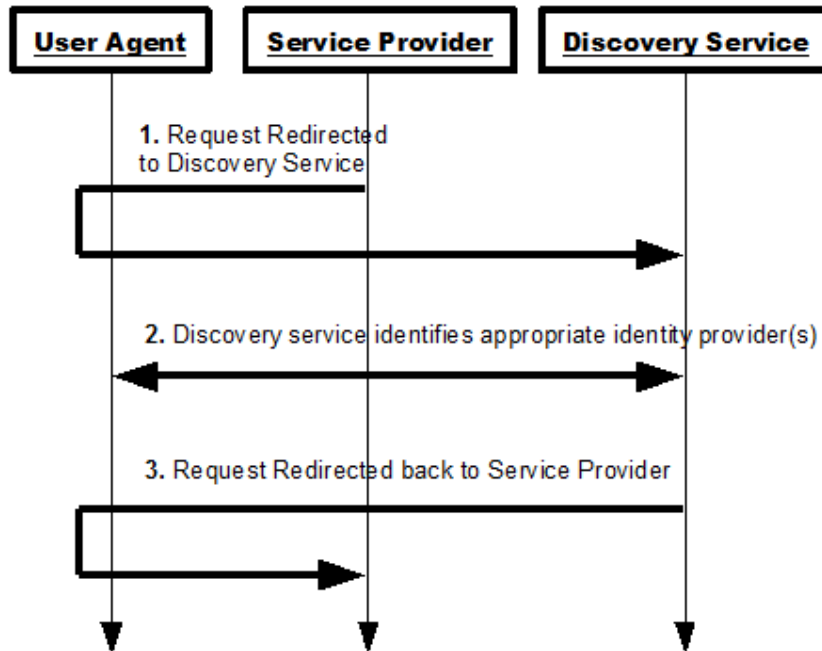


Figure 3.3-1. Identity Provider Discovery

[Sstc-saml-idp-discovery-cs- 01; 27 March 2008]

3.4 ASSERTION QUERY

As described, in a federated login process, the SP will be presented with security assertions. The assertions provided by the IdP may not contain all the information the SP requires to establish a session with the user. If this is the case, the SP has the ability to ask the IdP for additional assertions about the user, via an *assertion query (or attribute query)* service. **Figure 3.4-1** illustrates an assertion query. The SP may issue an assertion query at any time during a session, not just when the session is initially established. One common example is the case when an SP application is used by multiple IdPs: The SP may query for additional identity attributes from a different IdP than the one that originally authenticated the user.

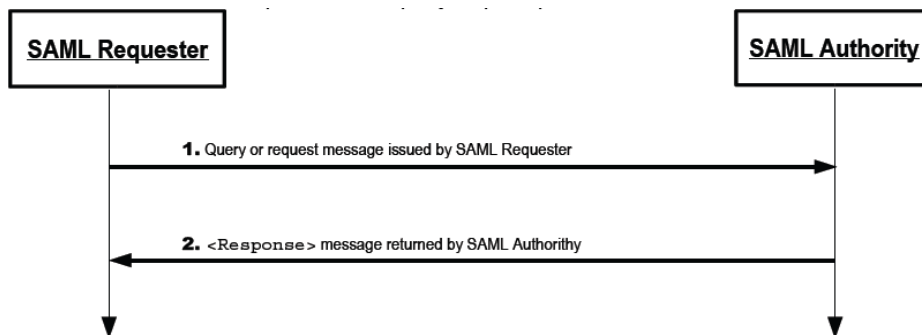


Figure 3.4-1. Assertion Query Service

[saml-profiles-2.0-os 15 March 2005]

Assertion queries are also used in cases when a standard SSO flow is not used. In the case without an SSO flow, the authentication between the client browser and the SP uses SSL and client certificates, authenticating the user directly. In direct authentication, the information from the client certificate is only available to the SP. Using information from the client certificate, an SP can submit an assertion query to an IdP in order to receive additional information about the user.

3.5 ATTRIBUTE DEFINITIONS

Assertions are the core information provided to the SP from the IdP, either directly at application login or as a result of an assertion query. Assertions contain user attribute information. The core attribute of an assertion will be the identifier of the user. This is typically called the subject, username or user ID. Additional attributes can be simple values like name, e-mail address and title. Additional attributes (*structural attributes*) may also be communicated such as organizational information, entitlement or application rights.

Because assertions contain user attribute information, the attribute definitions must be precisely defined in order for the SP to be able to use the assertions. It is essential to define the attributes that must and may be present in an assertion when establishing the federation. Since Personally Identifiable Information (PII) may be exchanged between parties, the trust agreement should document the PII being exchanged, whether the data needs to be encrypted within the assertions, and the requirements on the trusting party to securely maintain, store and expose this information.

3.6 PRIVACY AND PSEUDONYMS

In a federated environment, with identity information and other assertions passing through a network between systems, protecting the user's privacy becomes paramount. With SSO, it is possible to track the user across several SPs. *Pseudonyms* provide a way to obfuscate the identity of the user across SPs. When the IdP delivers the assertions to the SP, the use of pseudonyms makes it possible to have a different user ID for the same user at each SP. **Figure 3.6-1** illustrates pseudonyms.

A pseudonym can be either persistent or transient. If a persistent pseudonym is used, the SP will see the same pseudonym each time the user accesses the SP. When transient pseudonyms are used, the SP is presented with a different pseudonym each time a user gains access to the SP. Pseudonyms are typically used together with assertion queries when delivering the information required to the SPs. Established policies allow the SP to gain access to general information on the user, such as roles and entitlements, while also protecting the user's privacy. To minimize assertion queries when pseudonyms are used, the IdP assertion query service can be configured using the policies so as to deliver the pseudonym and a standard set of assertions to the SP during SSO.

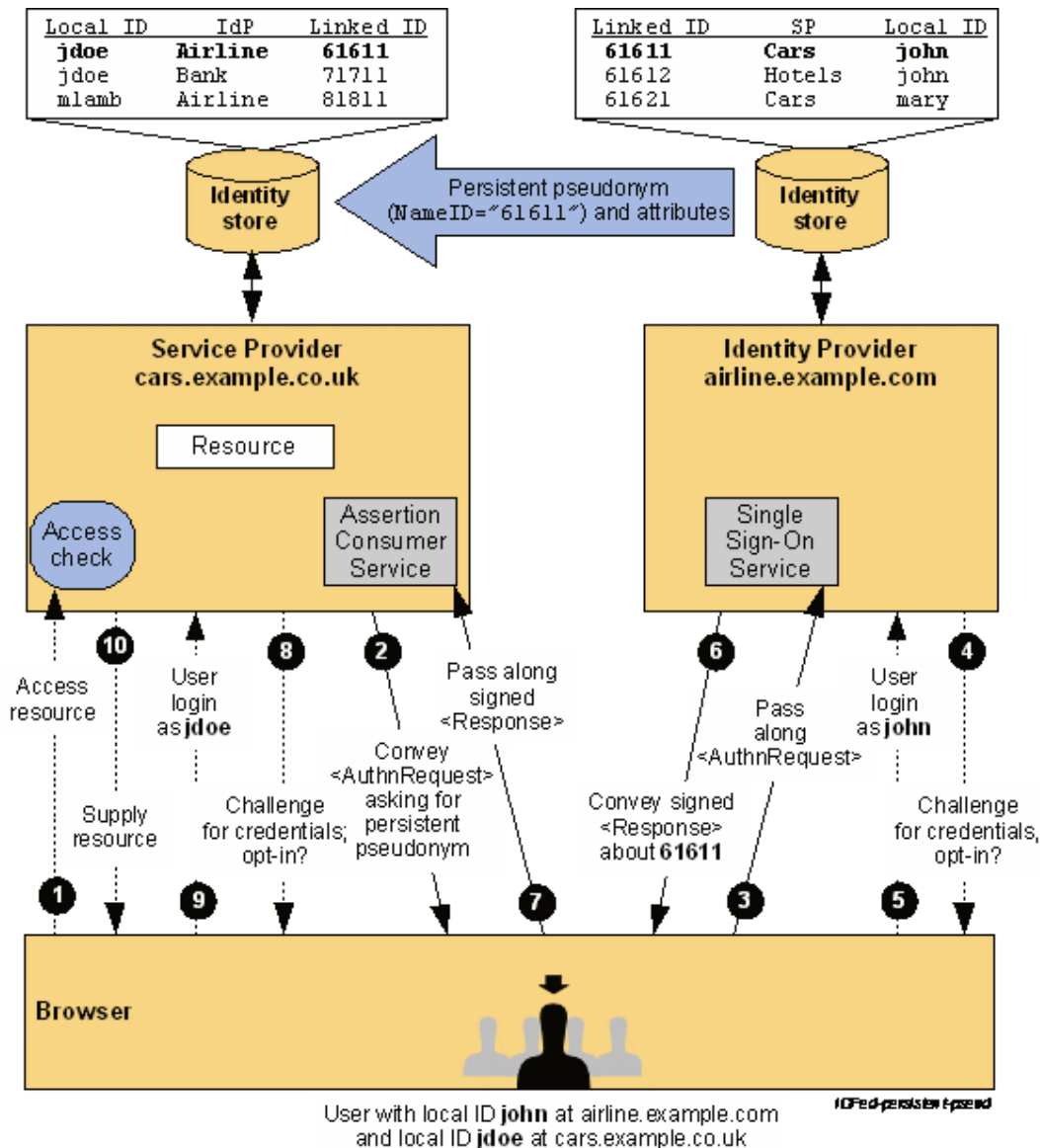


Figure 3.6-1. Protecting Privacy Using Persistent Pseudonyms in Assertions

[Nick Ragouzis et al. Security Assertion Markup Language (SAML) V2.0 Technical Overview. OASIS SSTC, March 2008.]

3.7 NAME MAPPING

When pseudonyms are used, the user's username is specific to the SP. For some SPs, it is convenient to have the pseudonym mapped to an identifier local to the SP. When a pseudonym is used for the first time, the SP will ask the user to identify with a local identifier. The SP initiates a name mapping service with the IdP to map the supplied pseudonym with the SP's local identifier. The next time the user accesses the SP, the assertion from the IdP will contain only the pseudonym.

A name mapping service can be used with both persistent and transient pseudonyms. However, it is most pragmatic to use name mapping with persistent pseudonyms — because transient pseudonyms are normally used for SPs when the exact identity of the user is not required.

A name mapping service can also be used to terminate mappings. When a user no longer has a local identifier with an application, the local identifier is removed from the local store. Termination of

mappings also occurs when the association between the pseudonym and the local identifier is no longer desired. A user's initiation of the removal of an SSO service with a particular SP is an example of this. Termination of a name mapping is also called *federation termination*, indicating that the federation between the pseudonym and the local identifier has ended.

3.8 NAME LINKING

An SP can use assertions to communicate identity information with other SPs. Because pseudonyms are unique to an SP application, direct assertions between SPs do not work. Name linking services are used to overcome the obstacles presented with pseudonyms. To preserve the user's privacy, encryption is recommended. When encryption is used, the requesting SP will not be able to see the actual pseudonym used by the other service provider. Name linking is illustrated in **Figure 3.8-1**.

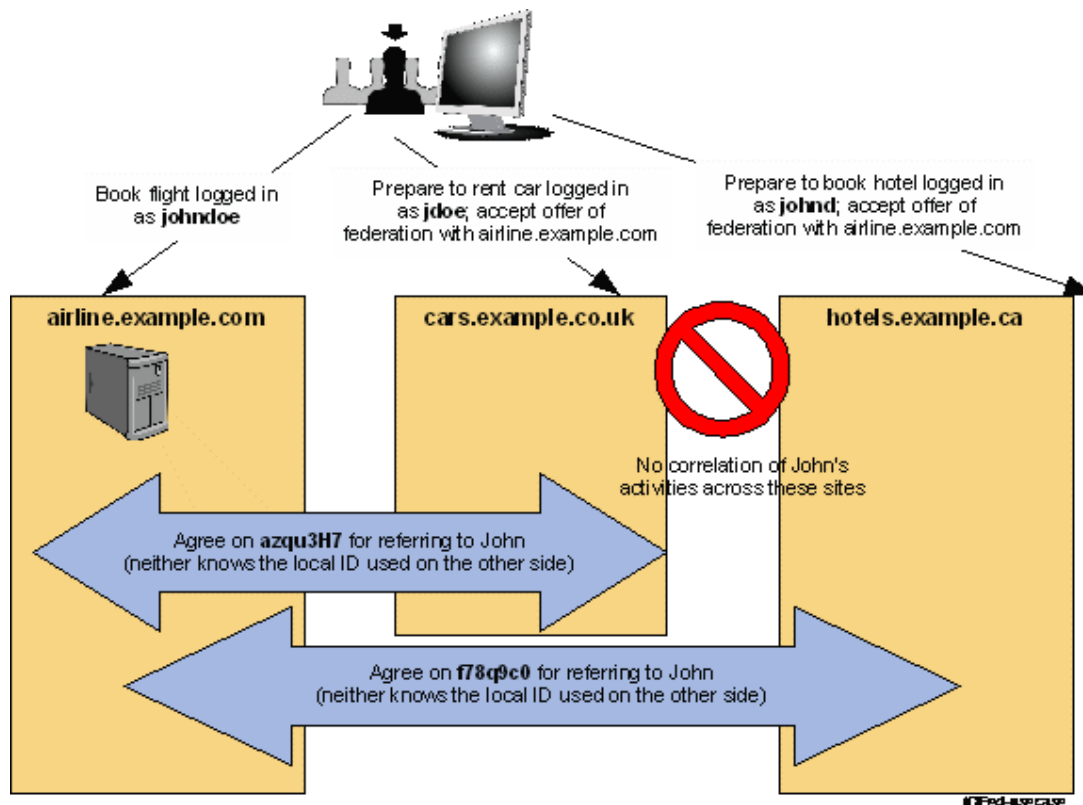


Figure 3.8-1. Name Linking

[Nick Ragouzis et al. Security Assertion Markup Language (SAML) V2.0 Technical Overview. OASIS SSTC, March 2008.]

3.9 FEDERATION TERMINATION

When a user no longer requires access to a federated resource, the standards have defined the process for an entity or individual to request that any established account linking be terminated, a process described in **Figure 3.9-1**. In many federation cases, an entity or individual needs to establish a persistent local identity with the service provider, e.g., to enable personalized content when they return to the SP application or for internal auditing by the SP application. In these cases, when the user terminates the federation, it is best to remove the association for the user and any PII entered in the federated application.

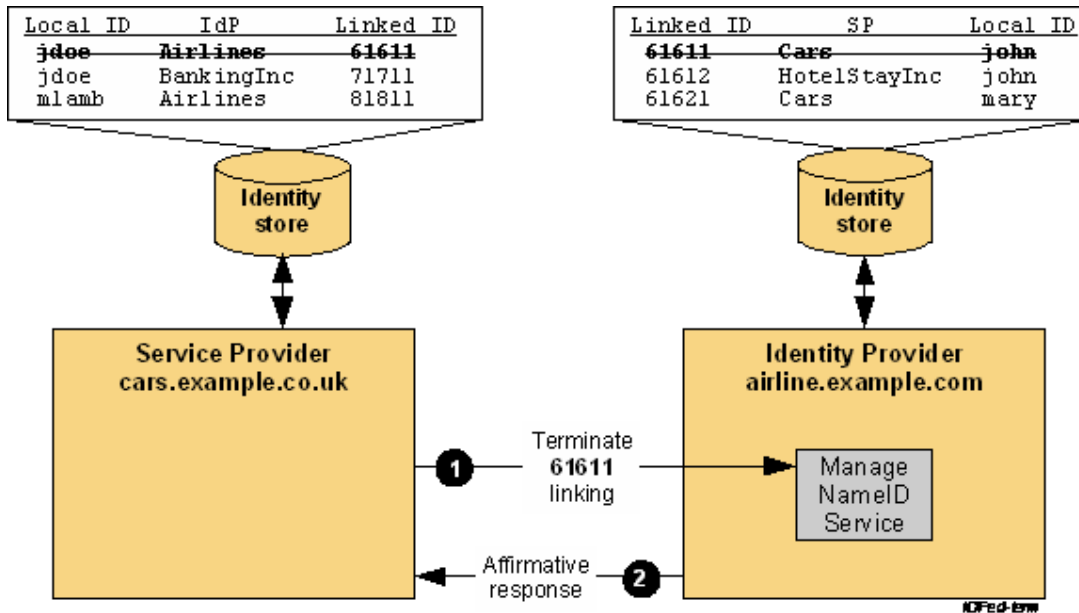


Figure 3.9-1. Federation Termination

[Nick Ragouzis et al. Security Assertion Markup Language (SAML) V2.0 Technical Overview. OASIS SSTC, March 2008.]

3.10 SINGLE LOGOFF

Although SSO is an obvious feature to provide in identity federation, single logoff (SLO) might not be quite as obvious. With SSO, a user authenticates only once. Typically, the user will not distinguish between the SPs and the IdP. After the first sign-on, the user will move between SPs seamlessly, without noting different SPs in his path. In a scenario where the user is moving seamlessly between applications, a business case can be made to reduce the requirement for subsequent logoffs by the user in each SP location. The concept of SLO is a pragmatic and secure solution for ensuring that logoff will close all active sessions created from the SSO.

When the user selects logoff in an application, two potential options must be offered. Does the user want to logoff from this specific application, maintaining the current SSO session — or does the user want to end their SSO session, closing all individual application sessions? In the latter case, the local SP communicates the logoff request to the IdP. The IdP, based on its session store and information from the metadata, issues a logoff request to all SPs for which an active session is present. When the SP receives a logout request, it will close the current session and notify the application, allowing the application to perform required cleanup. **Figure 3.10-1** illustrates user logout from App 2 and the resulting logout from App 1.

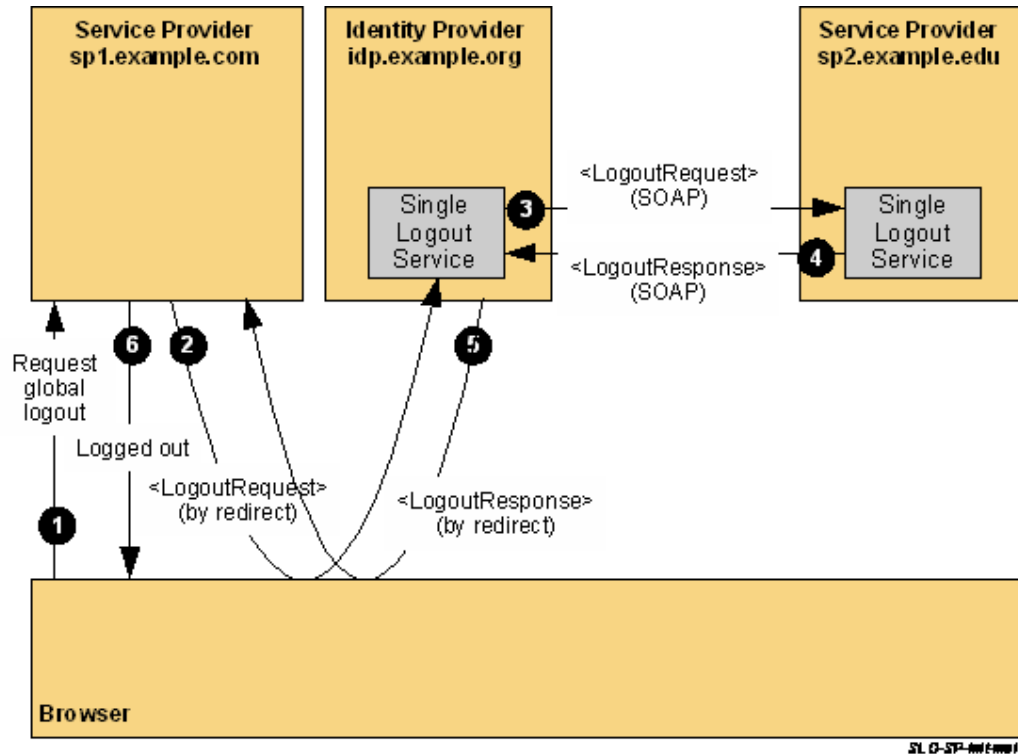


Figure 3.10-1. Single Logoff in a Federation

[Nick Ragouzis et al. Security Assertion Markup Language (SAML) V2.0 Technical Overview. OASIS SSTC, March 2008.]

Session timeout is another issue handled by SLO. With SSO, the user is using the same login for several applications; it is inefficient to manage SSO session timeouts by each application. With SLO, applications can through the IdP consolidate and manage single oversight for a user's idle time. Consolidating session timeouts and establishing a consistent session timeout period is another policy that must be considered when a federation forms.

3.11 COMMUNICATION CHANNELS AND ARTIFACT RESOLUTION

Communication between the IdP and the SP is vital for a federation to exist successfully. In each of the scenarios described previously, communication is occurring via the user browser for each redirect and response between the parties. For SSO, it makes sense for the IdP to manage the browser when a user is logging in and communicate this information back to the SP. Other features, such as the assertion query, require direct communication between the SP and the IdP. However, situations exist where a direct channel of communication is not possible between SPs and IdPs. If so, *redirect* is used to route assertions and assertion queries through the browser.

An alternative approach to using the user's browser for communications between the IdP and SP is to combine the front-channel communications through the browser with back-channel communications directly between the SP and the IdP. By using back-channel communications, both the size of the redirect message and any security issues (e.g., man in the middle attacks) from feeding the SSO result directly through the browser, can be minimized. **Figure 3.11-1** illustrates a process for combining the front channel and the back channel, passing the SSO result *by reference* through the front channel by the IdP to the SP increases the security of SSO. The SP receives the reference and translates it through the back channel to obtain the actual SSO result.

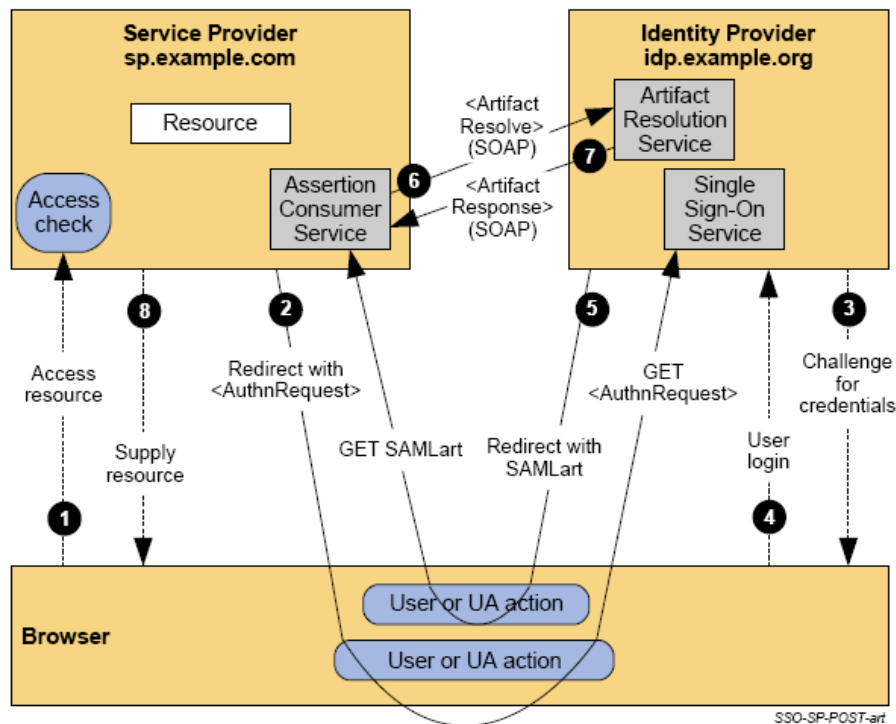


Figure 3.11-1. SSO Utilizing Front-Channel and Back-Channel Communication

[Nick Ragouzis et al. Security Assertion Markup Language (SAML) V2.0 Technical Overview. OASIS SSTC, March 2008.]

The various steps in **Figure 3.11-1** are as follows:

1. The user requests access to a resource on the SP.
2. The request is intercepted by the SP and redirected to the IdP because it is a first time access, and no security context has been established.
3. The IdP checks the user's security context. If necessary, it requests authentication from the user.
4. The user authenticates (typically username/password, but some federations require additional authentication processes such as SecurID tokens, biometrics, etc.)
5. The request is sent to the SP, who checks the credentials.
6. If the credentials are deemed okay by the SP, an artifact request is sent from the SP back to the IdP application.
7. The IdP sends the referenced artifact to the SP.
8. The user is now authenticated, and the SP application passes the response to the browser, allowing access.

Reviewing the steps for authenticating in a federation using SSO (as discussed in Section 3.2 of this white paper), it is seen that for both front and back channel communication, the process is the same up until step 5. With back-channel communications, instead of directly sending the identity information to the SP, the result delivered from the IdP is just a reference, or artifact, to the actual identity assertion. The SP uses the reference to retrieve the actual identity information from the IdP in steps 6 and 7. The identity information is checked and the user achieves access in step 8.

Another scenario is when the user agent is not a *passive* client, as a browser, but an *active* client, as is the case with a mobile device. In the active client context, the client is able to make calls directly to the entities in the federation. Most often, such clients will communicate with the IdP via a Simple Object Access Protocol (SOAP) message to supply information to the SP via a PAOS (reverse SOAP) message. If there is no direct communication channel between the SP and the IdP, an active client may be required to support some kind of relaying communication between the SP and the IdP. Defining of the communication channels and combinations of these channels must be completed when a federation protocol is established.

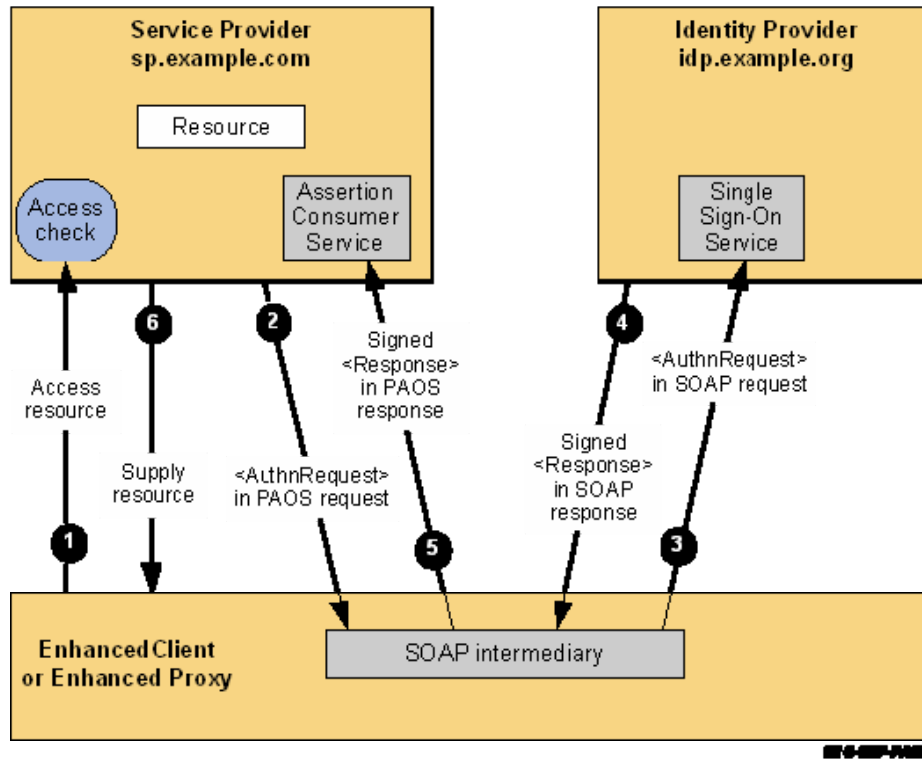


Figure 3.11-2. Federation Using an Enhanced Client or Proxy (ECP) Device

[Nick Ragouzis et al. Security Assertion Markup Language (SAML) V2.0 Technical Overview. OASIS SSTC, March 2008.]

3.12 ADDING WEB SERVICES AND SOA

Until this point, most of the scenarios and architecture discussed used a client browser **to mediate information exchange among the user, the SP, and the IdP**. In cases where an application is communicating with another application, the exchange of information can be achieved via Web services. This request can be either directly to another application or indirectly to one or more applications through an SOA infrastructure. Web services that use the user's identity information are also called *identity services*. A Web service consumer (WSC) is an application that utilizes Web services to invoke actions. To validate a request for action, authentication is performed by the Web service provider (WSP) using either the application user's identity information or application to application authentication. Identity federations propagate identity information using Web services.

When an application uses Web services for actions, security tokens are used to authenticate the application. Security tokens are issued by a security token service (STS). The token is validated by the WSP and validates the consumer's actions. The relationships among an STS, WSC, and WSP are illustrated in **Figure 3.12-1**.

- Web service consumer (WSC)
- Web service provider (WSP)
- Security token service (STS)

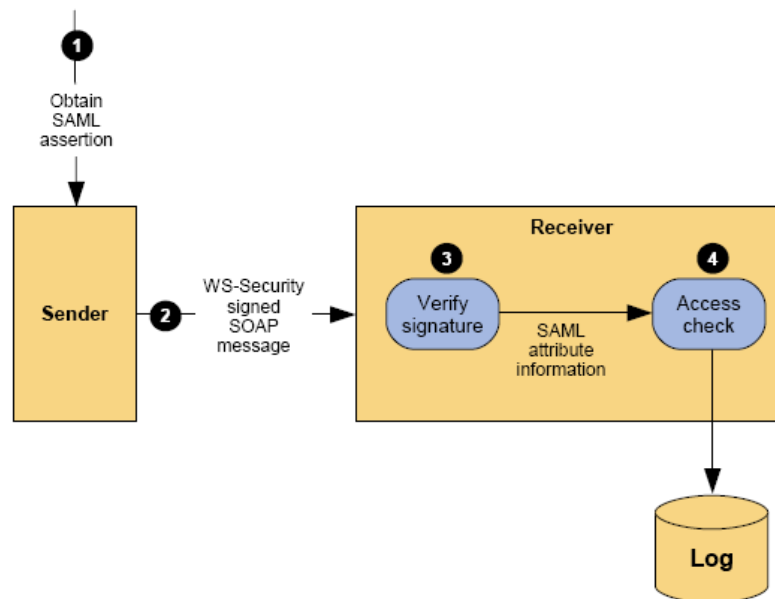


Figure 3.12-1. Relationships Among an STS, WSC, and WSP

[Nick Ragouzis et al. Security Assertion Markup Language (SAML) V2.0 Technical Overview. OASIS SSTC, March 2008.]

The token service plays the same role for Web services as the IdP plays for browser-based login. Note that the consumer communicates directly with the STS — redirection is not possible with Web services. Metadata is available for the Web services and is checked by the consumer to determine the required type of security token for a given service.

“Chaining” (**Figure 3.12-2**) describes the relationship when a WSP is a WSC of services with another WSP.

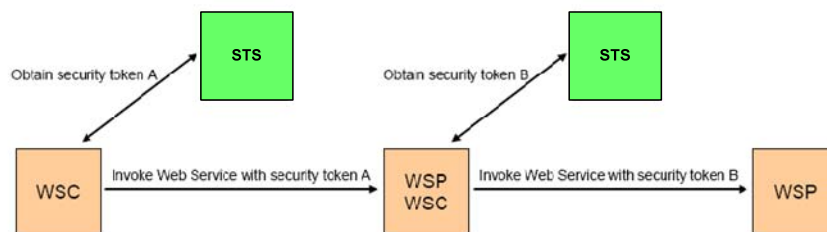


Figure 3.12-2. Chaining

Generally, an STS is not a separate service. An application can act as its own STS by building a security token using the application or user’s identity information when issuing a Web service request. In an identity federation, having the IdP play the role of STS is particularly useful when the security tokens used are related directly to the actual user at the front of the chain.

The separation between applications and STS(s) depends on the trust model used and how the security token is constructed. Let’s assume that the security token contains only a username signed by the issuer. When an application is acting as its own token service, it is possible for the application to fake the identity of the user upon whose behalf it is acting. The content of the security token can be strengthened using a signature issued by the IdP, along with the username. The username cannot be forged by the application when an IdP signature is issued. IdP signatures can be obtained when the IdP is acting as an STS.

Separating the application and the STS can also improve security. Application signatures can also be applied when working in a rich client environment. Rich clients normally have the ability to invoke Web service methods that can both obtain and use a security token.

3.13 FEDERATION SSO VS. TRADITIONAL SSO

Before moving on with federation, let's look more closely at a typical scenario with an SSO solution deployed. **Figure 3.13-1** illustrates a typical setup: two SSO systems, each supporting a number of applications. To interact with an application, a user is required to "pass-through" the SSO system that manages access to the application.

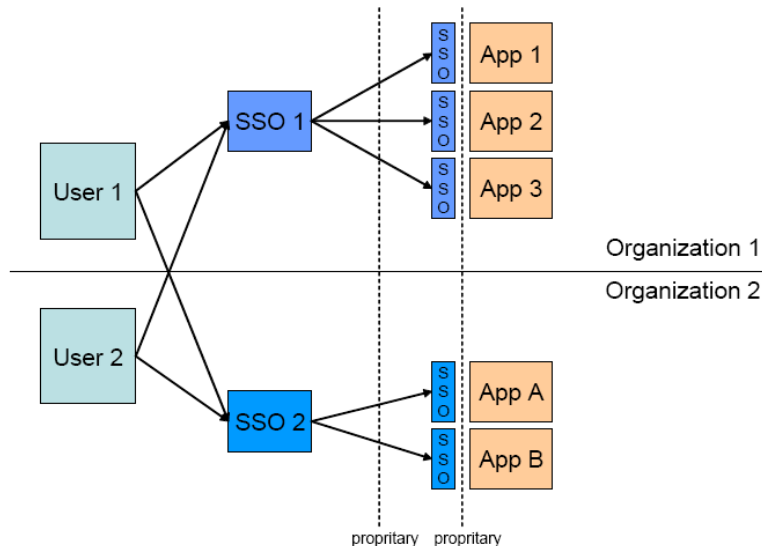


Figure 3.13-1. Traditional SSO: Using Multiple SSOs to Manage Application Access

In a multiple SSO environment, the user's identity will be with each SSO system in order to access the applications. In **Figure 3.13-1**, **SSO1** is the SSO managing access for **App 1**, **App 2** and **App 3**. Likewise **SSO2** manages access for **App A** and **App B**. In a multiple SSO environment, a user registration is required in the directory for *each* SSO.

The interfaces for communication between the SSO and the application are unique to the SSO supplier. Because each SSO provides their own proprietary solution for interfacing with applications, crossing SSO environments is not possible unless changes are made to the applications.

3.14 FEDERATION SCENARIO

Let's investigate a similar situation that instead employs an identity federation model. A federation is formed among the owners of the five applications, and they have adopted a federation standard. **Figure 3.14-1** illustrates a typical federated scenario. Note in the illustration that a user only needs to establish identity with one authority (IdP).

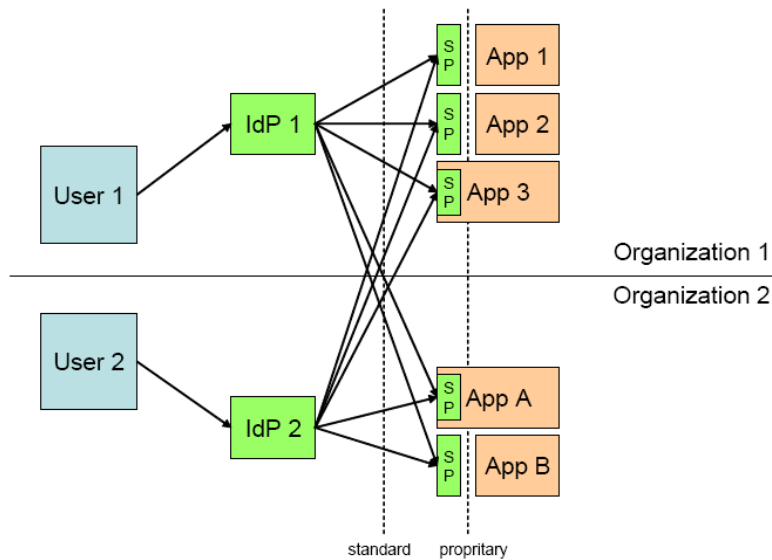


Figure 3.14-1. Typical Federation SSO

When the user has established identity with the federation through the user's selected IdP, the user will have access to all applications in the federation. In contrast to the SSO solution presented in Section 3.12, a user is only required to be registered once in their IdP to access the applications.

In the scenario outlined in **Figure 3.14-1**, each application is serving in the role of an SP in the federation. SPs require an interface into the application to support the communication with the IdP's. The federation protocol is defined by the federation agreement. Therefore, any SP interface which supports the selected protocol can be used. Again, in contrast to the multiple SSO scenario, the SP interface is not tied to the IdP. **Figure 3.14-1** also suggests that App 3 and App A applications have built-in support for playing the role of an SP. For applications with built-in support for SP roles, making the application participate in a federation might be a matter of configuring the application.

Comparing the multiple SSO scenarios with the federation scenario illustrates the difference between traditional SSO and federation. With traditional SSO, you are bound to SSO within the scope of the selected SSO solution. Applications need to be able to support that solution. But in the case of an identity federation, each application needs to support the federation standard selected, making it possible to communicate with different IdPs.

3.15 ADOPTING A FEDERATION STANDARD

Several major federation standards — SAML 2.0, Liberty Alliance (ID-FF), WS-Federation, Shibboleth, CardSpace, and Open ID — have matured and stabilized. The standards offer similar features and provide interoperability between security domains as well as inherent security. SAML is a self-contained standard without much dependency on other standards. In contrast, the WS-Federation specification is built on the foundation of the Web service (WS-*) security standards and specifications, which have a more compostable nature. As the standards have evolved, a natural consolidation and convergence have occurred

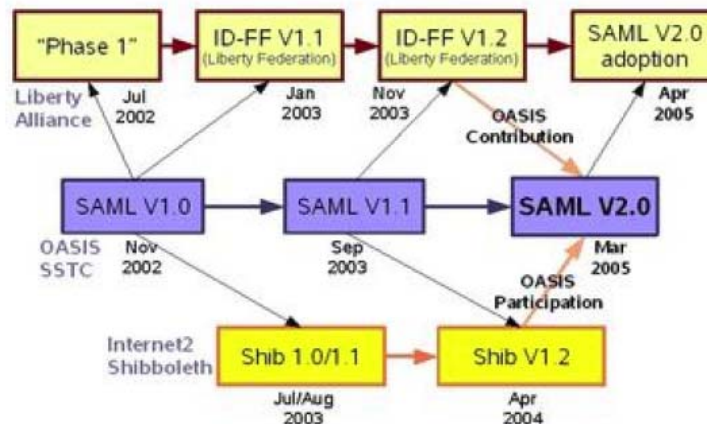


Figure 3.15-1. Federation Standard Evolution

[Source: Liberty Alliance]

When establishing a federation, the decision about which federation standard to use is critical. The primary factors to be considered are:

- Functional requirements of the federation
- Use cases supported by the standards
- Ease of implementation and support for the applications that will be federated

While the standards offer similar features, they are not compatible. SAML and WS-Federation are comprehensive standards, but they are only the foundation of a fully identity-enabled infrastructure. Therefore, a federation must select one or the other, while also actively participating and maintaining a focus on the continued evolution of federation studies.

- **SAML** is the foundation for much of the current identity federation activity. It has gone through three releases —1.0, 1.1, and (the most recently ratified) 2.0. SAML 2.0 is seen as a point of convergence, as it incorporates ID-FF 1.1 and 1.2 as well as Shibboleth 1.3 functionality. The **Shibboleth** initiative is being led by Internet 2 to provide peer-to-peer collaboration using a federated identity infrastructure based on SAML. Shibboleth has been largely adopted by the university and research communities around the world. Shibboleth 2.0, which was released in March 2008, is based on SAML 2.0.
- The **Liberty Alliance** is an organization of vendors and enterprises that is largely perceived as having formed in response to Microsoft's Passport efforts. Since that beginning, the Liberty Alliance has written several protocols that enable both browser-based identity federation as well as Web services identity federation. The Liberty Alliance protocols include the identity federation framework (ID-FF) and identity Web services framework (ID-WSF). Their ID-FF work, which originally resulted in two versions of the ID-FF specification, has now been incorporated into SAML 2.0. Liberty Alliance has also taken on the role of certifying conformance and interoperability of vendor products to federation standards. They provide testing services for SAML 2.0 as well as their own protocols.
- The **WS-***Federation suite of specifications is driven by a collaborative effort among Microsoft, IBM, VeriSign, RSA Security, Ping Identity and others. Some of these protocols, such as WS-Security, have been submitted to and ratified by existing standards organizations, such as Organization for the Advancement of Structured Information Standards (OASIS). Generally speaking, WS-* can be thought of as a composable suite of specifications for enabling secure Web services. Specifically, this collection of specifications — including WS-Trust, WS-Federation, and WS-Policy — is an evolving set of mechanisms for layering authentication, authorization, and policy across both single and multiple security domains.
- Recently, it was determined that a more user-defined and user-controlled federation SSO protocol was required to facilitate federation among a wide range of user Internet sites. These alternate approaches are being rapidly adopted for access control to public blogs and wikis.

SAML Specification vs. WS-Federation Specification

However, when it comes down to the details, the two specifications are quite different.

SAML 2.0 and WS-Federation: Many Commonalities	
<ul style="list-style-type: none"> • Same base principles — an identity provider and a service provider • Both support all the features discussed • Both address the issue of passive and active clients • Both can tunnel their communication through the browser when required • Both have SOAP as a communication channel for active clients 	

SAML 2.0 vs. WS-Federation: Key Differences		
Attribute query and assertion	SAML precisely defines messages for exchanging queries and assertions.	WS-Federation does not define any specific messages but suggests using a Web service for this purpose..
How services are defined	The SAML standard defines a transport-neutral protocol that can be used with various bindings, SOAP being one of them.	The WS-Federation specification defines its services using SOAP and then builds facilities for tunneling SOAP through the browser.
Assertions and security tokens	The SAML standard defines identity information in the form of assertions. Large parts of the standard go into defining the assertions and attribute profiles.	No definition of what the security token looks like; the token is opaque, with one possible security token being SAML assertions as defined in the SAML standard. Thereby WS-Federation can leverage part of SAML, which is well in line with the composability concept normally used in WS-* specifications.
Handling of single logout	Logout information will always be sent to as many receivers as possible. No notion of session timeout.	An application has to register to receive logout information. No notion of session timeout.

Table 3.15-2. SAML Specification vs. WS-Federation Specification

- OpenID is a newer “open, decentralized, free framework for user-centric digital identity”³. OpenID is designed for users who want a single login for several applications on the Internet. The framework is driven by the needs of Web 2.0 applications such as blogs and wikis. OpenID has a much more lightweight nature and is not based on several layers of XML schemas, WS-* standards, and a variety of data formats and communication channels.

Whereas these latter specifications amount to several hundred pages, the OpenID specification is only 14 pages long. One could say that the other specifications satisfy an organization’s wish to provide advanced functionality and fine-grained control.

Instead of using SAML to create identity assertions, OpenID uses eXtensible Resource Descriptor Sequence. This metadata format consists of eXtensible Resource Identifiers (XRIs) to identify users . After authenticating with an OpenID Provider (OP), the XRIs are validated by the OpenID Relying Partners(RP) before permitting access . Typically, the RP will host an authentication service that refers the user back to their selected OPs when first accessing the Web site. A typical OpenID sign-on exchange is shown below. In many respects the sequence of events does not appear very different from a SAML or WS-Federation use case.

³ <http://openid.net/>

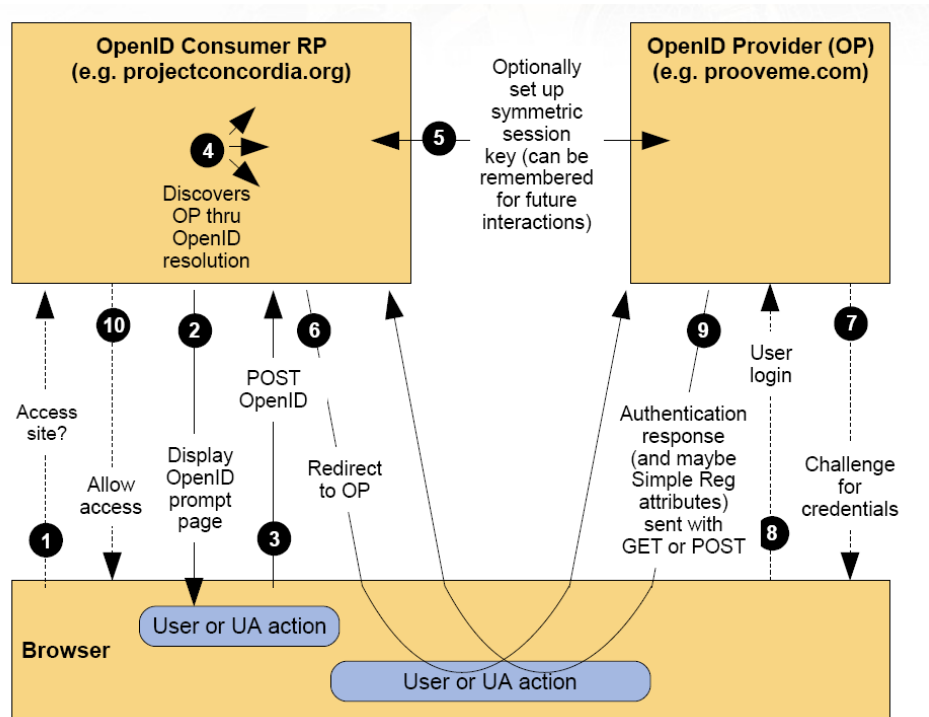


Figure 3.15-3. OpenID Sign-On Process

[SAML 2.0 Basics updated 2 October 2006 (eve.maler@sun.com)]

The identity provider is responsible for performing authentication of the user and thereafter supplying identity information to the relying party in a secure fashion. When using OpenID the user is able to select his favorite identity provider supporting OpenID. When registering with an identity provider the user is provided with a URL (or XRI) which is the identifier the user will present to an OpenID compliant relying party.

In OpenID, two different communication channels are used: *direct communication* and *indirect communication*. Direct communication is used when the relying party and the identity provider communicate directly with each other using HTTP; in indirect communication, the relying party and the identity provider communicate through the browser using HTTP POST (redirects are also supported but have been deprecated in the latest version of the specification). Data communicated using either form of communication is not encoded using XML but instead, uses simple key/value pairs. These are encoded using either Unicode text for direct communication or the HTTP encoding used when posting HTML forms.

There is no need for shared metadata. The URL that the user gets from the selected identity provider is used for the relying party to locate the OpenID provider. The relying party can establish a connection (called an association, in OpenID terminology) and security context with the identity provider using a Diffie-Hellman key exchange. Although OpenID is a lightweight specification, it is currently gaining support along with a number of other specifications currently proposed. These additions are related to assertion service (called attribute exchange, in OpenID) and a protocol for communicating additional arbitrary payloads between parties.

- CardSpace is a Microsoft.NET framework that “enables users to provide their digital identity to online services in a simple, secure, and trusted way.”⁴ Users create software “identity cards” that can store identity “claims” either locally or by selected IdPs. The user presents one of their “InfoCards” that satisfy the IdP and RP policy requirements to achieve application access. The CardSpace sign-on process is illustrated in Fig. 3.15-4.

⁴ <http://www.microsoft.com/net/WindowsCardSpace.aspx>

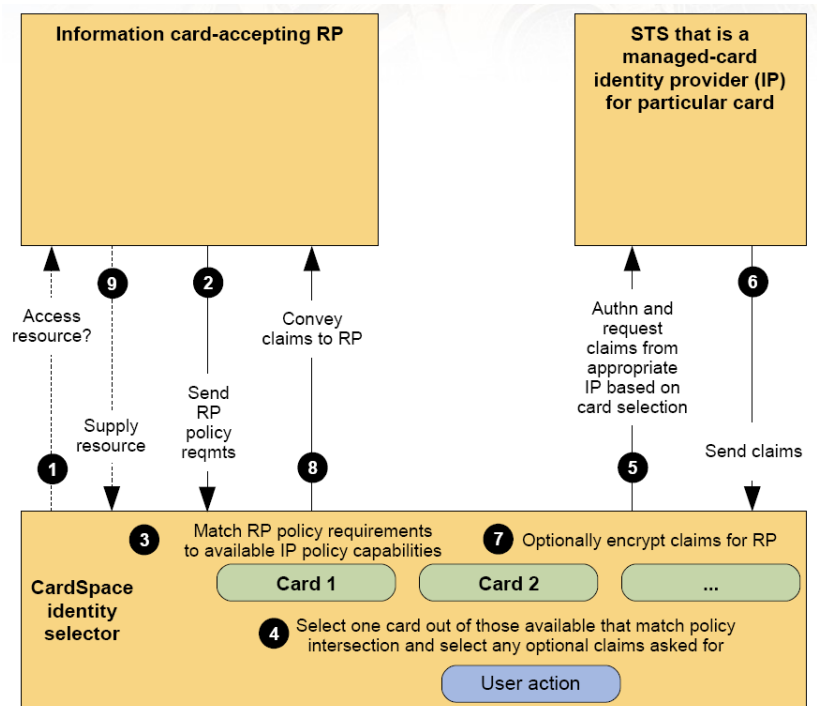


Figure 3.15-4. CardSpace Sign-On Process

[SAML 2.0 Basics updated 2 October 2006 (eve.maler@sun.com)]

The various features of these alternate protocols are compared to SAML 2.0 in **Table 3.15-5**.

	SAML 2.0	OpenID	CardSpace
Federated identity use cases	Covers a huge range; offers interoperability and flexibility but also complexity in messaging and data models for IdPs, RPs, and clients	Offers simplified sign-on and some attribute exchange; single unified ecosystem and emphasis on correlation (vs. access control) are notable characteristics	Offers client-side support for authentication/simplified sign-on and attribute exchange
User-centric use cases	Has <i>potential</i> to cover all three, depending on profiling and deployment choices	Covers “me generation” by design; is philosophically aligned with “do what I mean” (vs., “trust no one”)	Covers “trust no one” by design; is philosophically aligned with “do what I mean” (self-asserted cards are “me generation”-like)
Identity data security	Offers protection at channel, message, assertion, and application semantics levels	Offers optional channel protection, Diffie-Hellman message protection	After initial card provisioning, uses PKI-based strong authentication between client and IdP
Web authentication	Can convey strong authorization requests and assertions but also allows username/password (a weakness)	Agnostic as to method, but the username/password mechanism (a weakness) is common	Client-based selector provides a stronger alternative to weak Web authentication, once card is provisioned
Smart clients	Enhanced Client or Proxy (ECP) profile allows but does not require them	No native solution	All the pros and cons of a smart client
IdP discovery	The only solution offered (i.e., cookies) is weak	Very strong — built into identifier itself	Achieved through the cards themselves
Privacy concerns	Strong on privacy of IdP-held data, depending on deployment choices; uses ECP for the “trust no one” use case	Architectural and philosophical bias toward revelation (one-time OpenID feature could help a little)	Philosophical bias toward privacy, although interaction ceremony and hosted models may weaken this
Vouched-for information	Strong, with strong authorization of issuers	Weak; all information is self-asserted, so far	Allows for third-party managed information (as well as self-asserted information)
Governance	Anticipates nontechnical aspects (Liberty Alliance offers lots of guidance); metadata covers some technical aspects	Makes a simplifying assumption to trust/serve all IdPs, RPs, and users; no nontechnical governance needed; metadata extensible for future technical aspects	Allows for IdP/RP policy matching; no nontechnical guidance

Table 3.15-5. Comparison of Federation Protocols

[Based on SAML 2.0 Basics updated 2 October 2006 (eve.maler@sun.com)]

4 SUMMARY

Identity federation has matured with the expansion of established standards and approaches. The growing number of applications whose user bases extend across organizational boundaries is forcing enterprises to integrate identity federation into their IT vision, strategy, infrastructure, and application support models. Enhanced security, user management costs, and enhanced user experience are driving this transition.

When developing an approach, it will be important to cover the whole spectrum of identity federation capabilities, including how to handle identity-based Web services. Implementing federations is now possible, with the growing number of available commercial products and open source projects.

Identity federation is an important consideration in the application infrastructure. The boundaries between applications are growing less distinct as applications become more modular, use Web portals as their front-end architecture, and are accessed by external organizations and services. Enterprises are choosing to obtain business-critical applications from a variety of Internet sources. In the absence of federated identities, users are compelled to have multiple identities and organizations compelled to manage a growing number of user accounts.

Tightening security requires that most applications authenticate a user before granting access. And due to their modular nature, applications may even be able to communicate user details via Web services between applications, to enrich the user's experience. Identity federation is the key to integrating the broad landscape of application resources efficiently and securely.

The continued existence of multiple identity stores has become a security, identity assurance, and privacy risk. By contrast, federations are based on trust agreements that define the identity management practices of the federation partners, information that will be exchanged, and technology that will be used. Within a federation, an identity provider manages the user identity life cycle while the service provider manages the application and the authorization policies being used. The identity provider is responsible for authenticating the user before sending an assertion to the service provider. The service provider validates the assertion and then makes any required authorization decisions.

The two major provider-centric federation standards, SAML 2.0 and WS-Federation, both offer similar features. SAML is a self contained standard built upon HTTP, XML, and SOAP core standards without much dependence on other federation standards. WS-Federation, on the other hand, builds on the whole foundation of the WS-* standards and specifications and has a more composable nature. Even though SAML and WS-Federation are comprehensive standards, they are only the foundation of a far bigger goal — a fully identity-enabled infrastructure. Much work is still required to define which information to share and how to build and participate in federations. The federation standards support numerous communications protocols between federation partners and use cases.

User-centric federation, where the user has greater control of the use of their individual identity information and the details of federations that they participate in, has been gaining greater interest. While there are several operating instances of user-centric federation, mature policies, processes, and procedures — that can assure individuals that their identities are being appropriately protected — are still in development.