



Identity Federation: Concepts, Use Cases and Industry Standards

January 2007

Table of Contents

Introduction	3
Document Purpose and Scope	3
Federation Requirements	3
Basic Concepts	4
Federation Use Cases	5
Browser-Based Scenarios.....	5
Federation Based On Account Linking.....	5
Federation Based On Roles	5
Federation With No Local User Account	5
Document-Based Scenarios	6
Chained Web Services	6
Authentication Service	6
Federation Standards	7
Passport	7
Kerberos.....	7
Kerberos Components	7
Kerberos Process	8
Security Assertion Markup Language (SAML).....	8
Introduction.....	8
SAML Architecture	9
SAML Assertions	9
SAML Protocol	11
SAML Bindings and Profiles	11
Liberty Alliance	11
Introduction	11
Liberty and Industry Standards	11
Liberty Terminology	12
ID-FF	12
Single Sign-On and Federation	13
Communication Between Identity Provider and Service Provider	13
ID-FF 1.2 and SAML 2.0.....	16
ID-WSF	16
ID-WSF Use Case Scenario	16
ID-WSF Overview	16
ID-SIS.....	17
WS-Security	17
WS-Security and SAML.....	18
WS-Trust	19
WS-SecureConversation	20
WS-Policy	21
WS-Federation	21
Conclusion	23
Glossary	23
Technical References	25

Introduction

The growth of partnerships into e-business networks is the most significant trend in the evolution of Internet commerce. Some of the most successful global enterprises have achieved a very high level of coordination between their own information technology (IT) systems and those of their customers, suppliers and partners.

In business-to-consumer (B2C) environments, where end-users communicate with a single enterprise that simultaneously presents products or services from multiple partners, access to shared resources must be secure and structured to meet the requirements of each partner in the business relationship, while meeting end-users' needs.

In application-to-application (A2A) or business-to-business (B2B) environments, where Web services are increasingly used, remote or partner access to corporate data and applications must be achieved securely and seamlessly.

Effective identity federation benefits both users and enterprises. It provides the end-user with a seamless cross-domain Internet experience through single sign-on (SSO) and it allows the enterprise to expose resources to a larger class of users not directly administered by the enterprise.

Several standards address various aspects of identity federation (SSO, trust, attribute sharing, etc.). Some of those standards combine to provide the basis for an identity federation framework, but there are still overlaps and competition between emerging specifications, which makes purchase decisions a challenge.

Document Purpose and Scope

The first part of this white paper discusses the basic concepts underlying identity federation, namely authentication, authorization and single sign-on (SSO) using a real-world example.

The second part introduces browser-based and document-based identity federation by describing typical use case scenarios.

The third part describes the various standards and industry initiatives that address identity federation and how some of these standards can co-exist and cooperate to offer efficient solutions to the problems encountered with identity federation.

This document is aimed at IT staff, application development managers, security architects and technical decision makers who need to learn more about identity federation and the industry standards that formalize federated environments.

Some sections of this document require a basic knowledge of the Extensible Markup Language (XML) and related XML technologies (XML Schema and the Simple Object Access Protocol (SOAP)).

Terms and concepts not directly defined in the text are explained in a short glossary provided at the end of the document, together with a list of technical references.

Federation Requirements

It is virtually impossible to rely on a universal point of control for identity information. In other words, no single security administrator has the responsibility to authenticate all users and manage their accounts. In some cases, companies have multiple identity repositories for their applications, thus creating a corporate infrastructure fragmented in silos of activities. In addition, when companies do business with each other, they need to exchange information about their respective users in a trusted way.

Identity federation can be relative to a single company (users of that company securely access the company's resources based on their identity information). Identity federation can span several companies, a network of federations as it were, whereby trust must be established between the multiple companies doing business together.

Companies involved in identity federation establish trusted relationships, allowing their respective users to access resources hosted by a business partner. In this case, companies issue security tickets to their users that can be processed by relying parties.

Identity federation provides a foundation for validating users (or services) from various organizations that are part of a network of business partners. In this way, users (or services) can seamlessly access resources provided by those trusted partners.

The identity federation requirements are as follows:

- Define a framework built on industry standards (data format, message structure) that is independent of specific implementations (client type or server type) and network protocols.
- Provide the ability for business partners to exchange information about their users in a secure way.
- Protect the privacy of users within a federation, i.e., keep user identity information secret and conform to international privacy regulations.
- Allow each company in a federation to manage the identities of their own users without relying on a centralized third-party.
- Provide standard security information descriptions or use existing standard security tokens.
- Provide a standard protocol to exchange security tokens amongst federation participants.
- Provide a way to establish trust amongst federation participants.

Basic Concepts

Federating users and/or services into trusted relationships starts from basic processes designed to identify a user or a service (Authentication) and granting access rights to an authenticated user or service (Authorization). In addition, successful and unsuccessful authentication and authorization activities must be recorded in ways that can easily be analyzed by system administrators or security experts (Auditing).

The table below describes a simple scenario of a person registering to two conferences, and maps each “real-world”

step to objects and specifications implemented in the network computing world. Topics mentioned in bold in the right-hand column (SAML assertions, WS-Federation, etc.) are explained in the next section of this document.

The very first step of this scenario involves Alice, who goes to a conference center to attend two conferences hosted by different organizations but related by the topics they cover: A Network Security conference is held on the first floor of the conference center, and an XML Security conference is held on the second floor. The organizers of both conferences have decided to work together to make it easy for registrants to attend both conferences.

Real-world steps for attending a conference	Mapping of real-world steps to the network computing world
Alice gets to the reception desk of the Network Security conference. She presents some ID (e.g., a <i>driver's license</i> in the US) to get her pass to the conference.	<i>Driver's license</i> : Credentials , for example username / password , X.509 certificate , etc.
The reception desk employee inspects the <i>list of registered participants</i> for Alice's identity information.	<i>List of registered participants</i> : User store , implemented in an LDAP directory , including Microsoft's Active Directory , a relational database, or some other mainframe implementation.
The reception desk employee finds Alice's entry, with her name and her entitlements (i.e., what presentations Alice is allowed to attend). It so happens that Alice has been registered as a <i>speaker</i> , and speakers at the Network Security conference can go to all the presentations offered at the conference.	<i>Speaker</i> : Role of an entity (or subject) for a particular transaction , based on specific entity attributes . Role-based authorization rules are defined in policies .
Once the reception desk employee has made sure that Alice's identity information matches the registration list, she gives Alice a <i>badge</i> which was issued by the conference organizers.	<i>Badge</i> : Session ticket generated by an issuing authority . This is the security token generated upon successful authentication, for example a SAML assertion . Holders of such session tickets can access any resources they're entitled to.
With her badge, Alice is able to go to any presentation freely, <i>without being registered again at each conference room</i> by the person verifying participants' badges at the entrance of the conference room.	<i>Without being registered again at each conference room</i> : Single sign-on (SSO) based on SAML or Liberty Alliance .
For convenience, Alice is wearing her badge in a <i>badge holder</i> , which she clipped on her lapel.	<i>Badge Holder</i> : Envelopes including security tokens (or session tickets), essentially WS-Security headers in SOAP messages or secure cookies.
On some occasions, Alice wished that her name were not showing on her badge. She would like to preserve her <i>privacy</i> .	<i>Privacy</i> : Entities (e.g., user identities) can be represented by an opaque string of data, such as that provided by Liberty Alliance .
Now Alice wants to also attend presentations at the XML Security conference. <i>Based on the agreement between the two conferences</i> , Alice has access to the XML Security conference.	<i>Based on the agreement between the two conferences</i> : Trusted relationship, based on WS-Trust , WS-Policy , WS-Federation , SAML , or Liberty Alliance .
Alice's entitlements are not the same for the XML Security conference (she's not a speaker there), so <i>she'll be able to only attend XML Security presentations offered for a basic registration</i> .	<i>She'll be able to only attend XML Security presentations offered for a basic registration</i> : Entitlements can be defined in the security token used with the federation framework, such as an attribute statement in a SAML assertion .

We will now put these basic concepts in action via some federation use cases.

Federation Use Cases

Identity federation can be achieved through browsers or using XML documents with Web services.

Browser-Based Scenarios

The following use cases demonstrate different ways of handling user identities to provide end-users with single sign-on across multiple companies involved in a partnership.

Federation Based On Account Linking

In this use case, Workplace.com contracts the management of its employees' health benefits to a partner company, Health.com. To access her account, an employee of Workplace.com authenticates at the employee portal (www.workplace.com) and clicks on a link to view her health benefits at www.health.com. The employee is taken to the Health.com web site and presented with all of her health benefit information without having to sign on to Health.com.

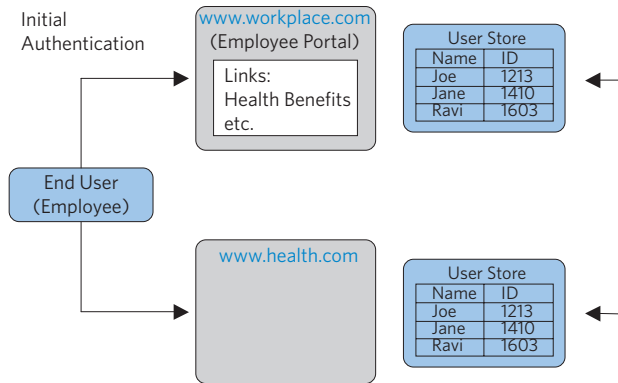


Figure 1: Federation Based On Account Linking

Health.com maintains all health-related information for employees at Workplace.com. To do this, Health.com maintains user identities for every employee of Workplace.com. When an employee of Workplace.com accesses Health.com, an identifier for the employee is passed from Workplace.com to Health.com, in a secure manner. This identifier allows Health.com to determine who the user is and what access to allow for that user.

Federation Based On Roles

In this use case Workplace.com buys parts from a partner company PartsSupplier.com. An engineer of Workplace.com authenticates at the employee portal (www.workplace.com) and clicks on a link to access information at PartsSupplier.com.

Because the user is an engineer at Workplace.com, he's taken directly to the technical documentation and troubleshooting portion of the PartsSupplier.com web site without having to sign on.

When purchasers for Workplace.com authenticate at Workplace.com and click on a link to access information at PartsSupplier.com, they are taken directly to the order portion of the PartsSupplier.com web site without having to sign on.

In either case, the PartsSupplier.com web site is personalized with information such as users names.

PartsSupplier.com does not want to maintain user identities for all employees of Workplace.com. However, PartsSupplier.com must control access to sensitive portions of its web site. To do this, PartsSupplier.com maintains a limited number of profile identities for Workplace.com users.

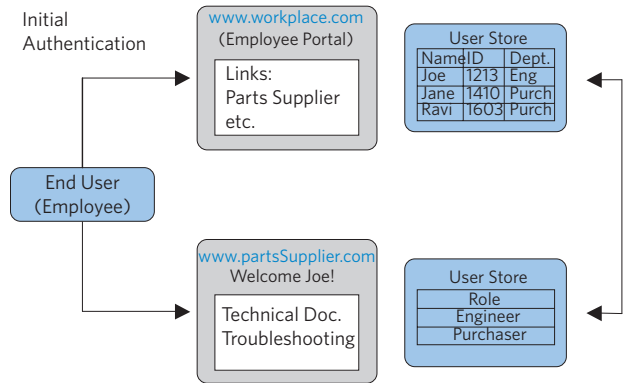


Figure 2: Federation Based On Roles

In this case, one profile identity is maintained for engineers, and one profile identity is maintained for purchasers. When an employee of Workplace.com accesses PartsSupplier.com, user attributes are sent from Workplace.com to PartsSupplier.com in a secure manner. These attributes define the role of the user and determine what profile identity is used to control access at PartsSupplier.com.

Attributes, such as user name, are passed from Workplace.com to PartsSupplier.com to personalize the interface for the individual user.

Federation With No Local User Account

In this use case, Workplace.com offers employee discounts by partnering with Bargains.com.

An employee of Workplace.com authenticates at an employee portal (www.workplace.com) and clicks on a link to access discounts at Bargains.com. The employee is taken to the Bargains.com web site and presented with the discounts available for employees of Workplace.com without having to sign on to the Bargains.com web site.

In this use case, Bargains.com does not maintain any identities or roles for employees of Workplace.com. Bargains.com simply allows all employees of Workplace.com to access Bargains.com as long as they have been authenticated at Workplace.com.

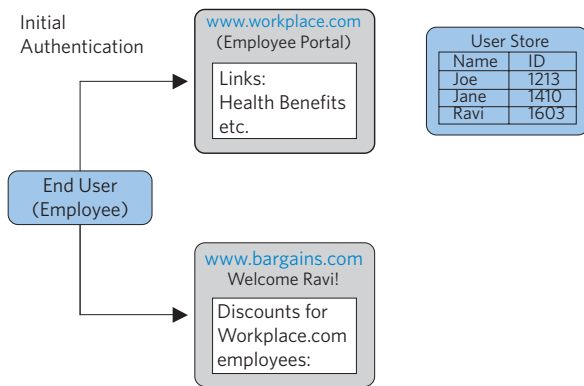


Figure 3: Federation With No Local User Account

When an employee of Workplace.com accesses Bargains.com, authentication information is sent from Workplace.com to Bargains.com in a secure manner. This authentication information is used to allow the user to access Bargains.com. Additional attributes, such as user name, are passed from Workplace.com to Bargains.com to personalize the interface for the individual user.

Document-Based Scenarios

Document-based federation is realized over Web services flows.

Chained Web Services

In this use case, Workplace.com has a purchasing agreement with PinSupplies.com, and PinSupplies.com has a business relationship with E-Ship.com.

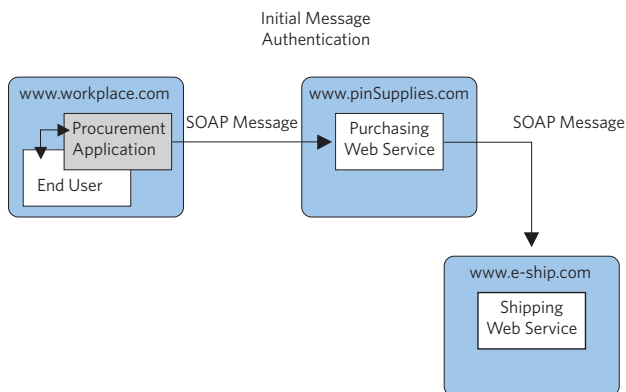


Figure 4: Chained Web Services

The end-user logs on to her procurement application with her username and password. The procurement application provides a list of various suppliers for Workplace.com. The end-user clicks on the PinSupplies button and is presented with a purchase order in an HTML page. She fills out the purchase order and then clicks the Submit button on the HTML form.

The procurement application turns the HTML form into an XML document that it inserts in the envelope body of a SOAP message. The procurement application then inserts the end-user's credentials in the envelope header of the SOAP message, together with the Workplace.com customer identity.

The procurement application posts the SOAP message to the PinSupplies.com Purchasing Web service.

The Purchasing Web service (or a security application on its behalf) authenticates the incoming SOAP message and processes the request. When the purchasing process is complete, the Purchasing Web service makes a request to E-Ship.com using a SOAP message. The SOAP message includes a PinSupply.com security token in the envelope header and the list of items to be shipped as well as the end-user's shipping information in the envelope body.

The Shipping Web service (or a security application on its behalf) authenticates the request and processes the shipment order.

Authentication Service

www.example1.com is an identity provider, i.e., Example1.com generates security tokens to be used to access resources hosted by service providers. In this scenario, the end-user may or may not be an employee of Example1.com.

www.example2.com is any service provider that accepts credentials produced by www.example1.com.

Example1.com and Example2.com are two separate companies (different domains), but they could also be two departments of the same company.

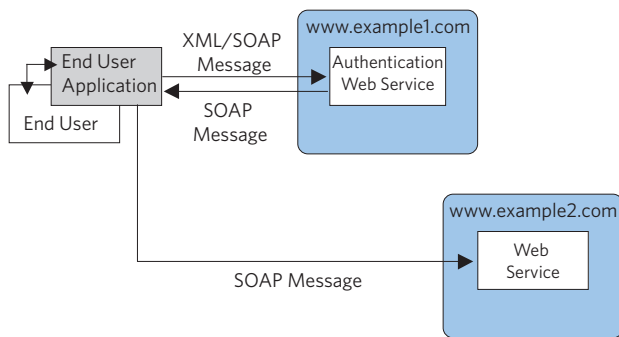


Figure 5: Authentication Service

The end-user's application sends an XML request to Example1.com including the end-user's credentials. Credentials may be a username / password or cryptographic information (e.g., an X.509 certificate).

Once Example1.com gets the request, it processes it and generates a security token for the user, signed with the Example1.com enterprise private key.

Example1.com then returns the signed security token in a raw XML document or a SOAP message.

The end-user's application inserts the security token in a SOAP request and posts the request to a Web service at Example2.com.

The Web service at Example2.com (or a security application on its behalf) authenticates the request (i.e., ensures that it comes from Example1.com), and validates the end-user's authentication information.

Federation Standards

There is no single industry standard meeting all of the federation requirements. As mentioned earlier, federation involves description of identities (i.e., security tokens), protocols to exchange security tokens, preservation of privacy, and establishment of trust.

This section describes the main standards and industry initiatives involved in identity federation and trust:

- Passport
- Kerberos
- SAML
- Liberty Alliance
- Web Services Specifications

Passport

Passport (or .NET Passport) is Microsoft's centralized authentication service. A user can create a Passport at www.passport.com. A Passport includes credentials (email address and password), profile data (birth date, address, etc.), and (optional) wallet information (e.g., credit card number). With a Passport, you can subscribe to any site that is Passport-enabled.

When you visit a Passport-enabled web site, you're redirected to www.passport.com for authentication. Logging in to Passport sets a few cookies on your browser, some from the www.passport.com domain, others from the visited site's domain. All log-in cookies are deleted from the browser upon log-out.

When a user gets a Passport, the user's account is assigned a unique, 64-bit, opaque Passport User ID (PUID). This PUID is used by visited sites as the authentication credentials. Since the initial release of Passport, Microsoft has narrowed the strategic focus of this service and no longer positions it as a general authentication service. It is now used by Microsoft as an authentication service for Microsoft only web properties.

Kerberos

Kerberos is a cross-platform authentication and single sign-on system. Kerberos was originally designed at MIT in the late 1980s. The current release is Kerberos 5, implemented on various Unix platforms, Apple OS X, and Microsoft Windows (Kerberos is the default protocol used by Microsoft to authenticate users on a Windows 2000-2003 network).

Kerberos has its own terminology:

- **Principal:** an identity for a user (i.e., a user is assigned a principal), or an identity for an application offering Kerberos services.
- **Realm:** a Kerberos server environment; a Kerberos realm can be a domain name such as `example.com` (by convention expressed in uppercase).

For instance, Example.com could own a Kerberos realm called `EXAMPLE.COM` and assign Alice a principal such as `alice@EXAMPLE.COM`.

Kerberos Components

The Kerberos protocol provides mutual authentication between two entities relying on a shared secret (symmetric keys).

Kerberos involves a client, a server, and a trusted party to mediate between them, called the Key Distribution Center (KDC). Each Kerberos realm has at least one KDC. KDCs come in different packages based on the operating platform used (for example, on Windows, the KDC is a domain service, like the IIS web server).

The KDC includes:

- A database of all the principals together with their encryption keys (for example, Microsoft's Active Directory hosts the Kerberos database in its LDAP store),
- An Authentication Service (AS) which issues a Ticket Granting Ticket (TGT) to clients who log on to a Kerberos realm (the TGT is a proof of authentication for a user that successfully logged into Kerberos),
- A Ticket Granting Service (TGS) which issues session tickets to TGT-holding clients for a specific application server or resource (a session key is used for the immediate session between a client and a server).

Kerberos tickets (issued by the KDC) are encrypted data structures including a list of the IP addresses that a ticket can be used from, the principal's information, a validity period, and the encryption key used for the session.

Kerberos Process

A client (a user or an application server) that wants to use Kerberos authentication needs a Kerberos client package. When "kerberized" clients log in to their workstation, they can get a service ticket that they can reuse with other Kerberos resources without having to repeatedly log-in (single sign-on).

Typically, when a client wants to communicate with a server, the client sends a request to the KDC, and the KDC generates a session key allowing the client and the server to authenticate each other.

1. The (kerberized) client logs on to a Kerberos realm using a password.
2. The password is hashed by the Kerberos server to produce the user's encryption key, which is compared by the KDC to the user's key stored in the Kerberos database.
3. If the comparison is successful, the Authentication Service of the KDC issues a ticket-granting ticket (TGT).
4. When the client needs to contact a server, it sends the TGT to the KDC.
5. The Ticket Granting Service of the KDC returns a session ticket to the client.
6. The client can now use the session ticket to access the server.

Kerberos presents many advantages, such as cryptographic authentication (privacy) and cross-domain single sign-on, but it does not support authorization and requires clients to be "kerberized." In addition, all Kerberos 5 environments (Unix, Apple, Windows) should in theory be compatible (for example, a Unix kerberized client should be able to send requests to a Windows KDC), but it's not always the case.

A Kerberos profile of WS-Security is currently defined by the WS-Security technical committee at OASIS to allow business partners to use Kerberos in service-oriented architectures (Web services).

Security Assertion Markup Language (SAML)

In January 2001, Netegrity, now a part of CA, along with other companies, created the OASIS Security Services Technical Committee (SSTC) which culminated in the adoption of SAML as an industry standard in November 2002. SAML 2.0, the current version of SAML, was approved by the OASIS Board in March of 2005.

Introduction

SAML is an open, application-level, standard framework for sharing security information on the Internet through XML documents.

SAML's scope is based on three use cases:

- **Browser-driven interaction:** A user authenticates at a Source web site. The Source web creates a security context for the user, and the user can access protected resources at Destination web sites without repeated log-in.
- **XML message transfer:** The buyer of Enterprise A is allowed to purchase items from Enterprise B if authorized for that particular transaction. In this case, the buyer (at Enterprise A) goes to an authority known to the buyer and the seller, gets authenticated and qualified, and then can carry out the transaction with the seller (Enterprise B).
- **Remote authorization:** A user places an order with a supplier. The supplier checks the user's entitlements with a remote service (e.g., a credit-rating company) to grant authorization.

In a browser-based single sign-on environment, no user directory duplication or synchronization is necessary. Security information (in the form of a SAML assertion) "travels" with the user. As a result, users coming from a source web site do not have to be registered at the destination web site. User information does not have to be duplicated at each site involved in a trusted e-business network.

SAML Architecture

SAML takes advantage of the flexibility provided by XML Schema:

- Inheritance (abstract types to promote schema extensions allowing users to create their own assertions).
- Strong typing of elements.
- Inclusion of existing XML vocabularies (or “namespaces”), for example XML Signature for handling public key information, or more vendor-specific vocabularies such as SM as shown in Listing 1 below.

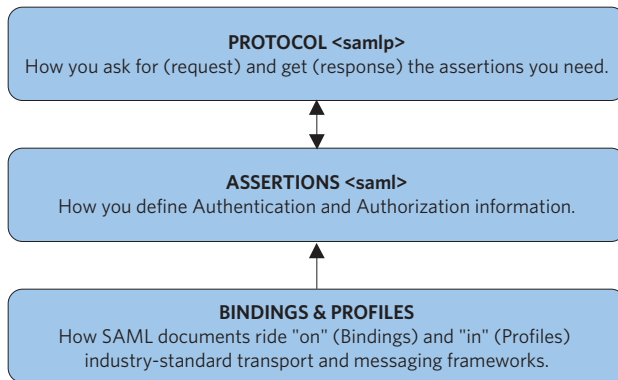


Figure 6: SAML Architecture

The SAML specification includes three distinct parts: Assertions, Protocol, and Bindings & Profiles. Assertions and Protocol are defined by two separate XML schemas, `saml` and `samlp`, respectively.

SAML Assertions

SAML assertions are XML documents (see *Listing 1* below) encoded in an XML schema. SAML assertions can be digitally signed.

SAML assertions are issued by “authorities” which can be security applications or access control infrastructures such as eTrust SiteMinder or eTrust TransactionMinder products.

A SAML assertion makes statements about an entity or “subject” (a human user or a service). All SAML assertions include the following common information:

- Issuer ID and issuance timestamp
- Assertion ID
- Subject
 - Name
 - Optional subject confirmation (e.g., a public key)
- Conditions (under which an assertion is valid), optional
- Advice (on how an assertion was made), optional

SAML assertions can include three types of statements:

- **Authentication statement:** issued by an authentication authority upon successful authentication of a subject. It asserts that Subject S was authenticated by Means M at Time T.
- **Attribute statement:** issued by an attribute authority, based on policies. It asserts that Subject S is associated with Attributes A, B, etc. with values a, b, etc.
- **Authorization decision statement:** issued by an authorization authority which decides whether to grant the request by Subject S, for Action A (e.g., read, write, etc.), to Resource R (e.g., a file, an application, a Web service), given Evidence E.

Listing 1: SAML Assertion Example

```
01 <saml:Assertion AssertionID="172.26.5.45.1070314815561"
    IssueInstant="2003-12-01T21:40:15.561Z"
    Issuer="http://www.netegrity.com/SiteMinder"
    MajorVersion="1" MinorVersion="0"
    xmlns:SM="http://www.netegrity.com/SiteMinder"
    xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
02   <saml:Conditions NotBefore="2003-12-01T21:39:45.561Z" NotOnOrAfter="2004-02-
09T08:20:45.561Z"/>
03   <saml:AuthenticationStatement
    AuthenticationInstant="2003-12-01T21:40:15.561Z"
    AuthenticationMethod="Unspecified">
04     <saml:Subject>
05       <saml:NameIdentifier
    Format="urn:oasis:names:tc:SAML:1.0:assertion"
    NameQualifier="www.netegrity.com">
06         uid=mchanliau,ou=People,dc=netegrity,dc=com
07       </saml:NameIdentifier>
08       <saml:SubjectConfirmation>
09         <saml:ConfirmationMethod>
10           urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
11         </saml:ConfirmationMethod>
12         <saml:SubjectConfirmationData/>
13       </saml:SubjectConfirmation>
14     </saml:Subject>
15   </saml:AuthenticationStatement>
16   <saml:AttributeStatement>
17     <saml:Subject>
18       <saml:NameIdentifier
    Format="urn:oasis:names:tc:SAML:1.0:assertion"
    NameQualifier="www.netegrity.com">
19         uid=mchanliau,ou=People, dc=netegrity,dc=com
20       </saml:NameIdentifier>
21       <saml:SubjectConfirmation>
22         <saml:ConfirmationMethod>
23           urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
24         </saml:ConfirmationMethod>
25         <saml:SubjectConfirmationData/>
26       </saml:SubjectConfirmation>
27     </saml:Subject>
28     <saml:Attribute AttributeName="SMContent"
    AttributeNamespace="http://www.netegrity.com/SiteMinder">
29       <saml:AttributeValue>
30         <SM:SMContent xmlns:SM="http://www.netegrity.com/SiteMinder">
31           <SM:SMsession>
32             <SM:SessionID>Sko0dBuvmfwcSlBY4AOLQkD5Rr8=</SM:SessionID>
33             <SM:timeIn>30</SM:timeIn>
34           </SM:SMsession>
35           <SM:SMlogin>
36             <SM:UserDN>uid=mchanliau,ou=People, dc=netegrity,dc=com</SM:UserDN>
37             <SM:Username>mchanliau</SM:Username>
38           </SM:SMlogin>
39         </SM:SMContent>
40       </saml:AttributeValue>
41     </saml:Attribute>
42   </saml:AttributeStatement>
43 </saml:Assertion>
```

Listing 1 includes an authentication statement and an attribute statement. The attribute value (Line 41) is defined by importing another namespace (SM or eTrust SiteMinder).

Authentication, attribute, and authorization decision authorities may be hosted by the same product (e.g., eTrust SiteMinder or eTrust TransactionMinder).

SAML Protocol

The SAML Protocol defines the interaction between a SAML requester and a SAML responder. The SAML Protocol is encoded in an XML schema as a set of request-response pairs.

A SAML request is made by a SAML-aware client, a SAML response is returned by a security service. A SAML request may include queries for authentication, attributes, or authorization decision. All types of SAML requests are met with a common SAML response.

An authentication query is of the type "Please provide the authentication information for this subject, if any." The requester and the responder must have a trust relationship (i.e., they must be talking about the same subject).

An attribute query is of the type "Please provide information on the listed attributes for this subject." If the requester is denied access to some of the attributes, only those for which access is permitted are returned.

An authorization-decision request is of the type "Is the subject allowed to access the specified resource in the specified manner, given this evidence?"

SAML responses may contain one or more assertions including status information.

SAML Bindings and Profiles

The SAML Protocol Bindings specify how SAML request-response message exchanges are mapped to standard messaging protocols. The SAML specification currently provides the SOAP-over-HTTP binding of SAML, whose purpose is to use SOAP to query a SAML authority and get back a SAML response.

SAML Profiles specify how SAML assertions are inserted in, and extracted from, a message framework or protocol. Currently, the SAML specification includes two Web Browser profiles, and a WS-Security profile (see the WS-Security section in this document).

The Web Browser profiles of SAML include a "Pull" method and a "Push" method.

- In the Pull method, a SAML artifact (a fixed-length, randomly-generated reference to a SAML assertion) is passed between sites by the browser on URL query strings. The corresponding assertion is then "pulled" by the destination site from the source site using the SOAP-over-HTTP binding of SAML.

- In the Push (or POST) method, HTML forms include SAML assertions "pushed" or "POSTed" to the destination site. The destination site then makes an authentication decision based on the SAML assertion information.

Liberty Alliance

The Liberty Alliance Project (loosely referred to as Liberty Alliance or Liberty) is an industry organization started in September 2001 that currently includes over 160 member companies worldwide. The purpose of the Liberty Alliance is to create a set of specifications for identity federation in network environments.

Introduction

Liberty includes three (cumulative) phases: Liberty Phase 1, Liberty Phase 2, and Liberty Phase 3.

- **Phase 1:** Identity Federation Framework (ID-FF) – Federated network identity services including single sign-on/out, opt-in account linking, privacy. This phase is now merged with OASIS SAML 2.0.
- **Phase 2:** Identity Web Services Federation (ID-WSF) – Framework for interoperable federated network identity services including identity data service definition, identity service discovery and invocation, attribute sharing, interaction, security profiles.
- **Phase 3:** Identity Services Interface Specification (ID-SIS) – Interoperable identity services providing implementation of ID-WSF definitions in specific Web services, e.g., personal profile, employee profile, etc.

All three phases are designed to build on each other: Phase 2 (ID-WSF) relies on Phase 1 (ID-FF), and Phase 3 (ID-SIS) instantiates the ID-WSF framework.

Liberty and Industry Standards

Liberty leverages the following industry standards:

- HTTP for application-level transport.
- SSL/TLS for transport-level message authentication, confidentiality, and integrity.
- SOAP for exchanging messages between entities (ID-FF authentication, ID-WSF security mechanisms).
- WS-Security for providing secure SOAP envelope headers (ID-WSF supports the SAML profile of WS-Security).
- XML Encryption for message-level confidentiality and privacy (not mandatory).
- XML Signature for message-level integrity (not mandatory).
- SAML assertions for defining Liberty security tokens (Note: Liberty tokens are extended SAML assertions).
- SAML protocol for requesting and obtaining SAML assertions.
- WAP (Wireless Application Protocol – WAP uses the Wireless Markup Language (WML) to display information in a WAP browser, for example a cellular telephone set).

Liberty Terminology

The Liberty Alliance has its own terminology. The following definitions (in alphabetical order) are adapted from the Liberty glossary.

- **Anonymity:** The ability for a service to request attributes from a user (e.g., ZIP code, birth date, gender, etc.) without needing to know the user's identity.
- **Attribute Provider (AP):** An ID-WSF Web service that hosts attribute data, for example identity personal file (ID-PP) information.
- **Circle of Trust (CoT):** A federation of service providers and identity providers that have business relationships based on the Liberty architecture and operational agreements, and with whom users can transact business in a secure and apparently seamless environment (See Figure 7).

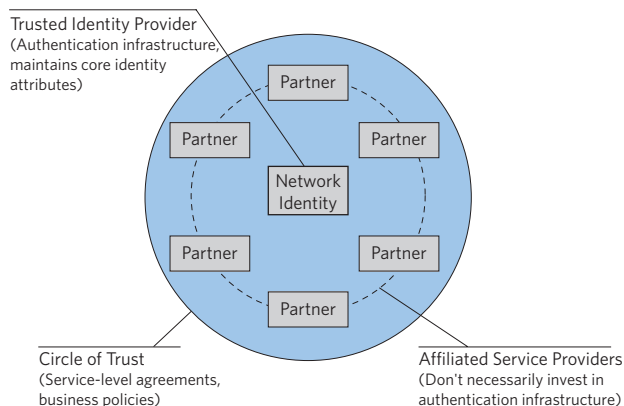


Figure 7: Identity and Service Providers in Circle of Trust

- **Federated Network Identity:** The grouping of isolated user accounts (or local identities) into a cohesive principal network identity.
- **Federation:** An association comprising any number of service providers and identity providers.
- **Identity Provider (IdP):** A Liberty-enabled entity that creates, maintains, and manages identity information for principals and provides principal authentication to other service providers within a circle of trust.
- **Liberty-Enabled Client or Proxy (LECP):** A Liberty-enabled client is a client that has, or knows how to obtain, knowledge about the identity provider that a principal wishes to use with the service provider. A Liberty-enabled proxy is an HTTP proxy (typically a WAP gateway) that emulates a Liberty-enabled client.
- **Network Identity:** The overall set of attributes obtained from a principal's various accounts (e.g., name, email address, credit card number, notification preferences, etc.).
- **Principal:** An individual user, a group of individuals, a corporation, or a component of the Liberty architecture.

- **Pseudonym:** An arbitrary name assigned by the identity provider or service provider to identify a principal to a given relying party so that the name has meaning only in the context of the relationship between partners.
- **Service Provider (SP):** An entity that provides services and/or goods to principals.

ID-FF

The ID-FF module is the foundation of the Liberty architecture (ID-WSF and ID-SIS rely on ID-FF). A basic ID-FF environment minimally includes three parts: an identity provider (e.g., a telecommunication company), a service provider (e.g., an online retailer, a financial institution, a government agency), and a user agent. The user agent is a thin client (e.g., a standard browser) or a Liberty-enabled client or proxy (LECP), e.g., a wireless (cellular) telephone handset.

Upon successful authentication of the principal, the identity provider produces a SAML assertion including an authentication statement describing the principal's security context, together with a name identifier (or "handle").

The name identifier is a randomly-generated character string designed to preserve the privacy of the principal and to facilitate account linking at the identity provider and service provider sites.

ID-FF supports the following functionality, defined in profile categories:

- **Single Sign-On and Federation:** How a service provider obtains a SAML assertion from an identity provider.
- **Name Registration:** How service providers and identity providers specify the name identifier to be used when communicating with each other about a principal.
- **Federation Termination Notification:** How service providers and identity providers are notified of federation termination.
- **Single Log-Out:** How service providers and identity providers are notified of authenticated session termination.
- **Identity Provider Introduction:** How a service provider discovers which identity provider(s) a principal may be using.
- **Name Identifier Mapping:** How a service provider may obtain a name identifier with which to refer to a principal at a SAML Authority.
- **Name Identifier Encryption:** How a provider may encrypt a name identifier for privacy.

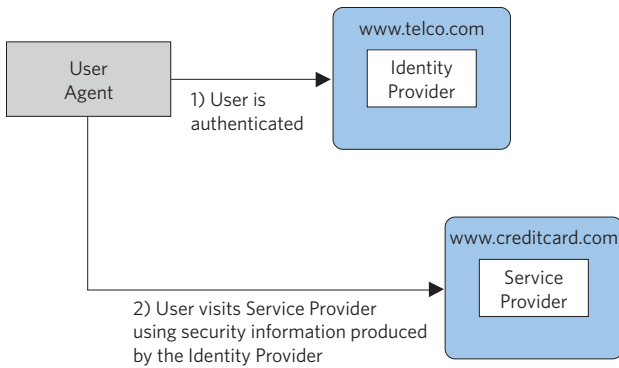


Figure 8: Liberty Single Sign-On

Single Sign-On and Federation

Single sign-on is enabled once a user’s identities at the identity provider and service provider are federated. Identity federation means that a user, for example Alice Smith, is known both at the identity provider (maybe as asmith) and at the service provider (maybe as alices). Each site must know Alice’s name identifier.

A federation is optionally created the first time a user logs in to a service provider using an identity provider (so-called “opt-in account linking”). The identity provider and the service provider can create their own handle (name identifier) for a user. Alternatively, a service provider may not create a handle for a user if it doesn’t want to maintain information about that user. The handle used by the identity provider and service provider links the user’s accounts at both sites.

Users may choose to federate in one of two ways (or a combination of both):

- One identity provider and multiple service providers: The identity provider creates a name identifier and manages the user’s identity for the federated service providers.
- One service provider and multiple identity providers: The user accesses a service provider from different clients, thus requiring different identities.

Federation can be undone by either the identity provider or the service provider using the Federation Termination Notification protocol.

Communication Between Identity Provider and Service Provider

To enable single sign-on, a service provider obtains a SAML assertion for a user agent from an identity provider.

Communication between the identity provider and service provider sites can be supported in two ways, based on the Web Browser Profiles of SAML.

- Back channel communication using the SAML artifact-based, pull model: Information is passed by sending SOAP messages from identity provider to service provider sites directly.
- Front channel communication using the push (or POST) model: Information is passed between the identity provider and service provider sites over HTTP redirects through the user agent (browser).

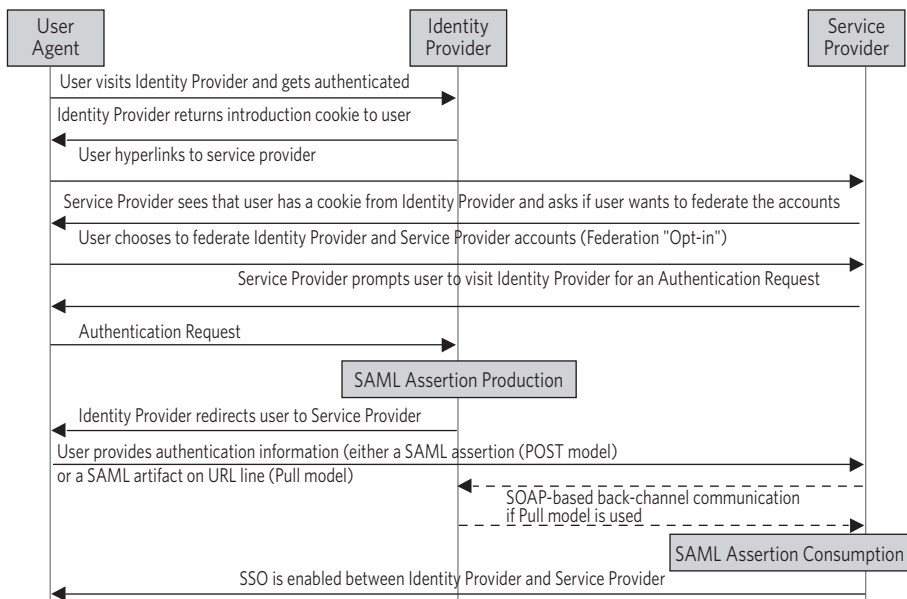


Figure 9: ID-FF Single Sign-On and Federation

Listing 2 describes the SAML request for an authentication assertion. The signed request is included in the SOAP envelope body.

Listing 2: SAML Request

```
01 POST/authn HTTP/1.1
02 Host:idp.example.com
03 Content-type:text/xml
04 Content-length:nnnn
05 <soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">
06   <soap-env:Header/>
07   <soap-env:Body>
08     <samlp:Request xmlns="urn:oasis:names:tc:SAML:1.0:protocol"
          xmlns:lib="urn:liberty:iff:2003-08"
          xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
          xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          IssueInstant="2002-12-12 T10:08:56Z"
          MajorVersion="1"
          MinorVersion="1"
          RequestID="e4d71c43-c89a-426b-853e-a2b0c14a5ed8"
          id="ericssonb6dc3636-f2ad-42d1-9427-220f2cf70ec1">
09       <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
10         <ds:SignedInfo>
11           <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
12             </ds:CanonicalizationMethod>
13           <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1">
14             </ds:SignatureMethod>
15           <ds:Reference URI="#ericssonb6dc3636-f2ad-42d1-9427-220f2cf70ec1">
16             <ds:Transforms>
17               <ds:Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature">
18                 </ds:Transform>
19               <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
20                 </ds:Transform>
21               </ds:Transforms>
22             <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1">
23               </ds:DigestMethod>
24             < ds:DigestValue>+k6Tno1GkIPKZlpUQVyok8dwkuE=</ds:DigestValue>
25             </ds:Reference>
26           </ds:SignedInfo>
27           <ds:SignatureValue>wXJMVoP01 - - - V1jFnWJPyO</ds:SignatureValue>
28           <ds:KeyInfo>
29             <ds:X509Data>
30               <ds:X509Certificate>MIIDMT - - - CCApqqA</ds:X509Certificate>
31             </ds:X509Data>
32           </ds:KeyInfo>
33           </ds:Signature>
34         <samlp:AssertionArtifact>
35           AAM1uXw6+f+jyA/4XuFHqPl7QDvc/LIQL9+t7YQtG1Gwk9bph0Adl+o+
36         </samlp:AssertionArtifact>
37       </samlp:Request>
38     </soap-env:Body >
39 </soap-env:Envelope>
```

Listing 3 describes the SAML response including the requested SAML assertion in the SOAP envelope body. The name identifier generated by the identity provider is shown on Line 29.

Listing 3: SAML Response

```
01 HTTP/1.1 200 OK
02 Content-Type:text/xml
03 Content-Length:nnnn
04 <soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">
05   <soap-env:Header/>
06   <soap-env:Body>
07     <samlp:Response xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
                          InResponseTo="RPCUk211+GVz+t11LURp51oFvJXk"
                          IssueInstant="2002-10-31T21:42:13Z" MajorVersion="1"
MinorVersion="1"
                          Recipient="http://localhost:8080/sp"
                          ResponseID="LANWfL2xLybnc+BCwgY+p1/vIVAj">
08       <samlp:Status>
09         <samlp:StatusCode xmlns:gns="urn:oasis:names:tc:SAML:1.0:protocol"
                              Value="gns:Success">
10           </samlp:StatusCode>
11         </samlp:Status>
12         <saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
                              xmlns:ds=" http://www.w3.org/2000/09/xml dsig#"
                              xmlns:lib="urn:liberty:iff:2003-08"
                              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                              AssertionID="SqMC8Hs2vJ7Z+t4UiLSmhKOSU00U"
                              InResponseTo="RPCUk211+GVz+t11LURp51oFvJXk"
                              IssueInstant="2002-10-31T21:42:13Z"
Issuer="http://localhost:8080/idp"
                              MajorVersion="1" MinorVersion="2"
                              xsi:type="lib:AssertionType">
13           <saml:Conditions NotBefore="2002-10-31T21:42:12Z" NotOnOrAfter="2002-10-
31T21:42:43Z">
14             <saml:AudienceRestrictionCondition>
15               <saml:Audience>http://localhost:8080/sp</saml:Audience>
16             </saml:AudienceRestrictionCondition>
17           </saml:Conditions >
18           <saml:AuthenticationStatement AuthenticationInstant="2002-10-31T21:42:13Z"
AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password"
                              xsi:type="lib:AuthenticationStatementType">
19             <saml:Subject xsi:type="lib:SubjectType">
20               <saml:NameIdentifier Format="urn:liberty:iff:nameid:federated">
21                 C9FfGouQdBJ7bpkismYgd8ygeVb3PlWK
22               </saml:NameIdentifier>
23             <saml:SubjectConfirmation>
24               <saml:ConfirmationMethod>
25                 urn:oasis:names:tc:SAML:1.0:cm:artifact-01
26               </saml:ConfirmationMethod>
27             </saml:SubjectConfirmation>
28             <lib:IDPProvidedNameIdentifier>
29               C9FfGouQdBJ7bpkismYgd8ygeVb3PlWK
30             </lib:IDPProvidedNameIdentifier>
31           </saml:Subject>
32         </saml:AuthenticationStatement>
```

```

33     <ds:Signature>
34         <ds:SignedInfo> - - - </ds:SignedInfo>
35         <ds:SignatureValue> H+q3nC3jUa1 - - -jluKUVkcC</ds:SignatureValue>
36     </ds:Signature>
37 </saml:Assertion>
38 </samlp:Response>
39 </soap-env:Body>
40 </soap-env:Envelope>

```

ID-FF 1.2 and SAML 2.0

Since the initial development of ID-FF, it has officially been rolled into SAML 2.0, thus merging two standards into a powerful customer solution, and allowing the Liberty Alliance Project to concentrate on the upper layers of the Liberty architecture (ID-WSF and ID-SIS).

ID-WSF

As seen in the previous section, ID-FF addresses the problems of federated network identity. ID-WSF builds upon ID-FF to provide a framework for identity-based Web services in a federated network identity environment.

ID-WSF Use Case Scenario

In a typical situation, a user may have a corporate account with his company and a personal account with an Internet service provider (ISP) describing his employee profile and personal profile, respectively. The user may choose to federate his corporate and personal accounts.

The user may want to set permissions at his ISP such that he is asked for permission before personal profile attributes can be released to service providers affiliated with the ISP. The ISP uses the Liberty Interaction Service (described in the next section) to query the user for permission to release selected personal profile attributes.

Alternatively, the user may set his personal profile service on a mobile device. The user can set permissions for his telephone area code, gender, and age, for example, to be available to service providers, which allows him to get personalized service when he visits those service providers.

Employee and personal profile information can be provided by data services implemented as Web services defined by ID-WSF, hosted by attribute providers.

ID-WSF Overview

ID-WSF defines a SOAP-based invocation framework (i.e., ID-WSF defines a SOAP binding whereby SOAP header blocks and processing rules enable invocation of identity services via SOAP requests and responses). ID-WSF does not specify any contents for the SOAP body, allowing the development of identity services within the context of ID-WSF.

ID-WSF provides the following services to ID-FF-based circles of trust:

- **ID-WSF Security Profiles:** Define the requirements for securing discovery and use of identity services. The ID-WSF Security Profiles specification includes security requirements for privacy as well as integrity and confidentiality of messages exchanged between service providers.
- **ID-WSF Discovery Service:** Allows an entity such as a service provider to dynamically discover a principal's registered identity services (attribute providers). Typically, a service provider queries the Discovery Service, which responds by providing a WSDL file describing the requested identity service.
- **ID-WSF Data Services Template:** A set of protocols (XML schemas) for querying and modifying a principal's attributes exposed by a data service (i.e., a Web service supporting the storage and modification of a principal's data attributes, such as name, email address, etc.).
- **ID-WSF Interaction Service:** Protocols and profiles for interactions enabling an identity service to get permission from a user to allow the identity service to share data with requesting services.
- **Liberty-Enabled User Agent or Device (LUAD):** Describes the profiles and requirements for Liberty-enabled clients interacting with a SOAP-based authentication service.
- **Metadata:** Schemas and protocols are designed to facilitate real-time requests for metadata (e.g., cryptographic keys information), as opposed to requesting that information through out-of-band communication.
- **Reverse HTTP Binding:** Enables an HTTP-bound user agent to receive SOAP requests inside an HTTP response, thus allowing end users to host identity services on their user agents without running an HTTP server or being IP addressable.
- **SOAP Authentication Service:** Defines how to authenticate parties using SOAP messages. This specification also defines an identity-based authentication security token service.

ID-SIS

ID-SIS defines components delivered as Web services. ID-SIS components are built upon the Liberty foundation (ID-FF) and the Liberty framework (ID-WSF).

The ID-SIS module initially consists of two components:

- **ID-SIS Personal Profile (ID-SIS-PP):** supports identity information about principals themselves.
- **ID-SIS Employee Profile (ID-SIS-EP):** supports identity information about principals in the context of their employment.

Liberty will provide additional component definitions to address various identity services that may be required by the industry.

ID-SIS components are instances of the ID-WSF Data Services Template. Because they are delivered as Web services, ID-SIS components are defined by XML schemas and described in WSDL files (a WSDL file defines the request sent to a Web service and the response provided by the Web service as well as the type of transport used; SOAP over HTTP in this case).

Each ID-SIS component defines discovery keywords to be included in discovery registrations and queries as mandated by the ID-WSF Discovery Service. Queries are executed using XPath expressions, as described in the ID-WSF Data Services Template.

WS-Security

The Web Services Security specification (WS-Security) was originally developed by independent vendors. It is now hosted by the OASIS Web Services Security Technical Committee (WSS TC).

WS-Security specifies SOAP security extensions providing data integrity and confidentiality.

WS-Security defines how to attach signature and encryption headers to SOAP messages. It also provides profiles that specify how to insert different types of binary and XML security tokens in WS-Security headers:

- Username / Password digest (defines how a Web services consumer can supply a username as a credential for authentication. The username can be accompanied by a hashed password equivalent, thus providing confidentiality without encryption)
- X.509 certificate
- Kerberos ticket
- SAML assertion
- XrML document (The Right Expression Language (REL) license tokens inserted in WS-Security headers are used for authorization and are based on the Extensible Rights Markup Language (XrML))
- XCBF document (defines how to use the XML Common Biometric Format (XCBF) language for authentication with the WS-Security specification)

Listing 4 shows a SOAP message including a WS-Security header enclosing a Password Digest security token.

Listing 4: WS-Security Header

```
01 <SOAP-ENV:Envelope xmlns:SOAPENV="http://schemas.xmlsoap.org/soap/envelope/
    xmlns:xsd="http://www.w3.org/1999/XMLSchema
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance">
02   <SOAP-ENV:Header>
03     <wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext"
        xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
04       <wsu:Timestamp wsu:Id="WSdaf01a0f-c224-6c1b-90c5-418500160b95">
05         <wsu:Created wsu:Id="WS7672d762-f43a-79c9-2ef8-a8baae0a7e2d">
06           2003-03-30T17:08:20Z
07         </wsu:Created>
08         <wsu:Expires wsu:Id="WSae428212-e8dc-8dc7-f19f-13c644a9738c">
09           2003-09-01T18:28:20Z
10         </wsu:Expires>
11       </wsu:Timestamp>
12       <wsse:UsernameToken wsu:Id="WS3d7dbfc8-e0d0-f5c4-0484-62313ebe5c8a">
13         <wsse:Username>mchanliau</wsse:Username>
14         <wsse:Password
Type="wsse:PasswordDigest">GMqyWWZ8xnxuoBw/XpHUAm3CYCE=</wsse:Password>
15         <wsse:Nonce>8929265</wsse:Nonce>
16         <wsse:Created>2003-03-30T17:08:20Z</wsse:Created>
17       </wsse:UsernameToken>
18     </wsse:Security>
19     <BillingInformation>
20       <CustomerCredentials>
21         <username>mchanliau</username>
22       </CustomerCredentials>
23     </BillingInformation>
24   </SOAP-ENV:Header>
25   <SOAP-ENV:Body>
26     <purchaseOrder orderDate="2002-03-06" xmlns="urn:po-processor">
27       <shipTo country="US">Shipping Information inserted here</shipTo>
28     </purchaseOrder>
29   </SOAP-ENV:Body>
30 </SOAP-ENV:Envelope>
```

WS-Security and SAML

The WS-Security profile of SAML is based on a single interaction between a sender and a receiver.

- The sender (a Web service consumer) obtains one or more SAML assertions and/or assertion identifiers
- The sender adds the assertions and / or assertion identifiers to a SOAP message using WS-Security headers
- The sender sends the SOAP message to the receiver (a Web service provider)
- The receiver processes the assertions and / or assertion identifiers present in the SOAP message

SAML assertions and references to assertion identifiers are contained in the <wsse:Security> element, which in turn is included in the <SOAP-ENV:Header> element as shown in Listing 5.

Listing 5: SAML Assertion in a WS-Security Header

```
01 <SOAP-ENV:Envelope>
02   <SOAP-ENV:Header>
03     <wsse:Security>
04       <saml:Assertion> - - - </saml:Assertion>
05     </wsse:Security>
06   </SOAP-ENV:Header>
07   <SOAP-ENV:Body> - - - </SOAP-ENV:Body>
08 </SOAP-ENV:Envelope>
```

Assertion identifier references and information about assertion retrieval services are included in the `<wsse:SecurityTokenReference>` element. One or more `<saml:AssertionIDReference>` elements holding the assertion identifier references may be included within the `<wsse:SecurityTokenReference>` element. The URI attribute of the `<wsse:Reference>` element specifies the location of a SAML responder implementing the SOAP-over-HTTP binding of SAML.

Listing 6: Referring SAML information in WS-Security

```
01 <wsse:SecurityTokenReference>
02   <saml:AssertionIDReference>
03     XVB12#$21abc - - -
04   </saml:AssertionIDReference>
05   <wsse:Reference URI="http://www.example.com/SAMLservice"/>
06 </wsse:SecurityTokenReference>
```

WS-Security does not support secure communication. This is achieved by the Web Services Secure Conversation specification (WS-SecureConversation), which defines how security context tokens (and their derived keys) are created, propagated, and shared to allow seamless secure communication between parties.

WS-Trust

The Web Services Trust Language (WS-Trust) is a specification jointly developed by independent vendors including CA.

While WS-Security adds security information to a SOAP message by defining how standard security tokens (such as X.509 certificates or SAML assertions) are added to SOAP envelope headers, it assumes that the receiving party involved in the trusted relationship is able to process the security tokens produced by the sending party.

For a seamless transaction, business partners must use security tokens they both understand. This requirement is currently met by a trust agreement between the business partners, defining the type of security tokens they must use in WS-Security headers.

In some cases, however, there is no such agreement and a business partner (or a Web service on their behalf) can require that an incoming message prove a set of claims (e.g., name, password, key, etc.). If the message arrives without having the required proof of claims, the Web service should reject the message.

The WS-Trust security model involves claims, policies (based on WS-Policy described later in this document), and standard security tokens.

A Web service requestor that does not have the necessary security token to prove the claims expressed in the partnership agreement can request the required security token from an authority referred to as a Security Token Service (STS). An STS is designed to issue security tokens used to broker trust relationships between different trust domains.

WS-Trust defines the protocol used for security token acquisition or challenges to a requestor to ensure the validity of a security token.

WS-Trust (wst) uses the following namespaces:

- SOAP 1.1 Envelope (S11)
- SOAP 1.2 Envelope (S12)
- WS-Security Extensions to SOAP (wsse)
- WS-Security Utility (wsu)
- WS-Policy (wsp)
- WS-Addressing (wsa) (WS-Addressing provides transport-neutral mechanisms to address Web services and messages)
- XML Signature (ds)
- XML Encryption (xenc)

WS-SecureConversation

The Web Services Secure Conversation Language (WS-SecureConversation) is jointly developed by independent vendors including CA.

WS-SecureConversation extends WS-Security by defining the creation and sharing of security contexts between communicating parties using security context tokens (SCT).

WS-SecureConversation also specifies how derived keys (used for signing and encrypting messages associated with the security context) are computed and passed. Using a common secret, communicating parties define different key derivations, possibly using different algorithms for deriving keys.

WS-SecureConversation (wsc) uses the following namespaces:

- SOAP 1.1 Envelope (S11)
- SOAP 1.2 Envelope (S12)
- WS-Security Extensions to SOAP (wsse)
- WS-Security Utility (wsu)
- WS-Trust (wst)
- WS-Policy (wsp)
- XML Signature (ds)
- XML Encryption (xenc)

Listing 7: Simple WS-SecureConversation Example

```
01 <S11:Envelope xmlns:s11="..." xmlns: wsse="..." xmlns:wsu="..." xmlns:wst="..."
    xmlns:wsp="..."
                                xmlns:wsc="..." xmlns:ds="..." xmlns:xenc="...">
02   <S11:Header>
03     ...
04     <wsse:Security>
05       <wsc:SecurityContextToken wsu:Id="MyID">
06         <wsc:Identifier>uuid:...</wsc:Identifier>
07       </wsc:SecurityContextToken>
08       <ds:Signature>
09         ...
10       <ds:KeyInfo>
11         <wsse:SecurityTokenReference>
12           <wsse:Reference URI="#MyID"/>
13         </wsse:SecurityTokenReference>
14       </ds:KeyInfo>
15     </ds:Signature>
16   </wsse:Security>
17 </S11:Header>
18 <S11:Body wsu:Id="MsgBody">
19   ...
20 </S11:Body>
21 </S11:Envelope>
```

Lines 05-07 define an SCT associated with the message.

Lines 08-15 define a digital signature associated with the SCT.

Line 13 provides a URI link to the SCT specified on lines 05-07.

An SCT needs to be created and shared between the communicating parties before being used. A security context can be established in three ways:

- The SCT is created by a security token service: The security context initiator uses WS-Trust to request an SCT from a security token service.
- The SCT is created by one of the communicating parties: The security context initiator creates an SCT and uses WS-Trust to propagate it to the communicating parties.
- The SCT is created through negotiation between communicating parties: The security context initiator uses WS-Trust requests / responses to create an SCT.

WS-Policy

The Web Services Policy Framework (WS-Policy) is a specification jointly developed by independent vendors.

WS-Policy is used in conjunction with WS-Security to express the capabilities and requirements of entities used in Web services environments.

A policy is expressed as policy assertions. A policy assertion represents a capability or a requirement. Policy assertions are defined in a companion specification (Web Services Policy Assertions Language or WS-PolicyAssertions).

For example, a policy can be defined to exclusively select (<wsp:ExactlyOne>) an authentication mechanism, as shown in Listing 7 (borrowed from the WS-Policy specification).

Listing 8: WS-Policy Example

```
01 <wsp: Policy xmlns: wsse="- - -" xmlns:wssx="- - -">
02   <wsp:ExactlyOne>
03     <wsse:SecurityToken wsp:Usage="wsp:Required">
04       <wsse:TokenType>wsse:Kerberosv5TGT</wsse:TokenType>
05     </wsse:SecurityToken>
06     <wsse:SecurityToken wsp:Usage="wsp:Required">
07       <wsse:TokenType>wsse:X509v3</wsse:TokenType>
08     </wsse:SecurityToken>
09   </wsp:ExactlyOne>
10   <wssx:Audit wsp:Usage="wsp:Observed"/>
11 </wsp:Policy>
```

WS-Policy expressions are associated with various Web services components using the Web Services Policy Attachment specification (WS-PolicyAttachment).

WS-Federation

Web Services Federation Language (WS-Federation) is a specification jointly developed by independent vendors.

WS-Federation provides support for secure propagation of identity, attribute, authentication, and authorization information.

By relying on the models defined in WS-Security, WS-Trust, and WS-Policy, WS-Federation enables brokering of trust and security token exchange, support for privacy by hiding identity and attribute information, and federated sign-out.

The following diagram (Figure 10) shows how the WS-* specifications relate to each other, SOAP, XML Signature, and XML Encryption.

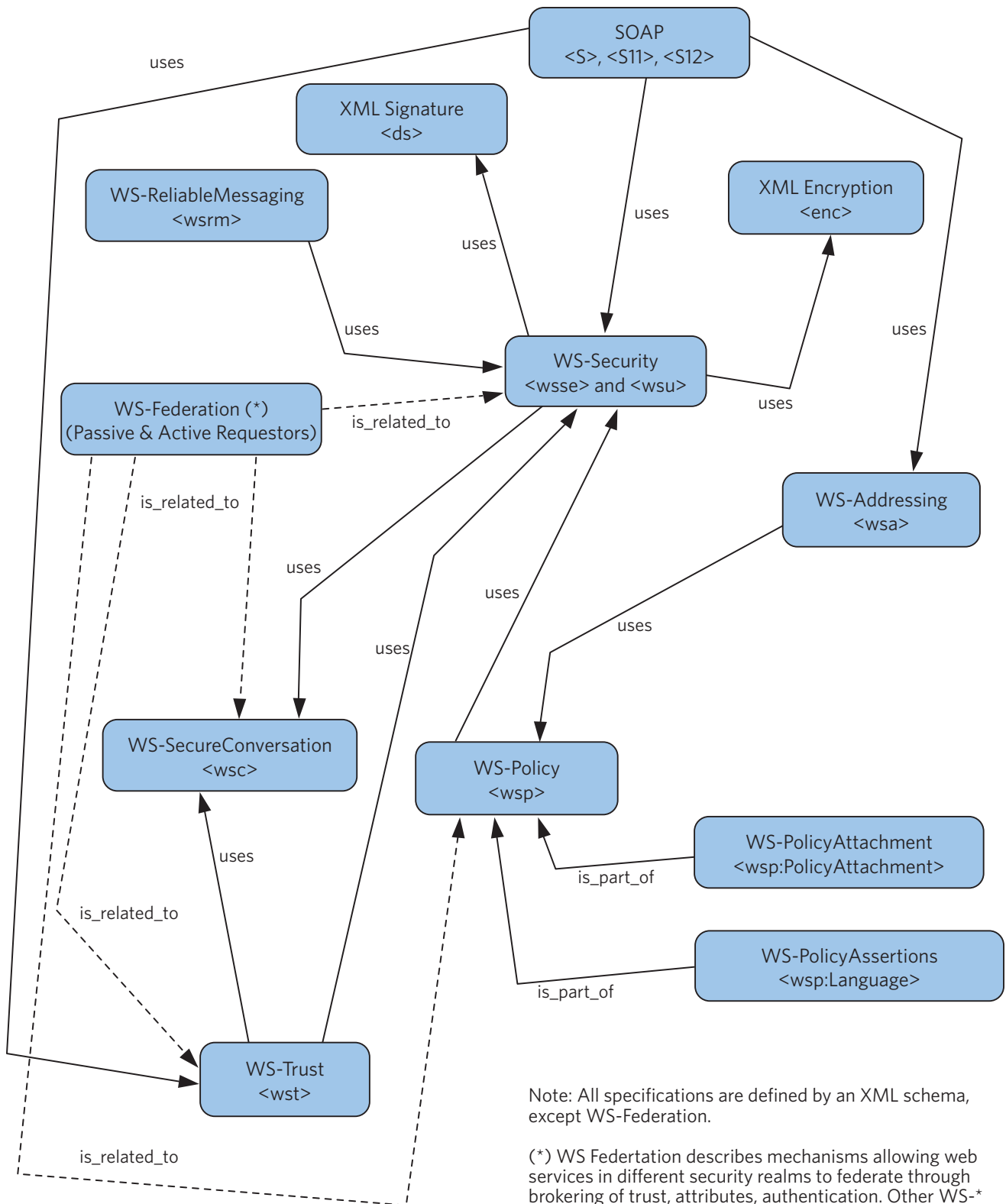


Figure 10: WS-* Specifications - Relationship Diagram

Conclusion

As identity and access management becomes the integration point for user and application administration across heterogeneous environments and architectures, emerging standards and industry initiatives play a strong role in the development of the identity and access management market, including intra- and inter-enterprise identity federation.

Today, SAML, Liberty Alliance (ID-FF), and WS-Security provide complete identity federation solutions for both web applications and Web services, successfully deployed by multiple companies worldwide.

Over the next few years, the various Web Services specifications (WS-*) currently drafted by independent vendors (including CA) will provide solutions for complex, large-scale identity federation deployments in service-oriented architectures (SOA).

Please refer to **Universal Federation Architecture: eTrust SiteMinder and eTrust TransactionMinder Solutions**, a White Paper from CA which describes how some of the standards presented in this document are implemented in eTrust SiteMinder and eTrust TransactionMinder, the company's platforms for web applications and Web services security and access management.

Glossary

This glossary is designed as a quick reference to the words or concepts not explicitly defined in the text of this document. Words in italics are defined in their own entries.

Access: The ability to create, read, modify, or delete a computer resource such as a text document, a multimedia object (e.g., a picture or a video), or a business application.

Access Control: Protection of *resources* against unauthorized access.

Access Control List (ACL): A list of users or groups of users whose *identities* are associated with specific *access rights*.

Access Rights: Authorized interactions an *entity* (e.g., a user) can have with a *resource* (e.g., Read, Delete, Modify, Add, etc.).

Assertion: A set of *claims*.

Attribute: The property or characteristic of an *entity* used for *authorization* purposes.

Authentication: Verifying an *entity's identity* based on submitted credentials. There are three authentication factors: Something you have, e.g., credentials issued by a trusted authority such as a passport or a *certificate*; Something you know, e.g., a shared secret such as a password; Something you are, e.g., biometric information.

Authorization: Granting access to specific *resources* based on an authenticated *entity's entitlements*.

Base64: Binary information may need to be included in textual form in email or XML documents. If you want to insert binary information in an XML document, you need to first encode it in text. *Base64* is an algorithm that encodes binary data using 64 ASCII characters (hence the name). Every binary input stream is broken down into six-bit segments, each segment is mapped to an ASCII character, and the "=" (equal) sign (the 64th ASCII character) is used for padding a segment that has fewer than 6 bits. Note that *Base64* information is not human readable, but it's not encrypted either, it's just a string of characters representing binary input.

Certificate (or X.509 Public-Key Certificate): A data structure digitally signed by the authority that issues it. A *certificate* binds an *entity* to a *key* (a *certificate* is used to send the *public key* to a receiving party). A *certificate* includes standard fields such as certificate ID, issuer's *DN*, validity period, owner's *DN*, owner's *public key*, etc.

Certificate Authority (CA): An *entity* with the authority to issue *public keys*, along with *certificates* that bind owners and *keys*.

Cipher: *Encryption* and *decryption* algorithms.

Ciphertext: Encrypted data (the opposite is plaintext).

Claim: A statement made by an *entity* (user *identity* and *attributes* are typical *claims*).

Confidentiality: see *Privacy*.

Credentials: Data used to establish an *entity's identity*. Typically, *credentials* can be a username / password or a *certificate* (see also *Authentication*).

Data Integrity: Making sure that data can't be altered when in transit between sending and receiving parties. *Data integrity* is ensured by *digital Signatures*.

Decryption: The process of transforming *ciphertext* back into its original clear text using a cryptographic *key*.

Digest (or Message Digest): The result of applying a one-way algorithm (e.g., MD5, SHA-1, etc.) to an input stream in order to compress it (see also *MAC*).

Digital Signature: Verifying that a message came from the intended *entity* (authentication) and that the message was not altered during transit (data integrity). Digital signatures use public key encryption (in reverse) as follows: The sending party generates a digest of the data to be sent, encrypts the digest with its private key, and then sends the result with its public key and the (plaintext) data. The receiving party uses the sending party's public key to decrypt the signature and performs the same calculation (hash function) on the data (see also XML Signature Specification).

Distinguished Name (DN): A string of characters with several fields used to describe an *identity*. Typically, a *DN* includes a common name (CN), an organizational unit (OU), an organization (O), a location (L), a state or province (S), a country (C) and other optional fields.

Example of a DN: CN=James Dean, OU=Engineering, O=Computer Associates, L=Waltham, S=Massachusetts, C=US

Domain Name: Plain text *identity* of an organization (or *entity*) on the Internet (for example, ca.com).

Domain Name Service (DNS): A program that translates the domain name into an Internet Protocol (IP) address (for example 172.26.6.32).

Encryption: The process of creating *ciphertext* from clear text using a cryptographic *key* (see also *XML Encryption Specification*). *Encryption* provides *confidentiality* (or *privacy*).

Entitlement: *Access right* defined by one or several *attributes*. Synonym: *Privilege*.

Entity: Any object that interacts with other *entities*, such as a person (e.g., an end-user) or a system (e.g., an application program or a hardware device). Synonyms: *Principal* (human *entity*), *Subject* (human or system *entity*).

Federation: A network environment where partners can interoperate seamlessly by sharing *identities*. Synonym: *Single sign-on* (SSO).

Hash: See *Digest*.

Hash Function: A one-way transformation that maps a variable-size input (e.g., a *key*, a password, etc.) to a fixed-size string (the hash value, typically between 64 and 256 bits).

Identity: Unique and meaningful information associated with an *entity*. An *identity* can be described simply by a user name and a password, or more universally by an *X.500 Distinguished Name (DN)*. An *identity* can also include *attributes*, such as an Internet Protocol (IP) address or a *role*.

Identity Federation: Ability to manage and deliver different representations of an *entity's identity* to different partners.

Integrity: See *Data Integrity*.

Issuing Party (or Issuing Entity): The party (or *entity*) that authenticates a user or a service (also see *Relying Party*).

Key (or Cryptographic Key): A mathematical function used to encrypt or decrypt data. In a symmetric *cipher*, the *key* to encrypt and decrypt data is the same. In an asymmetric *cipher*, a *public key* and a *private key* pair is used for confidentiality and data signing. The owner of the *key* pair has to make the *public key* known to other parties. Cryptographic assurances for confidentiality and digital signing work differently. Confidentiality is supported by encrypting with a *public key* and decrypting with a matching *private key*. Conversely, digital signing is supported by encrypting with a *private key* and decrypting with a matching *public key*.

Log-in: Presenting credentials to an *authentication* system in order to start a *session* with that system. Synonyms: Log-on, Sign-on.

Log-out: Terminating a current session. Synonym: Sign-out.

Message Authentication Code (MAC): A secure message *digest*. A MAC requires a secret *key* that is shared by the sending and the receiving parties.

Non-Repudiation: Making sure that a document comes from the party that actually issued it. Support for *non-repudiation* is provided by a *digital signature* binding the issuer's *identity* to the document.

Policy (or Security Policy): A set of rules designed to control access to a *resource*.

Policy Domain: A logical set of *resources* grouped together for administrative purposes, e.g., a Marketing policy domain, a Support policy domain, etc.

Policy Expression: The mapping of an *entity's identity* and/or *attributes* to authorized actions on a protected *resource*.

Policy Decision Point (PDP): A logical entity that makes authorization decisions on its own behalf or on behalf of other *entities* that requested it. For example, the company's Policy Server is the physical implementation of a PDP.

Policy Enforcement Point (PEP): A logical entity that requests and enforces authorization decisions. For example, SiteMinder and TransactionMinder agents are physical implementations of PEPs.

Principal: See *Entity*.

Privacy: The ability to keep information secret. *Privacy* involves a message, for example a Web service request, an email, or a business document, as well as the identity of the sending and receiving parties. *Privacy* can be achieved by encrypting the content of a message and obfuscating the sending and receiving parties' identities.

Private Key: In an asymmetric *cipher* model, a *private key* is used to decrypt *ciphertext*. Also, a *private key* is used for encryption with *digital signatures* (see *Key, Public Key, Digital Signature*).

Public Key: In an asymmetric *cipher* model, the receiving party's *public key* (possibly stored in a public repository) is used to encrypt plaintext, and the receiving party's matching *private key* is used to decrypt the *ciphertext*. In other words, when a party wants to send sensitive data to another party, all the sending party needs is the *public key* of the receiving party. *Public-key certificates* are used to guarantee the integrity of *public keys*. Also, the *public key* is used for verifying *digital signatures* (see *Key, Private Key, Digital Signature*).

Realm: A collection of resources grouped together according to security requirements. *Realms* are generally associated with *policy domains* (a *policy domain* can contain several *realms*). Also, a *realm*, as defined in many standards specifications, represents a single unit of trust between the source and the destination of a request.

Role: The active part of an *entity*, e.g., an administrator *role* or an agent *role*.

Relying Party (or Relying Entity): A party (or *entity*) that participates in a *federation* and accepts to take an action based on information provided by an *Issuing Party*.

Secure Socket Layer (SSL): Transport-level data-communication protocol providing *authentication* (the communication is established between two trusted parties using *certificates*), *confidentiality* (the data exchanged is encrypted), and message *integrity* (the data is checked for possible corruption).

Security Token: A collection of *claims* (e.g., a SAML *assertion*, a Kerberos ticket, a *certificate*, etc.).

Session: The interactions between various *entities*.

Single Sign-On (SSO): The ability to be authenticated at one site and be able to visit other sites without repeated *log-in*.

Subject: See *Entity*.

Trust: A relationship between *entities* whereby one *entity* relies on the other for delivering a service securely. *Trust* is predicated on *privacy* and security *policies* between interacting parties.

Uniform Resource Identifier (URI): A string of characters used to identify a *resource* on the Web. A *URI* can be expressed in an abstract way as a *URN* (persistent) or as a *URL* (changeable).

Uniform Resource Name (URN): The persistent identifier (or item name) of a *resource* (the *URN* for a particular *resource* cannot be changed). A *URN* can be expressed as a *URL*.

Uniform Resource Locator (URL): The location of a *resource* on the web and the protocol used to access that *resource*. A *URL* can be changed.

XML Signature Specification: The binding of the sender's *identity* (or "signing *entity*") to an XML document. The document is signed using the sender's *private key*, the signature is verified using the sender's *public key*. A digital signature ensures non-repudiation of the signing *entity* (*authentication*) and proves that messages have not been altered since they were signed (see also *Digital Signature*).

XML Encryption Specification: Defining how digital content is encrypted and decrypted, how the *encryption key* information is passed to a recipient, and how encrypted data is identified to facilitate *decryption*.

Technical References

- A roadmap for message delivery in Web services: <http://www-106.ibm.com/developerworks/webservices/library/ws-rmdev/>
- Kerberos: <http://web.mit.edu/kerberos/www/>
- Security Assertion Markup Language (SAML): http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security
- The Liberty Alliance Project: <http://www.projectliberty.org/>
- Web Services Federation Language (WS-Federation): <http://www-106.ibm.com/developerworks/library/ws-fed/>
- Web Services Policy Assertions Language (WS-PolicyAssertions): <http://www-106.ibm.com/developerworks/library/ws-polas/>
- Web Services Policy Attachment (WS-PolicyAttachment): <http://www-106.ibm.com/developerworks/library/ws-polatt/>
- Web Services Policy Framework (WS-Policy): <http://www-106.ibm.com/developerworks/library/ws-polfram/>
- Web Services Secure Conversation (WS-SecureConversation): <http://www-106.ibm.com/developerworks/library/ws-secon/>
- Web Services Security (WS-Security): http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
- Web Services Security Policy (WS-SecurityPolicy): <http://www-106.ibm.com/developerworks/library/ws-secpol/>
- Web Services Security Trust (WS-Trust): <http://www-106.ibm.com/developerworks/library/ws-trust/>
- XML Encryption: <http://www.w3.org/TR/xmlenc-core/>
- XML Signature: <http://www.w3.org/TR/xmldsig-core/>

