Security Assertion Markup Language (SAML) V2.0 Technical Overview

Working Draft 10, 9 October 2006

Document identifier:

sstc-saml-tech-overview-2.0-draft-10

Location:

http://www.oasis-open.org/committees/documents.php?wg_abbrev=security

Editors:

Nick Ragouzis, Enosis Group LLC
John Hughes, PA Consulting
Rob Philpott, RSA Security
Eve Maler, Sun Microsystems

Contributors:

Hal Lockhart, BEA
Thomas Wisniewski, Entrust
Scott Cantor, Internet2
Prateek Mishra, Oracle

Abstract:

The Security Assertion Markup Language (SAML) standard defines a framework for exchanging security information between online business partners. It was developed by the Security Services Technical Committee (SSTC) of the standards organization OASIS (the Organization for the Advancement of Structured Information Standards). This document provides a technical description of SAML V2.0.

Status:

This draft is a non-normative document that is intended to be approved as a Committee Draft by the SSTC. This document is not currently on an OASIS Standard track. Readers should refer to the normative specification suite for precise information concerning SAML V2.0.

Committee members should send comments on this specification to the security-services@lists.oasis-open.org list. Others should submit them by filling in the form at http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Security Services TC web page (http://www.oasis-open.org/committees/security/).

Table of Contents

109 # Table of Figures

110

1 Introduction

The OASIS Security Assertion Markup Language (SAML) standard defines an XML-based framework for describing and exchanging security information between on-line business partners. This security information is expressed in the form of portable SAML assertions that applications working across security domain boundaries can trust. The OASIS SAML standard defines precise syntax and rules for requesting, creating, communicating, and using these SAML assertions.

The OASIS Security Services Technical Committee (SSTC) develops and maintains the SAML standard. The SSTC has produced this technical overview to assist those wanting to know more about SAML by explaining the business use cases it addresses, the high-level technical components that make up a SAML deployment, details of message exchanges for common use cases, and where to go for additional information.

1.1 Drivers of SAML Adoption

Why is SAML needed for exchanging security information? There are several drivers behind the adoption of the SAML standard, including:

- ***Single Sign-On:*** Over the years, various products have been marketed with the claim of providing support for web-based SSO. These products have typically relied on browser cookies to maintain user authentication state information so that re-authentication is not required each time the web user accesses the system. However, since browser cookies are never transmitted between DNS domains, the authentication state information in the cookies from one domain is never available to another domain. Therefore, these products have typically supported multi-domain SSO (MDSSO) through the use of proprietary mechanisms to pass the authentication state information between the domains. While the use of a single vendor's product may sometimes be viable within a single enterprise, business partners usually have heterogeneous environments that make the use of proprietary protocols impractical for MDSSO. SAML solves the MDSSO problem by providing a standard vendor-independent grammar and protocol for transferring information about a user from one web server to another independent of the server DNS domains.

- ***Federated identity:*** When online services wish to establish a collaborative application environment for their mutual users, not only must the systems be able to understand the protocol syntax and semantics involved in the exchange of information; they must also have a common understanding of who the user is that is referred to in the exchange. Users often have individual local user identities within the security domains of each partner with which they interact. Identity federation provides a means for these partner services to agree on and establish a common, shared name identifier to refer to the user in order to share information about the user across the organizational boundaries. The user is said to have a ***federated identity*** when partners have established such an agreement on how to refer to the user. From an administrative perspective, this type of sharing can help reduce identity management costs as multiple services do not need to independently collect and maintain identity-related data (e.g. passwords, identity attributes). In addition, administrators of these services usually do not have to manually establish and maintain the shared identifiers; rather control for this can reside with the user.

- ***Web services and other industry standards:*** SAML allows for its security assertion format to be used outside of a "native" SAML-based protocol context. This modularity has proved useful to other industry efforts addressing authorization services (IETF, OASIS), identity frameworks, web services (OASIS, Liberty Alliance), etc. The OASIS WS-Security Technical Committee has defined a ***profile*** for how to use SAML's rich assertion constructs within a WS-Security ***security token*** that can be used, for example, to secure web service SOAP message exchanges. In particular, the advantage offered by the use of a SAML assertion is that it provides a standards-based approach to the exchange of information, including attributes, that are not easily conveyed using other WS-Security token formats.

1.2 Documentation Roadmap

The OASIS SSTC has produced numerous documents related to SAML V2.0. This includes documents that make up the official OASIS standard itself, outreach material intended to help the public better understand SAML V2.0, and several extensions to SAML to facilitate its use in specific environments or to integrate it with other technologies.

The documents that define and support the SAML V2.0 OASIS Standard are shown in Figure 1. The lighter-colored boxes represent non-normative information.

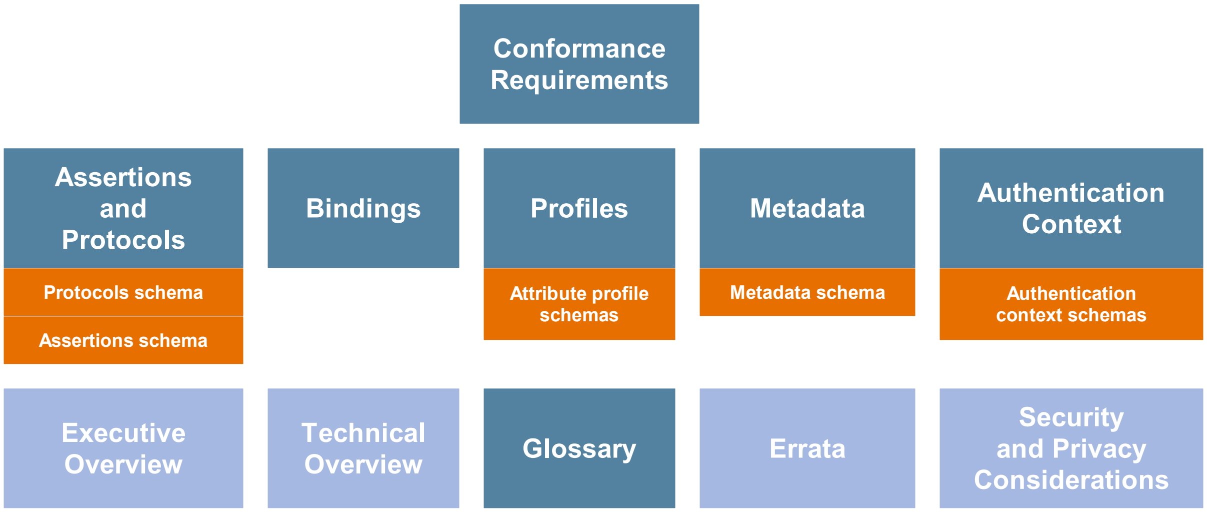


Figure 1: SAML V2.0 Document Set

- **Conformance Requirements [SAMLConform]** documents the technical requirements for SAML conformance, a status that software vendors typically care about because it is one measure of cross-product compatibility. If you need to make a formal reference to SAML V2.0 from another document, you simply need to point to this one.
- **Assertions and Protocol [SAMLCore]** defines the syntax and semantics for creating XML-encoded assertions to describe authentication, attribute, and authorization information, and for the protocol messages to carry this information between systems. It has associated schemas, one for assertions and one for protocols.
- **Bindings [SAMLBind]** defines how SAML assertions and request-response protocol messages can be exchanged between systems using common underlying communication protocols and frameworks.
- **Profiles [SAMLProf]** defines specific sets of rules for using and restricting SAML's rich and flexible syntax for conveying security information to solve specific business problems (for example, to perform a web SSO exchange). It has several associated small schemas covering syntax aspects of attribute profiles.
- **Metadata [SAMLMeta]** defines how a SAML entity can describe its configuration data (e.g. service endpoint URLs, key material for verifying signatures) in a standard way for consumption by partner entities. It has an associated schema.
- **Authentication Context [SAMLAuthnCxt]** defines a syntax for describing authentication context declarations which describe various authentication mechanisms. It has an associated set of schemas.
- **Executive Overview [SAMLExecOvr]** provides a brief executive-level overview of SAML and its primary benefits. This is a non-normative document.

- **Technical Overview** is the document you are reading.
- **Glossary [SAMLGloss]** normatively defines terms used throughout the SAML specifications. Where possible, terms are aligned with those defined in other security glossaries.
- **Errata [SAMLErrata]** clarifies interpretation of the SAML V2.0 standard where information in the final published version was conflicting or unclear. Although the advice offered in this document is non-normative, it is useful as a guide to the likely interpretations used by implementors of SAML-conforming software, and is likely to be incorporated in any future revision to the standard. This document is updated on an ongoing basis.
- **Security and Privacy Considerations [SAMLSec]** describes and analyzes the security and privacy properties of SAML.

Following the release of the SAML V2.0 OASIS Standard, the OASIS SSTC has continued work on several enhancements. As of this writing, the documents for the following enhancements have been approved as OASIS Committee Draft specifications and are available from the OASIS SSTC web site:

- **SAML Metadata Extension for Query Requesters [SAMLMDExtQ]**. Defines role descriptor types that describe a standalone SAML V1.x or V2.0 query requester for each of the three predefined query types.
- **SAML Attribute Sharing Profile for X.509 Authentication-Based Systems [SAMLX509Attr]**. Describes a SAML profile enabling an attribute requester entity to make SAML attribute queries about users that have authenticated at the requester entity using an X.509 client certificate.
- **SAML V1.x Metadata [SAMLMDV1x]**. Describes the use of the SAML V2.0 metadata constructs to describe SAML entities that support the SAML V1.x OASIS Standard.
- **SAML XPath Attribute Profile [SAMLXPathAttr]**. Profiles the use of SAML attributes for using XPath URI's as attribute names.
- **SAML Protocol Extension for Third-Party Requests [SAMLProt3P]**. Defines an extension to the SAML protocol to facilitate requests made by entities other than the intended response recipient.

2 High-Level SAML Use Cases

Prior to examining details of the SAML standard, it's useful to describe some of the high-level use cases it addresses. More detailed use cases are described later in this document along with specific SAML profiles.

2.1 SAML Participants

Who are the participants involved in a SAML interaction? At a minimum, SAML exchanges take place between system entities referred to as a SAML *asserting party* and a SAML *relying party*. In many SAML use cases, a user, perhaps running a web browser or executing a SAML-enabled application, is also a participant, and may even be the asserting party.

An asserting party is a system entity that makes SAML assertions. It is also sometimes called a *SAML authority*. A relying party is a system entity that uses assertions it has received. When a SAML asserting or relying party makes a direct request to another SAML entity, the party making the request is called a *SAML requester*, and the other party is referred to as a *SAML responder*. A replying party's willingness to rely on information from an asserting party depends on the existence of a trust relationship with the asserting party.

SAML system entities can operate in a variety of SAML *roles* which define the SAML services and protocol messages they will use and the types of assertions they will generate or consume. For example, to support Multi-Domain Single Sign-On (MDSSO, or often just SSO), SAML defines the roles called *identity provider (IdP)* and *service provider (SP)*. Another example is the *attribute authority* role where a SAML entity produces assertions in response to identity attribute queries from an entity acting as an *attribute requester*.

At the heart of most SAML assertions is a *subject* (a principal – an entity that can be authenticated – within the context of a particular security domain) about which something is being asserted. The subject could be a human but could also be some other kind of entity, such as a company or a computer. The terms subject and principal tend to be used interchangeably in this document.

A typical assertion from an identity provider might convey information such as “This user is John Doe, he has an email address of john.doe@example.com, and he was authenticated into this system using a password mechanism.” A service provider could choose to use this information, depending on its access policies, to grant John Doe web SSO access to local resources.

2.2 Web Single Sign-On Use Case

Multi-domain web single sign-on is the most important use case for which SAML is used. In this use case, a user has a login session (that is, a *security context)* on a web site (*AirlineInc.com*) and is accessing resources on that site. At some point, either explicitly or transparently, he is directed over to a partner's web site (*CarRentalInc.com*). In this case, we assume that a federated identity for the user has been previously established between *AirlineInc.com* and *CarRentalInc.com* based on a business agreement between them. The identity provider site (*AirlineInc.com*) asserts to the service provider site (*CarRentalInc.com*) that the user is known (by referring to the user by their federated identity), has authenticated to it, and has certain identity attributes (e.g. has a “Gold membership”). Since *CarRentalInc.com* trusts *AirlineInc.com*, it trusts that the user is valid and properly authenticated and thus creates a local session for the user. This use case is shown in Figure 2, which illustrates the fact that the user is not required to re-authenticate when directed over to the *CarRentalInc.com* site.

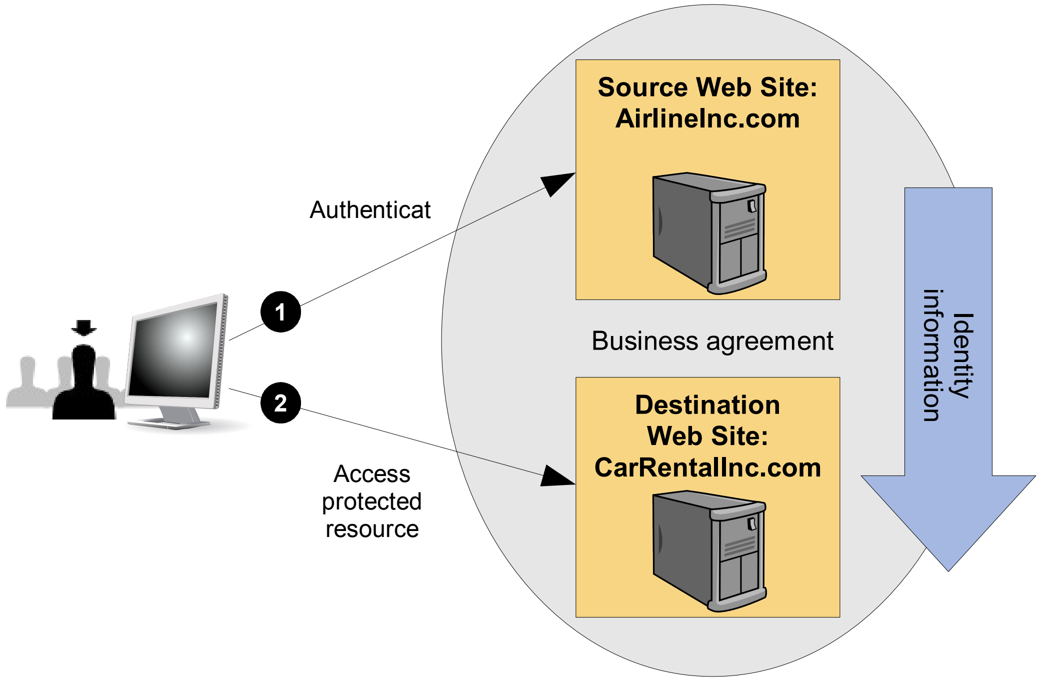


Figure 2: General Single Sign-On Use Case

This high-level description indicated that the user had first authenticated at the IdP before accessing a protected resource at the SP. This scenario is commonly referred to as an IdP-initiated web SSO scenario. While IdP-initiated SSO is useful in certain cases, a more common scenario starts with a user visiting an SP site through a browser bookmark, possibly first accessing resources that require no special authentication or authorization. In a SAML-enabled deployment, when they subsequently attempt to access a protected resource at the SP, the SP will send the user to the IdP with an authentication request in order to have the user log in. Thus this scenario is referred to as SP-initiated web SSO. Once logged in, the IdP can produce an assertion that can be used by the SP to validate the user's access rights to the protected resource. SAML V2.0 supports both the IdP-initiated and SP-initiated flows.

SAML supports numerous variations on these two primary flows that deal with requirements for using various types and strengths of user authentication methods, alternative formats for expressing federated identities, use of different bindings for transporting the protocol messages, inclusion of identity attributes, etc. Many of these options are looked at in more detail in later sections of this document.

2.3 Identity Federation Use Case

As mentioned earlier, a user's identity is said to be federated between a set of providers when there is an agreement between the providers on a set of identifiers and/or identity attributes by which the sites will refer to the user.

There are many questions that must be considered when business partners decide to use federated identities to share security and identity information about users. For example:

- Do the users have existing local identities at the sites that must be linked together through the federated identifiers?
- Will the establishment and termination of federated identifiers for the users be done dynamically or will the sites use pre-established federated identifiers?
- Do users need to explicitly consent to establishment of the federated identity?
- Do identity attributes about the users need to be exchanged?
- Should the identity federation rely on transient identifiers that are destroyed at the end of the user session?

- Is the privacy of information to be exchanged of high concern such that the information should be encrypted?

Previous versions of the SAML standard relied on out-of-band agreement on the types of identifiers that would be used to represent a federated identity between partners (e.g. the use of X.509 subject names). While it supported the use of federated identities, it provided no means to directly establish the identifiers for those identities using SAML message exchanges. SAML V2.0 introduced two features to enhance its federated identity capabilities. First, new constructs and messages were added to support the dynamic establishment and management of federated name identifiers. Second, two new types of name identifiers were introduced with privacy-preserving characteristics.

In some cases, exchanges of identity-related federation information may take place outside of the SAML V2.0 message exchanges. For example, providers may choose to share information about registered users via batch or off-line "identity feeds" that are driven by data sources (for example, human resources databases) at the identity provider and then propagated to service providers. Subsequently, the user's federated identity may be used in a SAML assertion and propagated between providers to implement single sign-on or to exchange identity attributes about the user. Alternatively, identity federation may be achieved purely by a business agreement that states that an identity provider will refer to a user based on certain attribute names and values, with no additional flows required for maintaining and updating user information between providers.

The high-level identity federation use case described here demonstrates how SAML can use the new features to dynamically establish a federated identity for a user during a web SSO exchange. Most identity management systems maintain *local identities* for users. These local identities might be represented by the user's local login account or some other locally identifiable user profile. These local identities must be linked to the federated identity that will be used to represent the user when the provider interacts with a parter. The process of associating a federated identifier with the local identity at a partner (or partners) where the federated identity will be used is often called *account linking*.

This use case, shown in Figure 3, demonstrates how, during web SSO, the sites can dynamically establish the federated name identifiers used in the account linking process. One identity provider, *AirlineInc.com,* and two service providers exist in this example; *CarRentalInc.com* for car rentals and *HotelBooking.com* for hotel bookings. The example assumes a user is registered on all three provider sites (i.e. they have pre-existing local login accounts), but the local accounts all have different account identifiers. At *AirlineInc.com*, user John is registered as **johndoe**, on *CarRentalInc.com* his account is **jdoe**, and on *HotelBooking.com* it is **johnd**. The sites have established an agreement to use ***persistent*** SAML privacy-preserving pseudonyms for the user's federated name identifiers. John has not previously federated his identities between these sites.

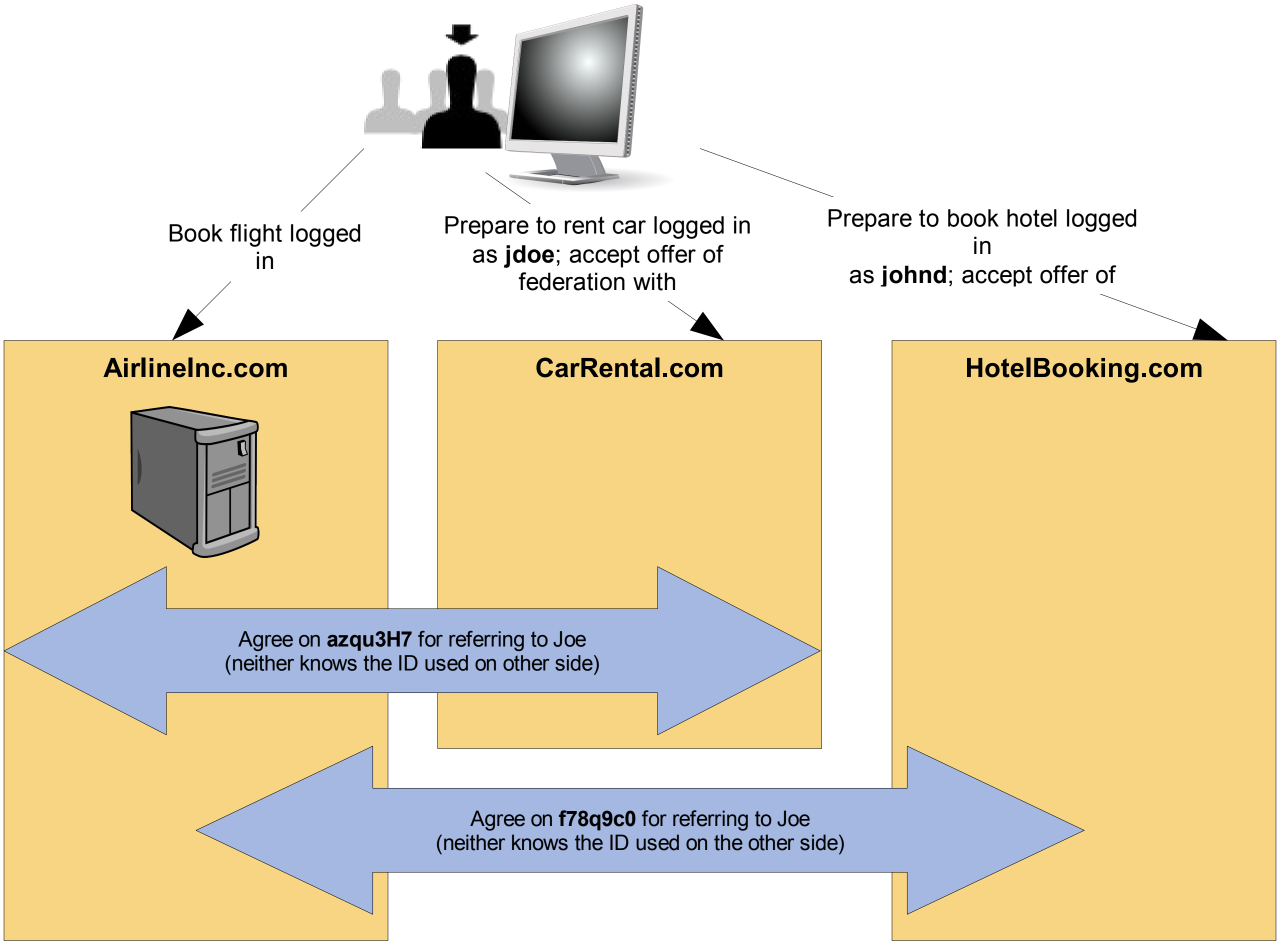

Book flight logged in
Prepare to rent car logged in as **jdoe**; accept offer of federation with
Prepare to book hotel logged in as **johnd**; accept offer of
AirlineInc.com
CarRental.com
HotelBooking.com
Agree on **azqu3H7** for referring to Joe (neither knows the ID used on other side)
Agree on **f78q9c0** for referring to Joe (neither knows the ID used on the other side)

Figure 3: General Identity Federation Use Case

The processing sequence is as follows:

1. John books a flight at *AirlineInc.com* using his **johndoe** user account.
2. John then uses a browser bookmark or clicks on a link to visit *CarRentalInc.com* to reserve a car. *CarRentalInc.com* sees that the browser user is not logged in locally but that he has previously visited their IdP partner site *AirlineInc.com* (optionally using the new IdP discovery feature of SAML V2.0). So *CarRentalInc.com* asks John if he would like to consent to federate a local identity with *AirlineInc.com*.
3. John consents to the federation and his browser is redirected back to *AirlineInc.com* where the site creates a new pseudonym, **azqu3H7** for John's use when he visits *CarRentalInc.com.* The pseudonym is linked to his **johndoe** account.
4. John is then redirected back to *CarRentalInc.com* with a SAML assertion indicating that the user represented by the federated persistent identifier **azqu3H7** is logged in at the IdP. Since this is the first time that *CarRentalInc.com* has seen this identifier, it does not know which local user account to which it applies.
5. Thus, John must log in at *CarRentalInc.com* using his **jdoe** account. Then *CarRentalInc.com* attaches the identity **azqu3H7** to the local **jdoe** account for future use with the IdP *AirlineInc.com*. The user accounts at the IdP and this SP are now *linked* using the federated name identifier **azqu3H7**.
6. After reserving a car, John selects a browser bookmark or clicks on a link to visit *HotelBooking.com* in order to book a hotel room.
7. The process is repeated with the IdP *AirlineInc.com*, creating a new pseudonym, **f78q9C0**, for IdP

user **johndoe** that will be used when visiting *HotelBooking.com*.

8. John is redirected back to the *HotelBooking.com* SP with a new SAML assertion. The SP requires John to log into his local **johnd** user account and adds the pseudonym as the federated name identifier for future use with the IdP *AirlineInc.com*. The user accounts at the IdP and this SP are now *linked* using the federated name identifier **f78q9C0**.

In the future, whenever John needs to books a flight, car, and hotel, he will only need to log in once to *AirlineInc.com* before visiting *CarRentalInc.com* and *HotelBooking.com*. The *AirlineInc.com* IdP will identify John as **azqu3H7** to *CarRentalInc.com* and as **f78q9C0** to *HotelBooking.com*. Each SP will locate John's local user account through the linked persistent pseudonyms and allow John to conduct business through the SSO exchange.

3 SAML Architecture

This section provides a brief description of the key SAML concepts and the components defined in the standard.

3.1 Basic Concepts

SAML consists of building-block components that, when put together, allow a number of use cases to be supported. The components primarily permit transfer of identity, authentication, attribute, and authorization information between autonomous organizations that have an established trust relationship. The ***core*** SAML specification defines the structure and content of both *assertions* and *protocol messages* used to transfer this information.

SAML assertions carry statements about a principal that an asserting party claims to be true. The valid structure and contents of an assertion are defined by the SAML assertion XML schema. Assertions are usually created by an asserting party based on a request of some sort from a relying party, although under certain circumstances, the assertions can be delivered to a relying party in an unsolicited manner. SAML protocol messages are used to make the SAML-defined requests and return appropriate responses. The structure and contents of these messages are defined by the SAML-defined protocol XML schema.

The means by which lower-level communication or messaging protocols (such as HTTP or SOAP) are used to transport SAML protocol messages between participants is defined by the SAML *bindings*.

Next, SAML *profiles* are defined to satisfy a particular business use case, for example the Web Browser SSO profile. Profiles typically define constraints on the contents of SAML assertions, protocols, and bindings in order to solve the business use case in an interoperable fashion. There are also Attribute Profiles, which do not refer to any protocol messages and bindings, that define how to exchange attribute information using assertions in ways that align with a number of common usage environments (e.g. X.500/LDAP directories, DCE).

Figure 4 illustrates the relationship between these basic SAML concepts.

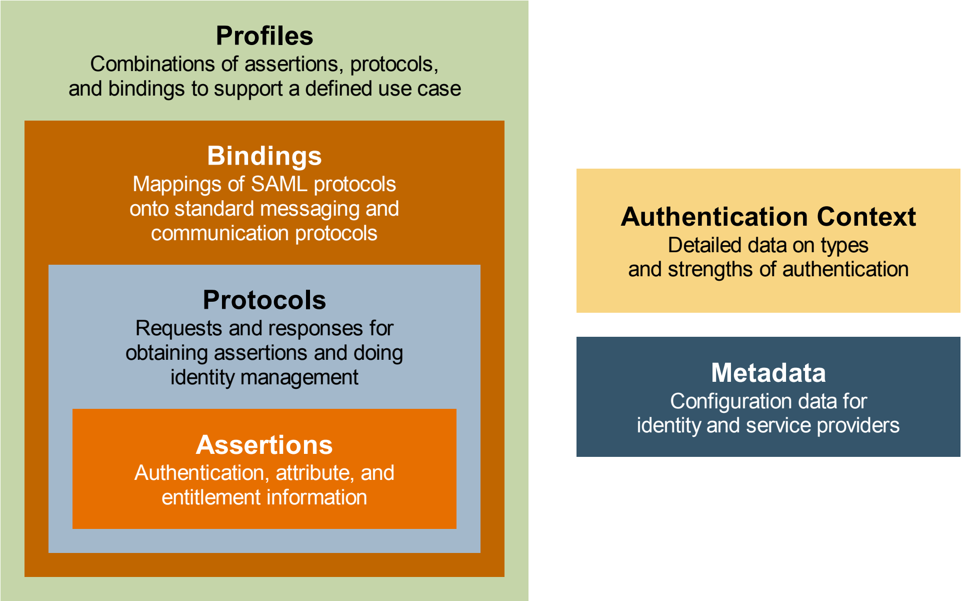

Profiles
Combinations of assertions, protocols, and bindings to support a defined use case
Bindings
Mappings of SAML protocols onto standard messaging and communication protocols
Protocols
Requests and responses for obtaining assertions and doing identity management
Assertions
Authentication, attribute, and entitlement information
Authentication Context
Detailed data on types and strengths of authentication
Metadata
Configuration data for identity and service providers

Figure 4: Basic SAML Concepts

Two other SAML concepts are useful for building and deploying a SAML environment:

- *Metadata* defines a way to express and share configuration information between SAML parties. For instance, an entity's supported SAML bindings, operational roles (IDP, SP, etc), identifier information, supporting identity attributes, and key information for encryption and signing can be expressed using SAML metadata XML documents. SAML Metadata is defined by its own XML schema.

- In a number of situations, a service provider may need to have detailed information regarding the type and strength of authentication that a user employed when they authenticated at an identity provider. A SAML *authentication context* is used in (or referred to from) an assertion's authentication statement to carry this information. An SP can also include an authentication context in a request to an IdP to request that the user be authenticated using a specific set of authentication requirements, such as a multi-factor authentication. There is a general XML schema that defines the mechanisms for creating authentication context declarations and a set of SAML-defined Authentication Context Classes, each with their own XML schema, that describe commonly used methods of authentication.

This document does not go into further detail about Metadata and Authentication Context; for more information, see the specifications that focus on them ([SAMLMeta] and [SAMLAuthnCxt], respectively).

It should be noted that the story of SAML need not end with its published set of assertions, protocols, bindings, and profiles. It is designed to be highly flexible, and thus it comes with extensibility points in its XML schemas, as well as guidelines for custom-designing new bindings and profiles in such a way as to ensure maximum interoperability.

3.2 SAML Components

This section takes a more detailed look at each of the components that represent the assertion, protocol, binding, and profile concepts in a SAML environment.

- **Assertions:** SAML allows for one party to assert security information in the form of ***statements*** about a ***subject***. For instance, a SAML assertion could state that the subject is named “John Doe”, has an email address of john.doe@example.com, and is a member of the “engineering” group. An assertion contains some basic required and optional information that applies all assertions, and usually contains a *subject* of the assertion, *conditions* used to validate the assertion, and assertion statements*.* SAML defines three kinds of statements that can be carried within an assertion:
  - **Authentication statements:** These are created by the party that successfully authenticated a user. At a minimum, they describe the particular means used to authenticate the user and the specific time at which the authentication took place.
  - **Attribute statements:** These contain specific identifying attributes about the subject (for example, that user “John Doe” has “Gold” card status).
  - **Authorization decision statements:** These define something that the subject is entitled to do (for example, whether “John Doe” is permitted to buy a specified item).
- **Protocols:** SAML defines a number of generalized request/response protocols:
  - **Authentication Request Protocol:** Defines a means by which a principal (or an agent acting on behalf of the principal) can request assertions containing authentication statements and, optionally, attribute statements. The Web Browser SSO Profile uses this protocol when redirecting a user from an SP to an IdP when it needs to obtain an assertion in order to establish a security context for the user at the SP.
  - **Single Logout Protocol:** Defines a mechanism to allow near-simultaneous logout of active sessions associated with a principal. The logout can be directly initiated by the user, or initiated by an IdP or SP because of a session timeout, administrator command, etc.
  - **Assertion Query and Request Protocol:** Defines a set of queries by which SAML assertions may be obtained. The *Request* form of this protocol can ask an asserting party for an existing assertion by referring to its assertion ID. The *Query* form of this protocol defines how a relying party can ask for assertions (new or existing) on the basis of a specific subject and the desired

statement type.

  - **Artifact Resolution Protocol:** Provides a mechanism by which SAML protocol messages may be passed by reference using a small, fixed-length value called an *artifact*. The artifact receiver uses the Artifact Resolution Protocol to ask the message creator to dereference the artifact and return the actual protocol message. The artifact is typically passed to a message recipient using one SAML binding (e.g. HTTP Redirect) while the resolution request and response take place over a synchronous binding, such as SOAP.
  - **Name Identifier Management Protocol:** Provides mechanisms to change the value or format of the name identifier used to refer to a principal. The issuer of the request can be either the service provider or the identity provider. The protocol also provides a mechanism to terminate an association of a name identifier between an identity provider and service provider.
  - **Name Identifier Mapping Protocol:** Provides a mechanism to programmatically map one SAML name identifier into another, subject to appropriate policy controls. It permits, for example, one SP to request from an IdP an identifier for a user that the SP can use at another SP in an application integration scenario.
- **Bindings:** SAML bindings detail exactly how the various SAML protocol messages can be carried over underlying transport protocols. The bindings defined by SAML V2.0 are:
  - **HTTP Redirect Binding:** Defines how SAML protocol messages can be transported using HTTP redirect messages (302 status code responses).
  - **HTTP POST Binding:** Defines how SAML protocol messages can be transported within the base64-encoded content of an HTML form control.
  - **HTTP Artifact Binding:** Defines how an artifact (described above in the Artifact Resolution Protocol) is transported from a message sender to a message receiver using HTTP. Two mechanisms are provided: either an HTML form control or a query string in the URL.
  - **SAML SOAP Binding:** Defines how SAML protocol messages are transported within SOAP 1.1 messages, with details about using SOAP over HTTP.
  - **Reverse SOAP (PAOS) Binding:** Defines a multi-stage SOAP/HTTP message exchange that permits an HTTP client to be a SOAP responder. Used in the Enhanced Client and Proxy Profile and particularly designed to support WAP gateways.
  - **SAML URI Binding:** Defines a means for retrieving an existing SAML assertion by resolving a URI (uniform resource identifier).
- **Profiles:** SAML profiles define how the SAML assertions, protocols, and bindings are combined and constrained to provide greater interoperability in particular usage scenarios. Some of these profiles are examined in detail later in this document. The profiles defined by SAML V2.0 are:
  - **Web Browser SSO Profile:** Defines how SAML entities use the Authentication Request Protocol and SAML Response messages and assertions to achieve single sign-on with standard web browsers. It defines how the messages are used in combination with the HTTP Redirect, HTTP POST, and HTTP Artifact bindings.
  - **Enhanced Client and Proxy (ECP) Profile:** Defines a specialized SSO profile where specialized clients or gateway proxies can use the Reverse-SOAP (PAOS) and SOAP bindings.
  - **Identity Provider Discovery Profile:** Defines one possible mechanism for service providers to learn about the identity providers that a user has previously visited.
  - **Single Logout Profile:** Defines how the SAML Single Logout Protocol can be used with SOAP, HTTP Redirect, HTTP POST, and HTTP Artifact bindings.
  - **Assertion Query/Request Profile:** Defines how SAML entities can use the SAML Query and Request Protocol to obtain SAML assertions over a synchronous binding, such as SOAP.

- **Artifact Resolution Profile:** Defines how SAML entities can use the Artifact Resolution Protocol over a synchronous binding, such as SOAP, to obtain the protocol message referred to by an artifact.
- **Name Identifier Management Profile:** Defines how the Name Identifier Management Protocol may be used with SOAP, HTTP Redirect, HTTP POST, and HTTP Artifact bindings.
- **Name Identifier Mapping Profile:** Defines how the Name Identifier Mapping Protocol uses a synchronous binding such as SOAP.

3.3 SAML XML Constructs and Examples

This section provides descriptions and examples of some of the key SAML XML constructs.

3.3.1 Relationship of SAML Components

An assertion contains one or more statements and some common information that applies to all contained statements or to the assertion as a whole. A SAML assertion is typically carried between parties in a SAML protocol response message, which itself must be transmitted using some sort of transport or messaging protocol.

Figure 5 shows a typical example of containment: a SAML assertion containing a series of statements, the whole being contained within a SAML response, which itself is within a SOAP body.

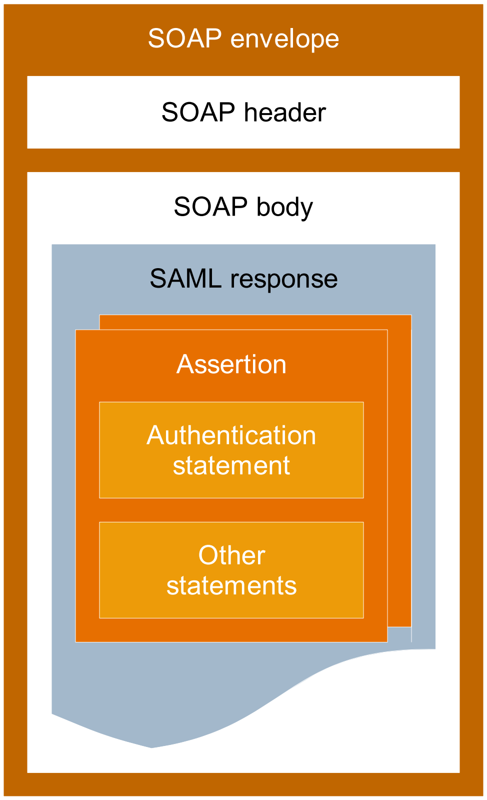


Figure 5: Relationship of SAML Components

3.3.2 Assertion, Subject, and Statement Structure

Figure 6 shows an XML fragment containing an example assertion with a single authentication statement. Note that the XML text in the figure (and elsewhere in this document) has been formatted for presentation purposes. Specifically, while line breaks and extra spaces are ignored between XML attributes within an XML element tag, when they appear between XML element start/end tags, they technically become part of the element value. They are inserted in the example only for readability.

- Line 1 begins the assertion and contains the declaration of the SAML assertion namespace, which is conventionally represented in the specifications with the `saml:` prefix.
- Lines 2 through 6 provide information about the nature of the assertion: which version of SAML is being used, when the assertion was created, and who issued it.
- Lines 7 through 12 provide information about the subject of the assertion, to which all of the contained statements apply. The subject has a name identifier (line 10) whose value is “j.doe@example.com”, provided in the format described on line 9 (email address). SAML defines various name identifier formats, and you can also define your own.
- The assertion as a whole has a validity period indicated by lines 14 and 15. Additional conditions on the use of the assertion can be provided inside this element; SAML predefines some and you can define your own. Timestamps in SAML use the XML Schema **dateTime** data type.

```
 1: <saml:Assertion xmlns:saml=”urn:oasis:names:tc:SAML:2.0:assertion”
 2:   Version="2.0"
 3:   IssueInstant="2005-01-31T12:00:00Z">
 4:   <saml:Issuer Format=urn:oasis:names:SAML:2.0:nameid-format:entity>
 5:     http://www.example.com
 6:   </saml:Issuer>
 7:   <saml:Subject>
 8:     <saml:NameID
 9:       Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
10:         j.doe@example.com
11:     </saml:NameID>
12:   </saml:Subject>
13:   <saml:Conditions
14:     NotBefore="2005-01-31T12:00:00Z"
15:     NotOnOrAfter="2005-01-31T12:10:00Z">
16:   </saml:Conditions>
17:   <saml:AuthnStatement
18:     AuthnInstant="2005-01-31T12:00:00Z" SessionIndex="67775277772">
19:     <saml:AuthnContext>
20:       <saml:AuthnContextClassRef>
21:         urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
22:       </saml:AuthnContextClassRef>
23:     </saml:AuthnContext>
24:   </saml:AuthnStatement>
25: </saml:Assertion>
```

Figure 6: Assertion with Subject, Conditions, and Authentication Statement

- The authentication statement appearing on lines 17 through 24 shows that this subject was originally authenticated using a password-protected transport mechanism (e.g. entering a username and password submitted over an SSL-protected browser session) at the time and date shown. SAML predefines numerous authentication context mechanisms (called classes), and you can also define your own mechanisms.

The `<NameID>` element within a `<Subject>` offers the ability to provide name identifiers in a number of different formats. SAML's predefined formats include:

- Email address
- X.509 subject name
- Windows domain qualified name
- Kerberos principal name
- Entity identifier

- Persistent identifier
- Transient identifier

Of these, persistent and transient name identifiers utilize privacy-preserving pseudonyms to represent the principal. ***Persistent identifiers*** provide a permanent privacy-preserving federation since they remain associated with the local identities until they are explicitly removed. ***Transient identifiers*** support “anonymity” at an SP since they correspond to a “one-time use” identifier created at the IdP. They are not associated with a specific local user identity at the SP and are destroyed once the user session terminates.

When persistent identifiers are created by an IdP, they are usually established for use only with a single SP. That is, an SP will only know about the persistent identifier that the IdP created for a principal for use when visiting that SP. The SP does not know about identifiers for the same principal that the IdP may have created for the user at other service providers. SAML does, however, also provide support for the concept of an ***affiliation*** of service providers which can share a single persistent identifier to identify a principal. This provides a means for one SP to directly utilize services of another SP in the affiliation on behalf of the principal. Without an affiliation, service providers must rely on the Name Identifier Mapping protocol and always interact with the IdP to obtain an identifier that can be used at some other specific SP.

3.3.3 Attribute Statement Structure

Attribute information about a principal is often provided as an adjunct to authentication information in single sign-on or can be returned in response to attribute queries from a relying party. SAML's attribute structure does not presume that any particular type of data store or data types are being used for the attributes; it has an attribute type-agnostic structure.

Figure 7 shows an XML fragment containing an example attribute statement.

```
 1: <saml:AttributeStatement>
 2:   <saml:Attribute
 3:      xmlns:x500="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
 4:      NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
 5:      Name="urn:oid:2.5.4.42"
 6:      FriendlyName="givenName">
 7:      <saml:AttributeValue xsi:type="xs:string"
 8:        x500:Encoding="LDAP">John</saml:AttributeValue>
 9:   </saml:Attribute>
10:   <saml:Attribute
11:     NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"
12:     Name="LastName">
13:     <saml:AttributeValue
14:       xsi:type="xs:string">Doe</saml:AttributeValue>
15:   </saml:Attribute>
16:   <saml:Attribute
17:     NameFormat=“http://smithco.com/attr-formats”
18:     Name=“CreditLimit”>
19:     xmlns:smithco=”http://www.smithco.com/smithco-schema.xsd”
20:     <saml:AttributeValue xsi:type=“smithco:type”>
21:       <smithco:amount currency=“USD”>500.00</smithco:amount>
22:     </saml:AttributeValue>
23:   </saml:Attribute>
24: </saml:AttributeStatement>
```

Figure 7: Attribute Statement

Note the following:

- A single statement can contain multiple attributes. In this example, there are three attributes (starting on lines 2, 10, and 16) within the statement.
- Attribute names are qualified with a name format (lines 4, 11, and 17) which indicates how the attribute name is to be interpreted. This example takes advantage of two of the SAML-defined ***attribute profiles*** and defines a third custom attribute as well. The first attribute uses the SAML

X.500/LDAP Attribute Profile to define a value for the LDAP attribute identified by the OID “2.5.4.42”. This attribute in an LDAP directory has a friendly name of “givenName” and the attribute's value is “John”. The second attribute utilizes the SAML ***Basic Attribute Profile***, refers to an attribute named “LastName” which has the value “Doe”. The name format of the third attribute indicates the name is not of a format defined by SAML, but is rather defined by a third party, SmithCo. Note that the use of private formats and attribute profiles can create significant interoperability issues. See the SAML Profiles specification [SAMLProf] for more information and examples.

- The value of an attribute can be defined by simple data types, as on lines 7 and 14, or can be structured XML, as on lines 20 through 22.

3.3.4 Message Structure and the SOAP Binding

In environments where communicating SAML parties are SOAP-enabled, the SOAP-over-HTTP binding can be used to exchange SAML request/response protocol messages. Figure 8 shows the structure of a SAML response message being carried within the SOAP body of a SOAP envelope, which itself has an HTTP response wrapper. Note that SAML itself does not make use of the SOAP header of a SOAP envelope but it does not prevent SAML-based application environments from doing so if needed.

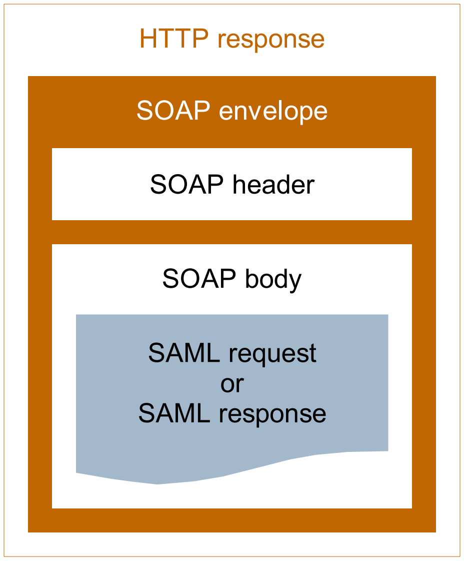


Figure 8: Protocol Messages Carried by SOAP Over HTTP

Figure 9 shows an XML document containing an example SAML authentication request message being transported within a SOAP envelope.

Note the following:

```
 1: <?xml version="1.0" encoding="UTF-8"?>
 2: <env:Envelope
 3:   xmlns:env=”http://www.w3.org/2003/05/soap/envelope/”>
 4:   <env:Body>
 5:     <samlp:AuthnRequest
 6:       xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
 7:       xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
 8:       Version="2.0"
 9:       ID="f0485a7ce95939c093e3de7b2e2984c0"
10:       IssueInstant="2005-01-31T12:00:00Z"
11:       Destination="https://www.AirlineInc.com/IdP/" >
12:       AssertionConsumerServiceIndex=”1”
13:       AttributeConsumingServiceIndex="0" >
14:       <saml:Issuer>http://www.CarRentalInc.com</saml:Issuer>
15:       <samlp:RequestedAuthnContext>
16:         <saml:AuthnContextClassRef>
17:           urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
18:         </saml:AuthnContextClassRef>
19:       <samlp:NameIDPolicy
20:          Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress"
21:       </samlp:NameIDPolicy>
22:     </samlp:AuthnRequest>
23:   </env:Body>
24: </env:Envelope>
```

Figure 9: Authentication Request in SOAP Envelope

- The SOAP envelope starts at line 2.
- The SAML authentication request starting on line 5 is embedded in a SOAP body element starting on line 4.
- The authentication request contains, from lines 6 through 13, various required and optional XML attributes including declarations of the SAML V2.0 assertion and protocol namespaces, the message ID, and the index of an assertion consumer service at the SP at which the IdP should return the response message.
- The request specifies a number of optional elements, from lines 15 through 21, that govern the type of assertion the requester expects back. This includes, for example, the requested type of name identifier (email address) and the authentication method with which the user must authenticate at the IdP (username/password over a protected transport).

An example XML fragment containing a SAML protocol Response message being transported in a SOAP message is shown in Figure 10.

```
 1: <?xml version="1.0" encoding="UTF-8"?>
 2: <env:Envelope  xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
 3:   <env:Body>
 4:     <samlp:Response
 5:       xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
 6:       xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
 7:       Version="2.0"
 8:       ID="i92f8b5230dc04d73e93095719d191915fdc67d5e"
 9:       IssueInstant="2005-11-10T06:47:42.000Z"
10:       InResponseTo="f0485a7ce95939c093e3de7b2e2984c0">
11:       <saml:Issuer>http://www.AirlineInc.com</saml:Issuer>
12:       <samlp:Status>
13:         <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
14:       </samlp:Status>
15:                   ...SAML assertion...
16:     </samlp:Response>
17:   </env:Body>
18: </env:Envelope>
```

Figure 10: Response in SOAP Envelope

Note the following:

- On line 10, the Response header `InResponseTo` XML attribute references the request to which the asserting party is responding, and specifies additional information (lines 7 through 14) needed to process the response, including status information. SAML defines a number of status codes and, in many cases, dictates the circumstances under which they must be used.

- Within the response (line 15; detail elided) is a SAML assertion, typically containing one or more statements as discussed earlier.

3.4 Security in SAML

Just providing assertions from an asserting party to a relying party may not be adequate to ensure a secure system. How does the relying party trust what is being asserted to it? In addition, what prevents a “man-in-the-middle” attack that might grab assertions to be illicitly “replayed” at a later date? These and many more security considerations are discussed in detail in the SAML Security and Privacy Considerations specification [SAMLSec]. SAML defines a number of security mechanisms to detect and protect against such attacks. The primary mechanism is for the relying party and asserting party to have a pre-existing trust relationship which typically relies on a Public Key Infrastructure (PKI). While use of a PKI is not mandated by SAML, it is recommended. Use of particular security mechanisms are described for each SAML binding. A general overview of what is recommended is provided below:

- Where message integrity and message confidentiality are required, then HTTP over SSL 3.0 or TLS 1.0 is recommended.
- When a relying party requests an assertion from an asserting party, bi-lateral authentication is required and the use of SSL 3.0 or TLS 1.0 using mutual authentication or authentication via digital signatures is recommended.
- When a response message containing an assertion is delivered to a relying party via a user's web browser (for example using the HTTP POST binding), then to ensure message integrity, it is mandated that the response message be digitally signed using the XML signature recommendation.

3.5 Use of SAML in Other Frameworks

SAML's components are modular and extensible, and it has been adopted for use with several other standard frameworks. Following are some examples.

3.5.1 Web Services Security (WS-Security)

SAML assertions can be conveyed by means other than the SAML Request/Response protocols or profiles defined by the SAML specification set. One example of this is their use with Web Services Security (WS-Security), which is a set of specifications that define means for providing security protection of SOAP messages. The services provided WS-Security are authentication, data integrity, and confidentiality.

WS-Security defines a `<Security>` element that may be included in a SOAP message header. This element specifies how the message is protected. WS-Security makes use of mechanisms defined in the W3C XML Signature and XML Encryption specifications to sign and encrypt message data in both the SOAP header and body. The information in the `<Security>` element specifies what operations were performed and in what order, what keys were used for these operations, and what attributes and identity information are associated with that information. WS-Security also contains other features, such as the ability to timestamp the security information and to address it to a specified Role.

In WS-Security, security data is specified using security *tokens*. Tokens can either be binary or structured XML. Binary tokens, such as X.509 Certificates and Kerberos Tickets are carried in an XML wrapper. XML tokens, such as SAML assertions, are inserted directly as sub-elements of the `<Security>` element. A Security Token Reference may also be used to refer to a token in one of a number of ways.

WS-Security consists of a core specification [WSS], which describes the mechanisms independent of the type of token being used, and a number of token profiles which describe the use of particular types of tokens. Token profiles cover considerations relating to that particular token type and methods of referencing the token using a Security Token Reference. The use of SAML assertions with WS-Security is described in the SAML Token Profile [WSSSAML].

Because the SAML protocols have a binding to SOAP, it is easy to get confused between that SAML-defined binding and the use of SAML assertions by WS-Security. They can be distinguished by their purpose, the message format, and the parties involved in processing the messages.

The characteristics of the SAML Request/Response protocol binding over SOAP are as follows:

- It is used to obtain SAML assertions for use external to the SOAP message exchange; they play no role in protecting the SOAP message.
- The SAML assertions are contained within a SAML Response, which is carried in the body of the SOAP envelope.
- The SAML assertions are provided by a trusted authority and may or may not pertain to the party requesting them.

The characteristics of the use of SAML assertions as defined by WS-Security are as follows:

- The SAML assertions are carried in a `<Security>` element within the header of the SOAP envelope as shown in Figure 11.
- The SAML assertions usually play a role in the protection of the message they are carried in; typically they contain a key used for digitally signing data within the body of the SOAP message.
- The SAML assertions will have been obtained previously and typically pertain to the identity of the sender of the SOAP message.

Note that in principle, SAML assertions could be used in both ways in a single SOAP message. In this case the assertions in the header would refer to the identity of the Responder (and Requester) of the message. However, at this time, SAML has not profiled the use of WS-Security to secure the SOAP message exchanges that are made within a SAML deployment.

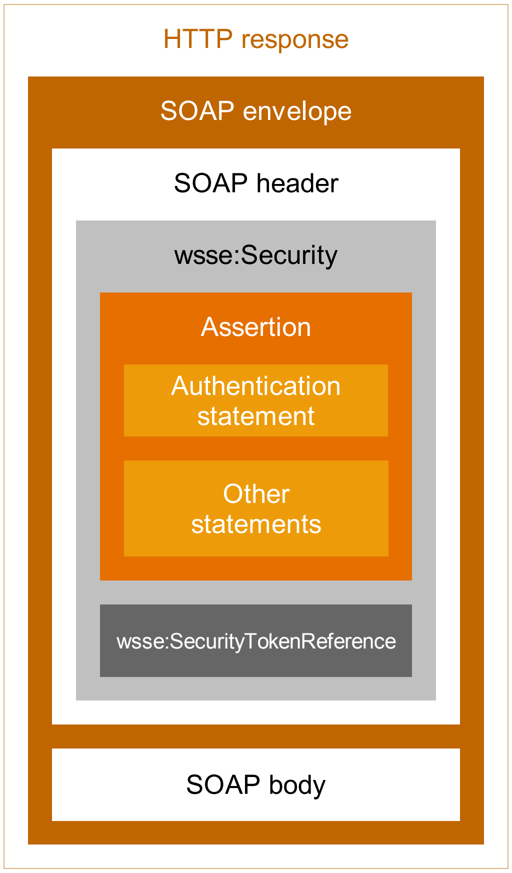


Figure 11: WS-Security with a SAML Token

The following sequence of steps typifies the use of SAML assertions with WS-Security.

A SOAP message sender obtains a SAML assertion by means of the SAML Request/Response protocol or other means. In this example, the assertion contains an attribute statement and a subject with a confirmation method called *Holder of Key* [@@turn this into a forward reference to an advanced topic?]. To protect the SOAP message:

1. The sender constructs the SOAP message, including a SOAP header with a WS-Security header. A SAML assertion is placed within a WS-Security token and included in the security header. The key referred to by the SAML assertion is used to construct a digital signature over data in the SOAP message body. Signature information is also included in the security header.
2. The message receiver verifies the digital signature.
3. The information in the SAML assertion is used for purposes such as Access Control and Audit logging.

Figure 12 illustrates this usage scenario.

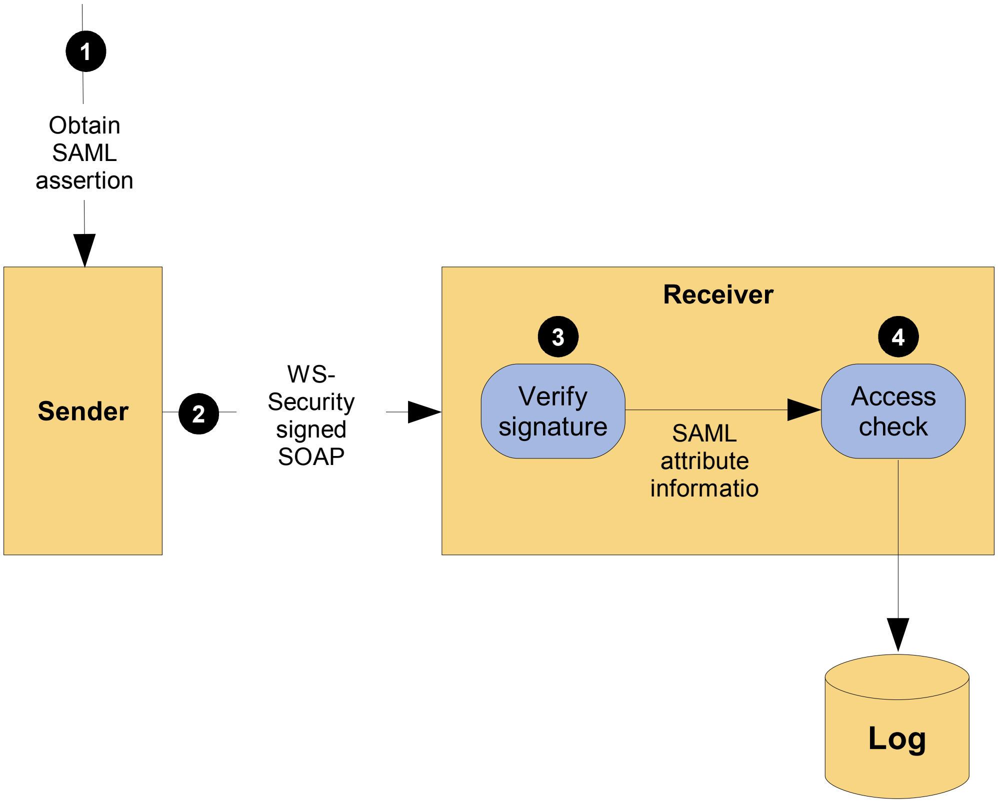


Figure 12: Typical Use of WS-Security with SAML Token

3.5.2 eXtensible Access Control Markup Language (XACML)

SAML assertions provide a means to distribute security-related information that may be used for a number of purposes. One of the most important of these purposes is as input to Access Control decisions. For example, it is common to consider when and how a user authenticated or what their attributes are in deciding if a request should be allowed. SAML does not specify how this information should be used or how access control policies should be addressed. This makes SAML suitable for use in a variety of environments, including ones that existed prior to SAML.

The eXtensible Access Control Markup Language (XACML) is an OASIS Standard that defines the syntax and semantics of a language for expressing and evaluating access control policies. The work to define XACML was started slightly after SAML began. From the beginning they were viewed as related efforts and consideration was given to specifying both within the same Technical Committee. Ultimately, it was decided to allow them to proceed independently but to align them. Compatibility with SAML was written in to the charter of the XACML TC.

As a result, SAML and XACML can each be used independently of the other, or both can be used together. Figure 13 illustrates the typical use of SAML with XACML.

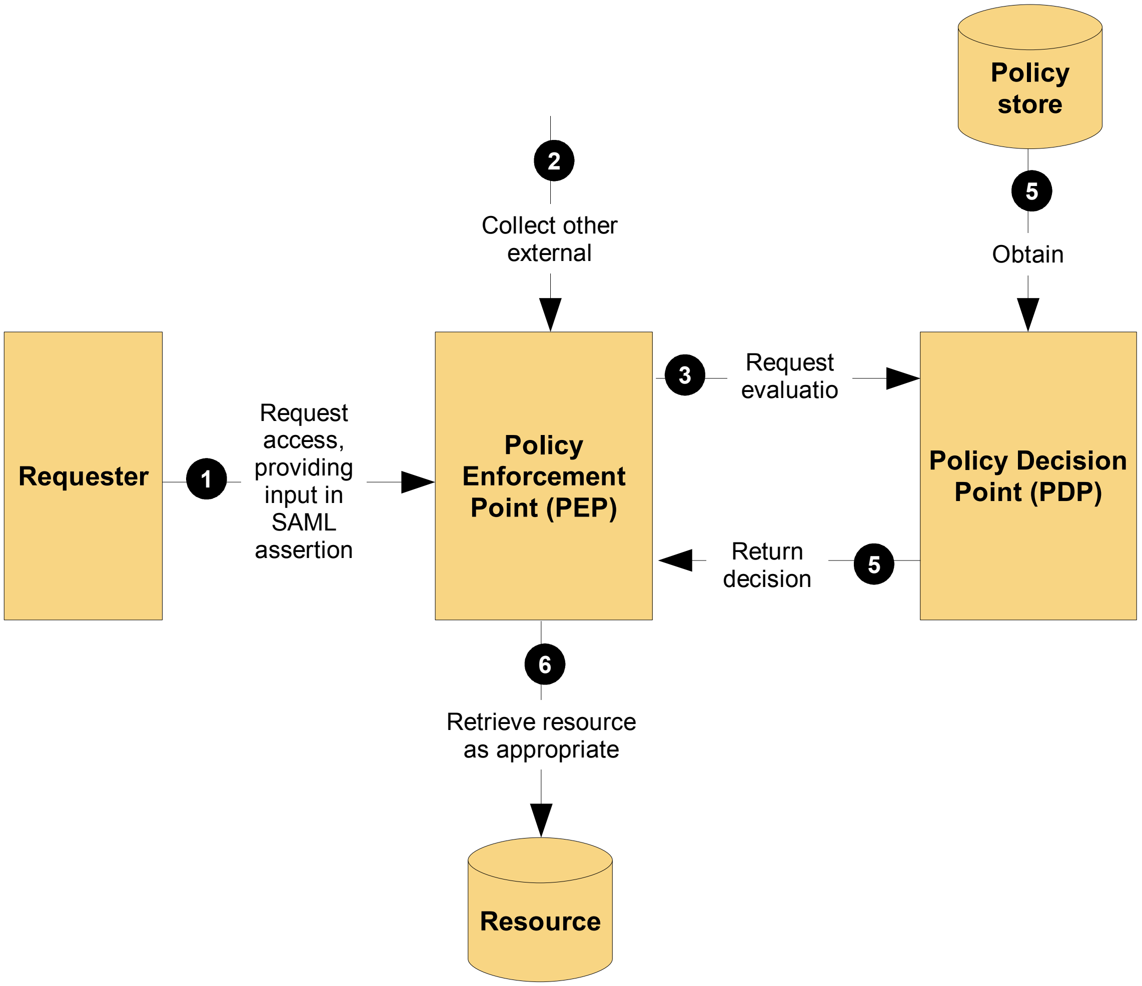

Policy store
5
Obtain
2
Collect other external
Requester
1
Request access, providing input in SAML assertion
Policy Enforcement Point (PEP)
3
Request evaluatio
Policy Decision Point (PDP)
Return decision
5
6
Retrieve resource as appropriate
Resource

Figure 13: SAML and XACML Integration

Using SAML and XACML in combination would typically involve the following steps.

1. An XACML Policy Enforcement Point (PEP) receives a request to access some resource.
2. The PEP obtains SAML assertions containing information about the parties to the request, such as the requester, the receiver (if different) or intermediaries. These assertions might accompany the request or be obtained directly from a SAML Authority, depending on the SAML profile used.
3. The PEP obtains other information relevant to the request, such as time, date, location, and properties of the resource.
4. The PEP presents all the information to a Policy Decision Point (PDP) to decide if the access should be allowed.
5. The PDP obtains all the policies relevant to the request and evaluates them, combining conflicting results if necessary.
6. The PDP informs the PEP of the decision result.
7. The PEP enforces the decision, by either allowing the requested access or indicating that access is not allowed.

The SAML and XACML specification sets contain some features specifically designed to facilitate their combined use.

The XACML Attribute Profile in the SAML Profiles specification defines how attributes can be described using SAML syntax so that they may be automatically mapped to XACML Attributes. A schema is provided by SAML to facilitate this.

A document that was produced by the XACML Technical Committee, SAML V2.0 profile of XACML v2.0, provides additional information on mapping SAML Attributes to XACML Attributes. This profile also defines a new type of Authorization decision query specifically designed for use in an XACML environment. It extends the SAML protocol schema and provides a request and response that contains exactly the inputs and outputs defined by XACML.

That same document also contains two additional features that extend the SAML schemas. While they are not, strictly speaking, intended primarily to facilitate combining SAML and XACML, they are worth noting. The first is the XACML Policy Query. This extension to the SAML protocol schema allows the SAML protocol to be used to retrieve XACML policy which may be applicable to a given access decision.

The second feature extends the SAML schema by allowing the SAML assertion envelope to be used to wrap an XACML policy. This makes available to XACML features such as Issuer, Validity interval and signature, without requiring the definition of a redundant or inconsistent scheme. This promotes code and knowledge reuse between SAML and XACML.

4 Major Profiles and Federation Use Cases

As mentioned earlier, SAML defines a number of profiles to describe and constrain the use of SAML protocol messages and assertions to solve specific business use cases. This section provides greater detail on some of the most important SAML profiles and identity federation use cases.

4.1 Web Browser SSO Profile

This section describes the typical flows likely to be used with the web browser SSO profile of SAML V2.0.

4.1.1 Introduction

The Web Browser SSO Profile defines how to use SAML messages and bindings to support the web SSO use case described in section 2.2. This profile provides a wide variety of options, primarily having to do with two dimensions of choice: first whether the message flows are IdP-initiated or SP-initiated, and second, which bindings are used to deliver messages between the IdP and the SP.

The first choice has to do with where the user starts the process of a web SSO exchange. SAML supports two general message flows to support the processes. The most common scenario for starting a web SSO exchange is the SP-initiated web SSO model which begins with the user choosing a browser bookmark or clicking a link that takes them directly to an SP application resource they need to access. However, since the user is not logged in at the SP, before it allows access to the resource, the SP sends the user to an IdP to authenticate. The IdP builds an assertion representing the user's authentication at the IdP and then sends the user back to the SP with the assertion. The SP processes the assertion and determines whether to grant the user access to the resource.

In an IdP-initiated scenario, the user is visiting an IdP where they are already authenticated and they click on a link to a partner SP. The IdP builds an assertion representing the user's authentication state at the IdP and sends the user's browser over to the SP's assertion consumer service, which processes the assertion and creates a local security context for the user at the SP. This approach is useful in certain environments, but requires the IdP to be configured with inter-site transfer links to the SP's site.

[@@Separate this info out into an advanced topic later on? - it could move to section 4.1.2 with the IDP-initiated scenario] SAML V2.0 does not specify a mechanism to indicate to the SP that the user would like access to a specific resource there. However, a common convention has been adopted by some SAML implementations to work around this limitation. The convention relies on the fact that no `RelayState` [@@need to introduce this concept – should it go in section 2.2?] data is exchanged since the user did not first visit the SP. In this use case, the IdP will create `RelayState` data containing the URL of a desired resource at the SP and send it to the SP with the SAML Response message. Note that `RelayState` is limited to 80 bytes of data, and thus the target resource URL is constrained to that size, although SP-relative URL's may help obviate the limitation.

Figure 14 compares the IdP-initiated and SP-initiated models.

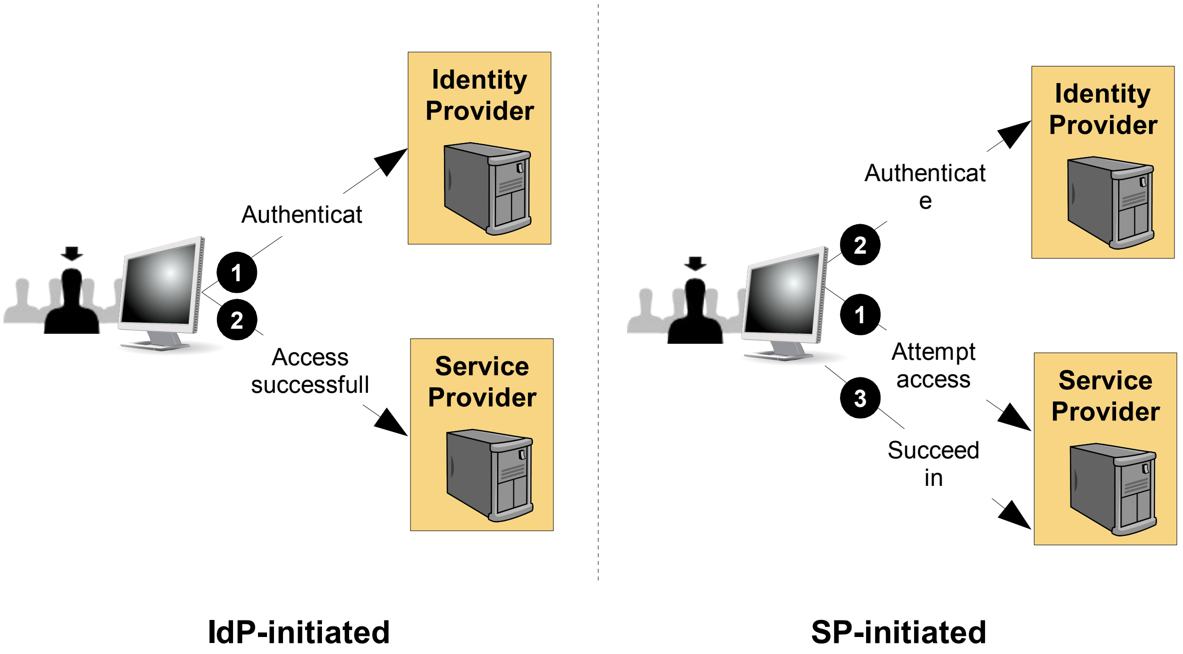


Figure 14: Differences in Initiation of Web Browser SSO

The second choice to be made when using the SAML profiles centers around which SAML bindings will be used when sending messages back and forth between the IdP and SP. There are many combinations of message flows and bindings that are possible, many of which are discussed in the following subsections. For the web SSO profile, we are mainly concerned with two SAML messages; namely an Authentication Request message sent from an SP to an IdP, and a Response message containing a SAML assertion that is sent from the IdP to the SP (and then, secondarily, with messages related to artifact resolution if that binding is chosen).

The SAML Conformance [SAMLConform] and Profiles [SAMLProf] specifications identify the SAML bindings that can legally be used with these two messages. Specifically, an Authentication Request message can be sent from an SP to an IdP using either the HTTP Redirect Binding, HTTP POST Binding, or HTTP Artifact Binding. The Response message can be sent from an IdP to an SP using either the HTTP POST Binding or the HTTP Artifact Binding. For this pair of messages, SAML permits asymmetry in the choice of bindings used. That is, a request can be sent using one binding and the response can be returned using a different binding. The decision of which bindings to use is typically driven by configuration settings at the IdP and SP systems. Factors such as potential message sizes, whether identity information is allowed to transit through the browser, etc. must be considered in the choice of bindings.

The following subsections describe the detailed message flows involved in web SSO exchanges for the following use case scenarios:

- SP-initiated SSO using a Redirect Binding for the SP-to-IdP `<AuthnRequest>` message and a POST Binding for the IdP-to-SP `<Response>` message
- SP-initiated SSO using a POST Binding for the `<AuthnRequest>` message and an Artifact Binding for the `<Response>` message
- IDP-initiated SSO using a POST Binding for the IdP-to-SP `<Response>` message; no SP-to-IdP `<AuthnRequest>` message is involved.

4.1.2 SP-Initiated SSO: Redirect/POST Bindings

This first example describes an SP-initiated SSO exchange. In such an exchange, the user attempts to access a resource on the SP www.abc.com. However they do not have a current logon session on this site and their federated identity is managed by their IdP, www.xyz.com. They are sent to the IdP to log on and the IdP provides a SAML web SSO assertion for the user's federated identity back to the SP.

For this specific use case, the HTTP Redirect Binding is used to deliver the SAML `<AuthnRequest>` message to the IdP and the HTTP POST Binding is used to return the SAML `<Response>` message containing the assertion to the SP. Figure 15 illustrates the message flow.

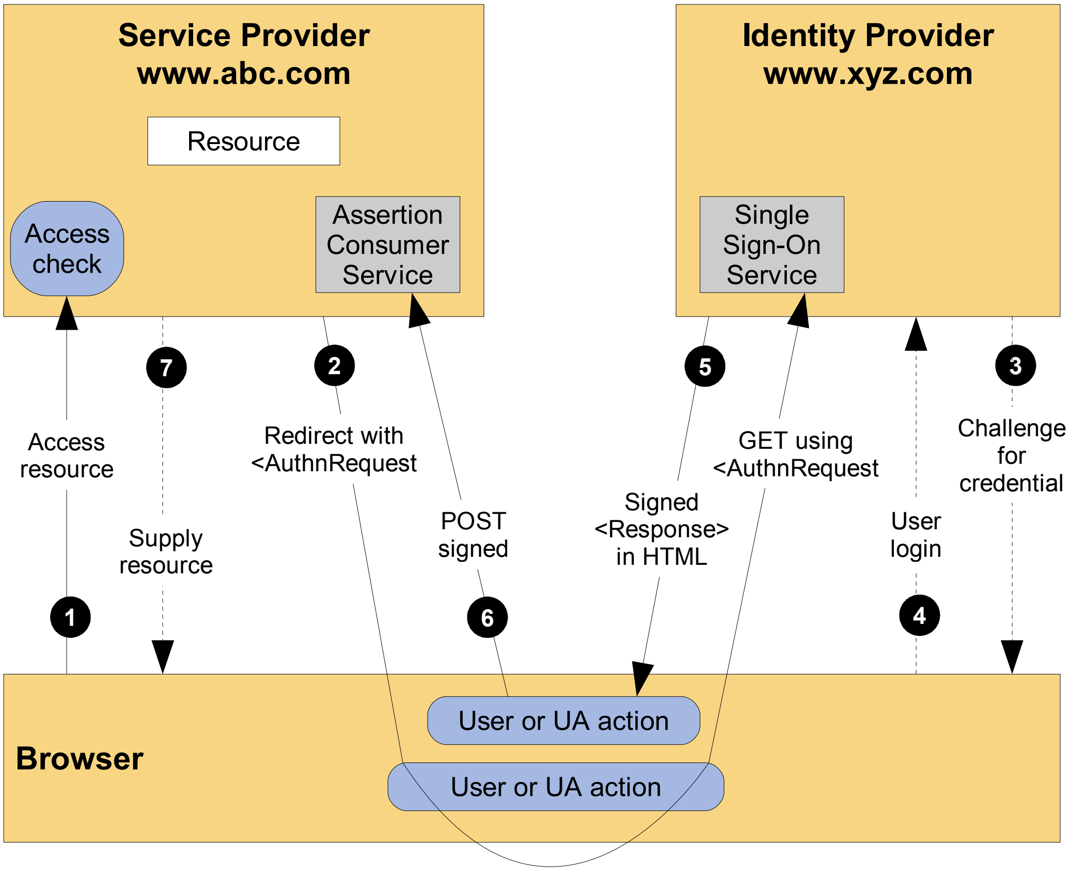


Figure 15: SP-Initiated SSO with Redirect and POST Bindings

The processing is as follows:

1. The user attempts to access a resource on www.abc.com. The user does not have a valid logon session (i.e. security context) on this site. The SP saves the requested resource URL in local state information that can be saved across the web SSO exchange.
2. The SP sends an HTTP redirect response to the browser (HTTP status 302 or 303). The Location HTTP header contains the destination URI of the Sign-On Service at the identity provider together with an `<AuthnRequest>` message encoded as a URL query variable named `SAMLRequest`. The query string is encoded using the DEFLATE encoding. The browser processes the redirect response and issues an HTTP GET request to the IdP's Single Sign-On Service with the `SAMLRequest` query parameter. The local state information (or a reference to it) is also included in the HTTP response encoded in a `RelayState` query string parameter.
3. The Single Sign-On Service determines whether the user has an existing logon security context at the identity provider that meets the default or requested (in the `<AuthnRequest>`) authentication policy requirements. If not, the IdP interacts with the browser to challenge the user to provide valid credentials.
4. The user provides valid credentials and a local logon security context is created for the user at the IdP.

5. The IdP Single Sign-On Service builds a SAML assertion representing the user's logon security context. Since a POST binding is going to be used, the assertion is digitally signed and then placed within a SAML `<Response>` message. The `<Response>` message is then placed within an HTML FORM as a hidden form control named `SAMLResponse`. If the IdP received a `RelayState` value from the SP, it must return it unmodified to the SP in a hidden form control named `RelayState`. The Single Sign-On Service sends the HTML form back to the browser in the HTTP response. For ease of use purposes, the HTML FORM typically will be accompanied by script code that will automatically post the form to the destination site.-
6. The browser, due either to a user action or execution of an "auto-submit" script, issues an HTTP POST request to send the form to the SP's Assertion Consumer Service. The service provider's Assertion Consumer Service obtains the `<Response>` message from the HTML FORM for processing. The digital signature on the SAML assertion must first be validated and then the assertion contents are processed in order to create a local logon security context for the user at the SP. Once this completes, the SP retrieves the local state information indicated by the `RelayState` data to recall the originally-requested resource URL. It then sends an HTTP redirect response to the browser directing it to access the originally requested resource (not shown).
7. An access check is made to establish whether the user has the correct authorization to access the resource. If the access check passes, the resource is then returned to the browser.

4.1.3 SP-Initiated SSO: POST/Artifact Bindings

This use case again describes an SP-initiated SSO exchange.

However, for this use case, the HTTP POST binding is used to deliver the SAML `<AuthRequest>` to the IdP and the SAML `<Response>` message is returned using the Artifact binding. The HTTP POST binding may be necessary for an <AuthnRequest> message in cases where it's length precludes the use of the HTTP Redirect binding. The message may be long enough to require a POST binding when, for example, it includes many of its optional elements and attributes or when it must be digitally signed.

When using the HTTP Artifact binding for the SAML `<Response>` message, SAML permits the artifact to be delivered via the browser using either an HTTP POST or HTTP Redirect response (not to be confused with the SAML HTTP POST and Redirect "bindings"). In this example, the artifact is delivered using an HTTP POST of an HTML form.

Once the SP is in possession of the artifact, it contacts the IdP's Artifact Resolution Service to obtain the SAML message using the synchronous SOAP binding that corresponds to the artifact. Figure 16 illustrates the message flow.

The processing is as follows: [@@rsp:I would prefer to see the artifact sent via redirect response since that is the most common and so people don't confuse it with the POST binding]

Figure 16: SP initiated: POST/Artifact Bindings

1. The user attempts to access a resource on www.abc.com. The user does not have a valid logon session (i.e. security context) on this site. The SP saves the requested resource URL in local state information that can be saved across the web SSO exchange.

2. The SP sends an HTML form back to the browser in the HTTP response (HTTP status 200). The HTML FORM contains a SAML `<AuthnRequest>` message encoded as the value of a hidden form control named `SAMLRequest`. The local state information (or a reference to it) is also included in the form in a hidden form control named `RelayState`. For ease of use purposes, the HTML FORM typically will be accompanied by script code that will automatically post the form to the destination site.

3. The browser, due either to a user action or execution of an “auto-submit” script, issues an HTTP POST request to send the form to the identity provider's Single Sign-On Service.

4. The Single Sign-On Service determines whether the user has an existing logon security context at the identity provider that meets the default or requested (in the `<AuthnRequest>)` authentication policy requirements. If not, the IdP interacts with the browser to challenge the user to provide valid credentials.

5. The user provides valid credentials and a local logon security context is created for the user at the IdP.

6. The IdP Single Sign-On Service builds a SAML assertion representing the user's logon security context and places the assertion within a SAML `<Response>` message. Since the HTTP Artifact binding will be used to deliver the SAML Response message, it is not mandated that the assertion be digitally signed. The IdP creates an artifact containing the source ID for the www.xyz.com site and a reference to the `<Response>` message (the MessageHandle). The HTTP Artifact binding allows the choice of either HTTP redirection or an HTML form POST as the mechanism to deliver the artifact to the partner. The figure shows the use of the HTML form POST mechanism. To do this, the Single Sign-On Service sends an HTML form back to the browser in the HTTP response. The HTML FORM contains the SAML artifact with the hidden form control named `SAMLart` and the `RelayState` data

(if any) in a hidden form control named `RelayState`. For ease of use purposes, the HTML FORM typically will be accompanied by script code that will automatically post the form to the destination site.

7. The browser, due either to a user action or execution of an "auto-submit" script, issues an HTTP POST request to send the form to the SP's Assertion Consumer Service. Upon receiving the HTTP message, the Assertion Consumer Service extracts the SourceID from the SAML artifact and locates the configuration of a partner entity represented by that SourceID (the www.xyz.com IdP in this example). The Assertion Consumer Service must also retrieve the `RelayState` data from the form for use after the IdP returns the message associated with the artifact.

8. The SP's Assertion Consumer Service now builds and sends a SAML `<ArtifactResolve>` message containing the artifact to the IdP's Artifact Resolution Service endpoint. This exchange is performed using a synchronous SOAP message exchange.

9. The IdP's Artifact Resolution Service extracts the MessageHandle from the artifact and locates the original SAML `<Response>` message associated with it. This message is then placed inside a SAML `<ArtifactResponse>` message which is returned to the SP over the SOAP channel. The SP extracts and processes the `<Response>` message and then processes the embedded assertion in order to create a local logon security context for the user at the SP. Once this completes, the SP retrieves the local state information indicated by the `RelayState` data to recall the originally-requested resource URL. It then sends an HTTP redirect response to the browser directing it to access the originally requested resource (not shown).

10. An access check is made to establish whether the user has the correct authorization to access the resource. If the access check passes, the resource is then returned to the browser.

4.1.4 IdP-Initiated SSO: POST Binding

In addition to supporting the new SP-Initiated web SSO use cases, SAML v2 continues to support the IdP-initiated web SSO use cases originally supported by SAML v1. In an IdP-initiated use case, the identity provider is configured with specialized links that refer to the desired service providers. These links actually refer to the local IdP's Single Sign-On Service and pass parameters to the service identifying the remote SP. So instead of visiting the SP directly, the user accesses the IdP site and clicks on one of the links to gain access to the remote SP. This triggers the creation of a SAML assertion that, in this example, will be transported to the service provider using the HTTP POST binding.

Figure 17 shows the process flow for an IdP-initiated web SSO exchange. This example assumes the SP and IdP support the RelayState convention described earlier.

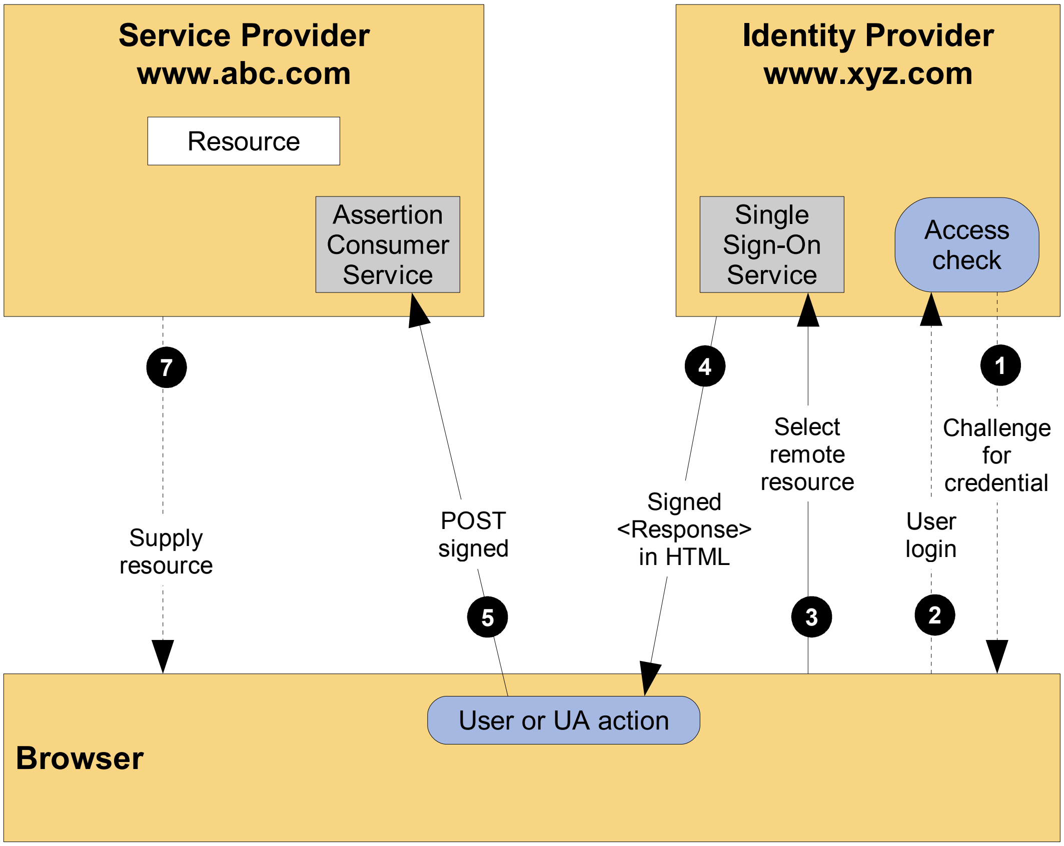


Figure 17: IdP-Initiated SSO with POST Binding

The processing is as follows: [@@rsp: need to show use of RelayState to carry resource URL?]

1. If the user does not have a valid local security context at the IdP, at some point the user will be challenged to supply their credentials to the IdP site (www.xyz.com).
2. The user provides valid credentials and a local logon security context is created for the user at the IdP.
3. The user selects a menu option or link on the IdP to request access to an SP web site (www.abc.com). This causes the IDP's Single Sign-On Service to be called.
4. The Single Sign-On Service builds a SAML assertion representing the user's logon security context. Since a POST binding is going to be used, the assertion is digitally signed before it is placed within a SAML `<Response>` message. The `<Response>` message is then placed within an HTML FORM as a hidden form control named `SAMLResponse`. If the convention for identifying a specific application resource at the SP is supported at the IdP and SP, the resource URL at the SP is also encoded into the form using a hidden form control named `RelayState.` The Single Sign-On Service sends the HTML form back to the browser in the HTTP response. For ease of use purposes, the HTML FORM typically will also contain script code that will automatically post the form to the destination site.
5. The browser, due either to a user action or execution of an “auto-submit” script, issues an HTTP POST request to send the form to the SP's Assertion Consumer Service. The service provider's Assertion Consumer Service obtains the `<Response>` message from the HTML FORM for processing. The digital signature on the SAML assertion must first be validated and then the assertion contents are processed in order to create a local logon security context for the user at the SP. Once this completes, the SP retrieves the `RelayState` data to determine the desired application resource URL. It then sends an HTTP redirect response to the browser directing it to access the requested resource (not shown).
6. An access check is made to establish whether the user has the correct authorization to access the

resource. If the access check passes, the resource is then returned to the browser.

4.2 ECP Profile

4.2.1 Introduction

The Enhanced Client and Proxy (ECP) Profile supports several SSO use cases, in particular:

- Use of a proxy server, for example a WAP gateway in front of a mobile device which has limited functionality
- Clients where it is impossible to use redirects
- It is impossible for the identity provider and service provider to directly communicate (and hence the HTTP Artifact binding cannot be used)

Figure 18 illustrates two use cases for using the ECP Profile.

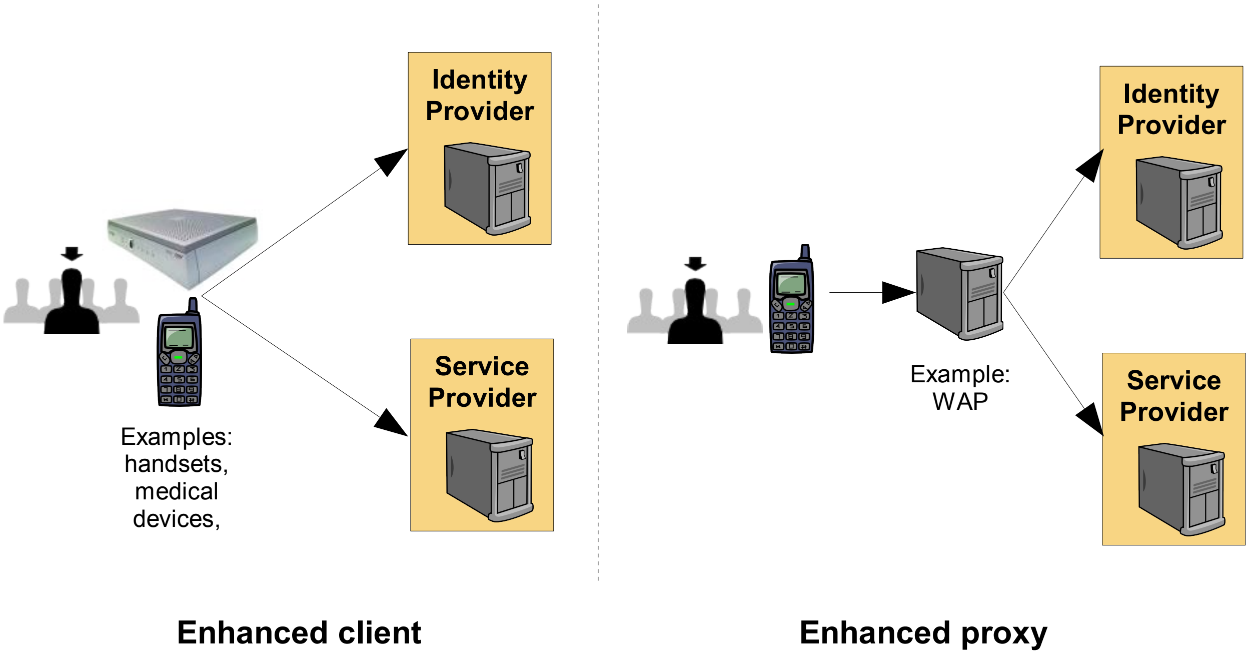


Figure 18: Enhanced Client/Proxy Use Cases

The ECP profile defines a single binding – PAOS (Reverse SOAP). The profile uses SOAP headers and SOAP bodies to transport SAML `<AuthnRequest>` and SAML `<Response>` messages between the service provider and the identity provider.

4.2.2 ECP Profile using PAOS binding

Figure 19 shows the message flows between the ECP, service provider and identity provider. The ECP is shown as a single logical entity.

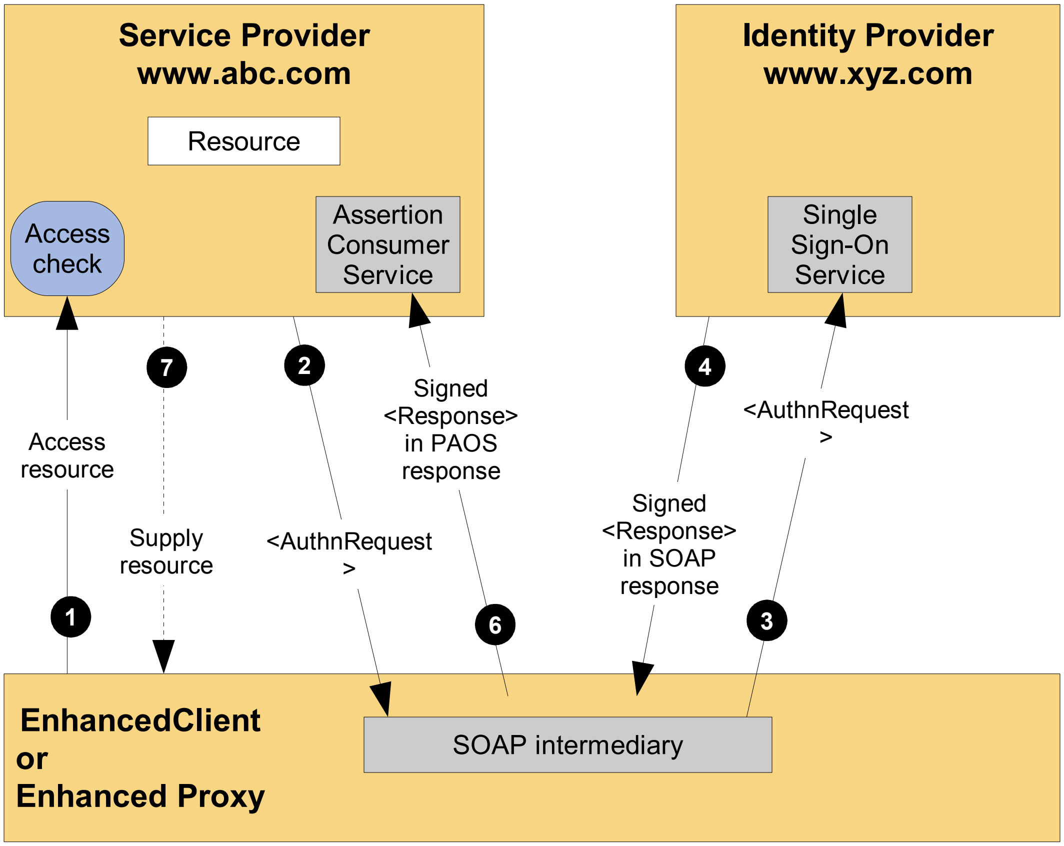


Figure 19: SSO Using ECP with the PAOS Binding

The processing is as follows:

1. The ECP wishes to gain access to a resource on the service provider (www.abc.com). The ECP will issue an HTTP request for the resource. The HTTP request contains a PAOS HTTP header defining that the ECP service is to be used.
2. Accessing the resource requires that the principal has a valid security context, and hence a SAML assertion needs to be supplied to the service provider. In the HTTP response to the ECP an `<AuthnRequest>` is carried within a SOAP body. Additional information, using the PAOS binding, is provided back to the ECP
3. After some processing in the ECP the `<AuthnRequest>` is sent to the appropriate identity provider using the SAML SOAP binding.
4. The identity provider validates the `<AuthnRequest>` and sends back to the ECP a SAML `<Response>`, again using the SAML SOAP binding.
5. The ECP extracts the `<Response>` and forwards it to the service provider as a PAOS response.
6. The service provider sends to the ECP an HTTP response containing the resource originally requested.

4.3 Single Logout Profile

4.3.1 Introduction

Single Logout permits near real-time session logout of a user from all participants in a session. A request can be issued by any session participant to request that the session is to be ended. As specified in the SAML Conformance specification [SAMLConform], the SAML logout messages can be exchanged over either the synchronous SOAP over HTTP binding or using the asynchronous HTTP Redirect, HTTP

POST, or HTTP Artifact bindings. Note that a browser logout operation often requires access to local authentication cookies stored in the user's browser. Thus, asynchronous front-channel bindings are typically preferred for these exchanges in order to force the browser to visit each session participant to permit access to the browser cookies. However, user interaction with the browser might interrupt the process of visiting each participant and thus, the result of the logout process cannot be guaranteed.

4.3.2 SP-Initiated Single Logout

In the example shown in Figure 20, a user visiting the *CarRentalInc.com* service provider web site decides that they wish to log out of their web SSO session. This example shows the use of the SOAP over HTTP binding for the message exchange.

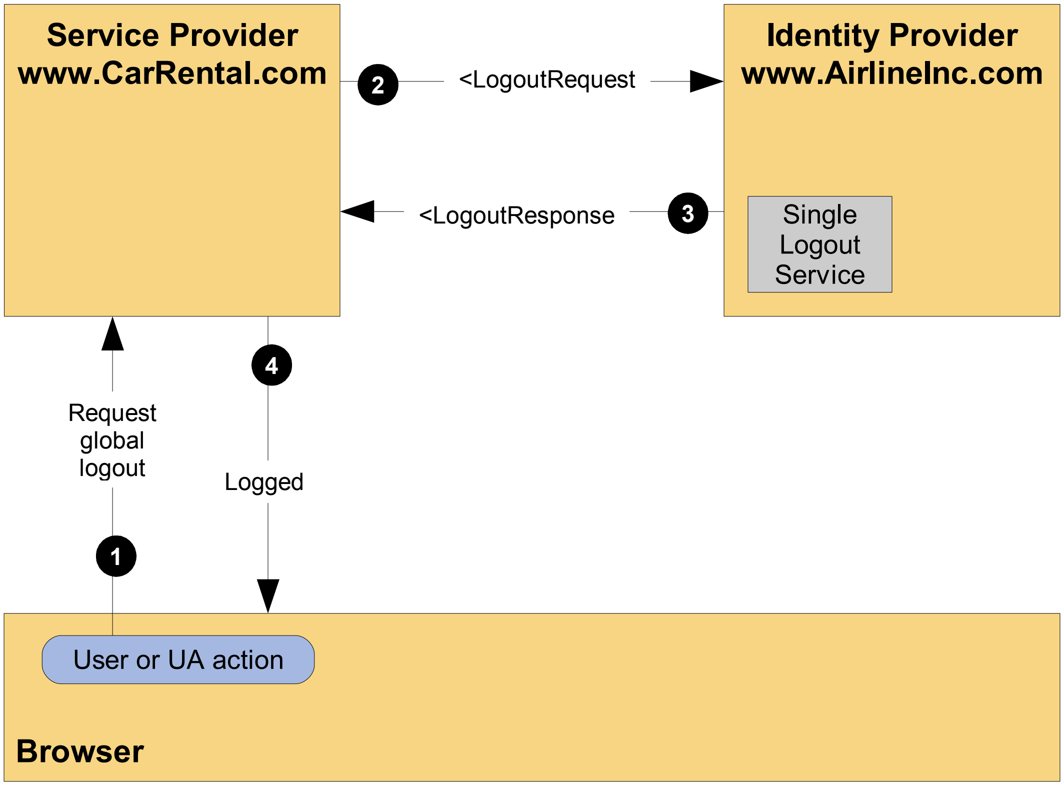


Figure 20: SP-Initiated Single Logout with a Single SP

The processing is as follows:

1. A user was previously authenticated by the *AirlineInc.com* identity provider and is interacting with the *CarRentalInc.com* service provider through a web SSO session. The user decides to terminate their session and selects a single logout link on the SP.
2. The SP destroys the local authentication session state for the user and then sends the *AirlineInc.com* identity provider a SAML `<LogoutRequest>` message requesting that the user's session be logged out. The request identifies the principal to be logged out using a `<NameID>` element, as well as providing a `<SessionIndex>` element to uniquely identify the session being closed. The `<LogoutRequest>` message is digitally signed by the service provider and is placed in a SOAP message which is transmitted using the SAML SOAP over HTTP binding.
3. The identity provider verifies the digital signature ensuring that the `<LogoutRequest>` originated from a known and trusted service provider. The identity Provider processes the request, destroys any local session information for the user, and returns a `<LogoutResponse>` message containing a suitable status code response. The response is digitally signed and returned using the SOAP over HTTP binding.

4.3.3 -Initiated Single Logout with Multiple SPs

If in step 3 above, the identity provider determines that other service providers are also participants in the web SSO session, the IdP will send `<LogoutRequest>` messages to each of the other SPs. Figure 21 illustrates this processing. In this example, different bindings are used between the exchanges bewtwwen the various session participants. The SP initiating the single logout uses the HTTP Redirect Binding with the IdP, while the IdP uses a back channel SOAP over HTTP Binding to communicate with the other SP

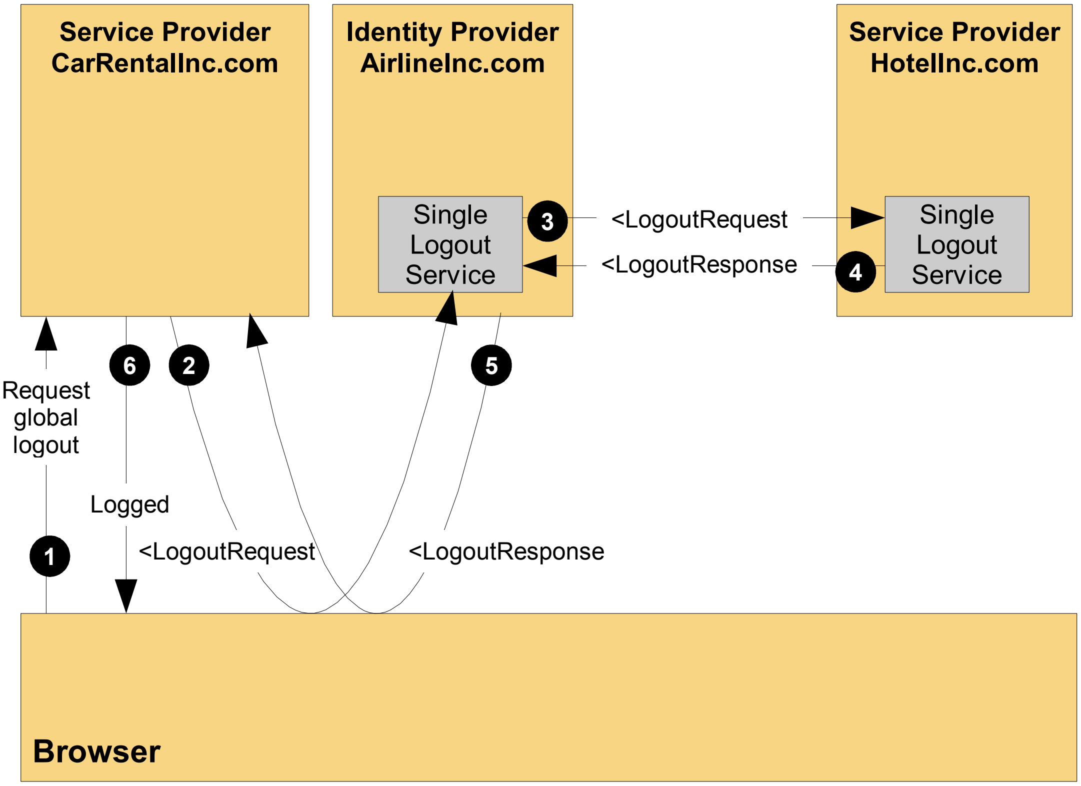


Figure 21: SP-Initiated Single Logout with Multiple SPs

4.3.4 IDP-Initiated Single Logout with Multiple SPs

The two previous examples showed the user initiating the logout request at a service provider. The logout process can, of course, also be initiated by the user visiting the IdP. Figure 22 illustrates this option:

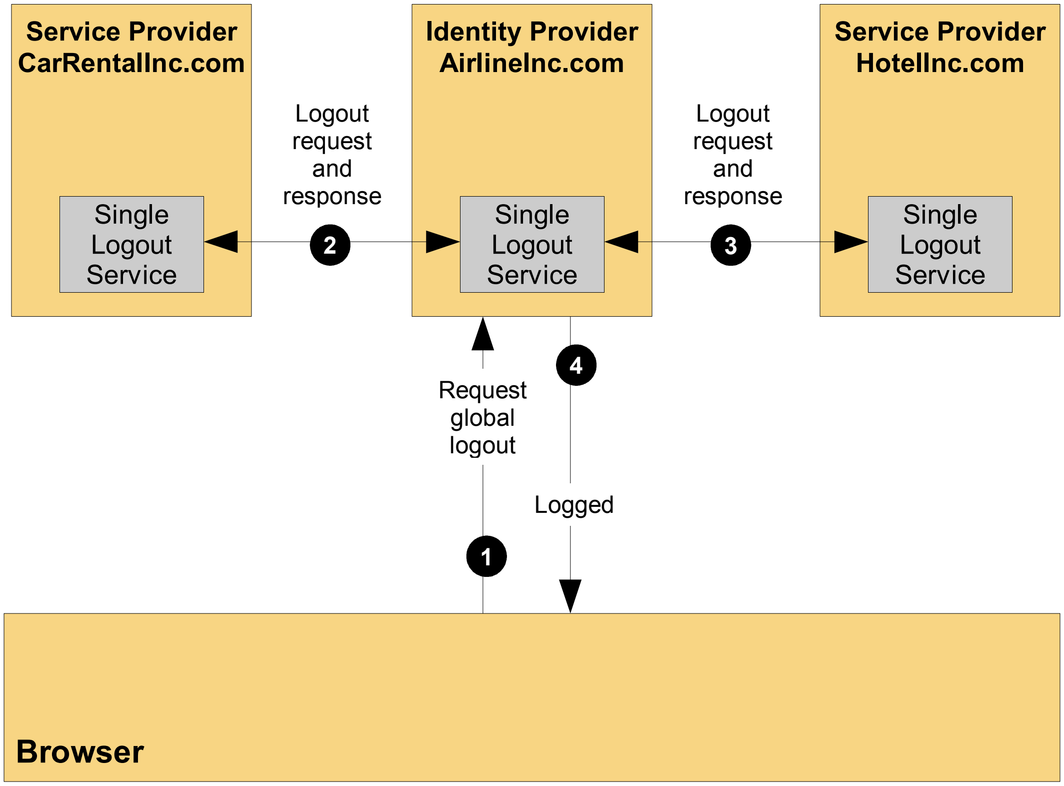

Service Provider CarRentalInc.com
Single Logout Service
Logout request and response
2
Identity Provider AirlineInc.com
Single Logout Service
Logout request and response
3
Service Provider HotelInc.com
Single Logout Service
Request global logout
1
4
Logged
Browser

Figure 22: IdP-Initiated Single Logout with Multiple SPs

4.4 Establishing and Managing Federated Identities

Thus far, the use case examples that have been presented have focused on the SAML message exchanges required facilitate the implementation of web single sign-on solutions. However, we have not yet examined issues surrounding how these message exchanges are tied to individual local and federated user identities shared between participants in the solution.

4.4.1 Introduction

This section describes mechanisms supported by SAML for establishing and managing federated identities. The following use cases are described:

- ***Federation via Out-of-Band Account Linking:*** The establishment of federated identities for users and the association of those identities to local user identities can be performed without the use of SAML protocols and assertions. This was the only style of federation supported by SAML V1 and is still supported in SAML v2.0.
- ***Federation via Persistent Pseudonym Identifiers:*** An identity provider federates the user's local identity principal with the principal's identity at the service provider using a persistent SAML name identifier.
- ***Federation via Transient Pseudonym Identifiers:*** A temporary identifier is used to federate between the IdP and the SP for the life of the user's web SSO session.
- ***Federation via Identity Attributes:*** Attributes of the principal, as defined by the identity provider, are used to link to the account used at the service provider.
- ***Federation Termination:*** termination of an existing federation.

To simplify the examples, not all possible SAML bindings are illustrated.

All the examples are based on the use case scenarios originally defined in Section 2.2, with AirlineInc.com being the identity provider.

4.4.2 Federation Using Out-of-Band Account Linking

In this example, shown in Figure 23, the user John has accounts on both AirlineInc.com and CarRentalInc.com each using the same local user ID (**john**). The identity data stores at both sites are synchronized by some out-of-band means, for example using database synchronization or off-line batch updates.

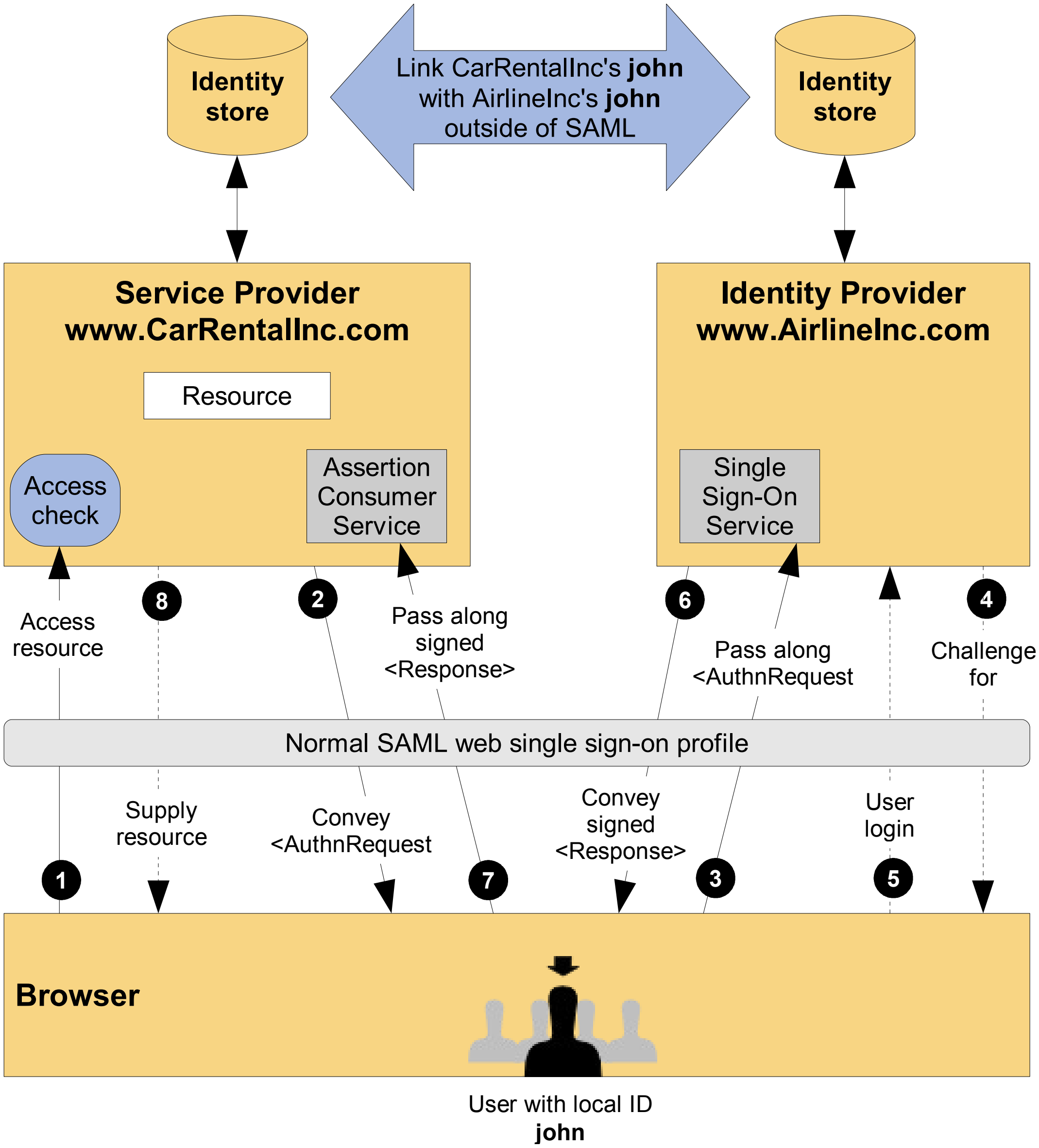

Identity store
Link CarRentalInc's john with AirlineInc's john outside of SAML
Identity store
Service Provider www.CarRentalInc.com
Resource
Access check
Assertion Consumer Service
Identity Provider www.AirlineInc.com
Single Sign-On Service
Access resource
8
2
Pass along signed <Response>
6
Pass along <AuthnRequest
4
Challenge for
Normal SAML web single sign-on profile
Supply resource
Convey <AuthnRequest
Convey signed <Response>
User login
1
7
3
5
Browser
User with local ID
john

Figure 23: Identity Federation with Out-of-Band Account Linking

The processing is as follows:

1. The user is challenged to supply their credentials to the site AirlineInc.com.
2. The user successfully provides their credentials and has a security context with the AirlineInc.com identity provider.

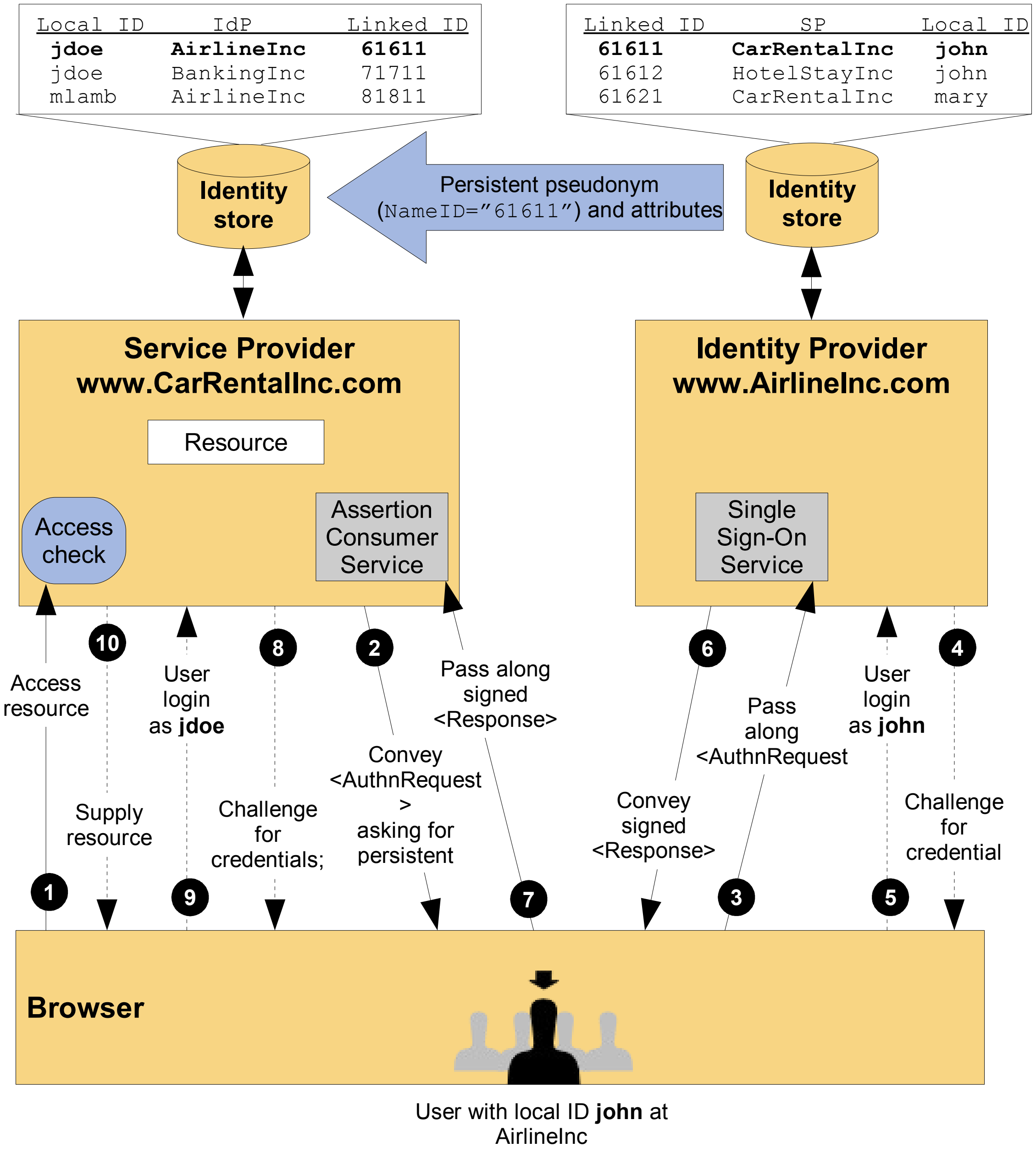
3. The user selects a menu option (or function) on the *AirlineInc.com* application that means the user wants to access a resource or application on *CarRentalInc.com*. The *AirlineInc.com* service provider sends a HTML form back to the browser. The HTML FORM contains a SAML response, within which is a SAML assertion about user john.
4. The browser, either due to a user action or via an "auto-submit", issues an HTTP POST containing the SAML response to be sent to the *CarRentalInc.com* Service provider.

The *CarRentalInc.com* service provider's Assertion Consumer Service validates the digital signature on the SAML Response. If this, and the assertion validate correctly it creates a local session for user john, based on the local john account. It then sends an HTTP redirect to the browser causing it to access the TARGET resource, with a cookie that identifies the local session. An access check is then made to establish whether the user john has the correct authorization to access the *CarRentalInc.com* web site and the TARGET resource. The TARGET resource is then returned to the browser. [@@rsp: TARGET is a SAML V1 concept and should not be used here. The IDP-initiated scenarios all rely on some out-of-band agreement on how to locate the desired application URL. This is done in some products via RelayState.]

4.4.3 Federation Using Persistent Pseudonym Identifiers

In this use case scenario, the partner sites take advantage of SAML V2.0's ability to dynamically establish a federated identity for a user as part of the web SSO message exchange. The user ***jdoe*** on *CarRentalInc.com* wishes to federate this account with his ***john*** account on the IdP, *AirlineInc.com*. Figure 24 illustrates dynamic identity federation using persistent pseudonym identifiers in an SP-initiated web SSO exchange.

[@@rsp: Show/discuss AllowCreate in AuthnRequest since we're doing dynamic federation.]



Local ID	IdP	Linked ID
jdoe	**AirlineInc**	**61611**
jdoe	BankingInc	71711
mlamb	AirlineInc	81811

Linked ID	SP	Local ID
61611	**CarRentalInc**	**john**
61612	HotelStayInc	john
61621	CarRentalInc	mary

Figure 24: SP-Initiated Identity Federation with Persistent Pseudonym

The processing is as follows:

1. The user attempts to access a resource on *CarRentalInc.com*. The user does not have any current logon session (i.e. security context) on this site, and is unknown to it. The resource that the user attempted to access is saved as `RelayState` information.
2. The service provider uses the HTTP Redirect Binding to send the user to the Single Sign-On Service at the identity provider (*AirlineInc.com*). The HTTP redirect includes a SAML `<AuthnRequest>` message requesting that the identity provider provide an assertion using a persistent name identifier for the user.
3. The user will be challenged to provide valid credentials.
4. The user provides valid credentials identifying himself as ***john*** and a local security context is created for the user at the IdP.

5. The Single Sign-On Service looks up user ***john*** in its identity store and creates a persistent name identifier (`61611`) to be used for the session at the service provider. It then builds a signed SAML web SSO assertion where the subject uses a transient name identifier format. The name ***john*** is not contained anywhere in the assertion. Note that depending on the partner agreements, the assertion might also contain an attribute statement describing identity attributes about the user (e.g. their membership level).
6. The browser, due either to a user action or execution of an "auto-submit" script, issues an HTTP POST request to send the form to the service provider's Assertion Consumer Service.
7. The *CarRentalInc.com* service provider's Assertion Consumer service validates the digital signature on the SAML Response and validates the SAML assertion. The supplied name identifier is then used to determine whether a previous federation has been established. If a previous federation has been established (because the name identifier maps to a local account) then go to step 9. If no federation exists for the persistent identifier in the assertion, then the SP needs to determine the local identity to which it should be assigned. The user will be challenged to provide local credentials at the SP. Optionally the user might first be asked whether he would like to federate the two accounts.
8. The user provides valid credentials and identifies his account at the SP as ***jdoe***. The persistent name identifier is then stored and registered with the ***jdoe*** account along with the name of the identity provider that created the name identifier.
9. A local logon session is created for user ***jdoe*** and an access check is then made to establish whether the user ***jdoe*** has the correct authorization to access the desired resource at the *CarRentalInc.com* web site (the resource URL was retrieved from state information identified by the `RelayState` information.
10. If the access check passes, the desired resource is returned to the browser.

4.4.4 Federation Using Transient Pseudonym Identifiers

The previous use case showed the use of persistent identifiers. So what if you do not want to establish a permanent federated identity between the parter sites? This is where the use of transient identifiers are useful. Transient identifiers allow you to:

- Completely avoid having to manage user ID's and passwords at the service provider.
- Have a scheme whereby the service provider does not have to manage specific user accounts, for instance it could be a site with a "group-like" access policy.
- Support a truly anonymous service

As with the Persistent Federation use cases, one can have SP and IdP-initiated variations. Figure 25 shows the SP-initiated use case using transient pseudonym name identifiers.

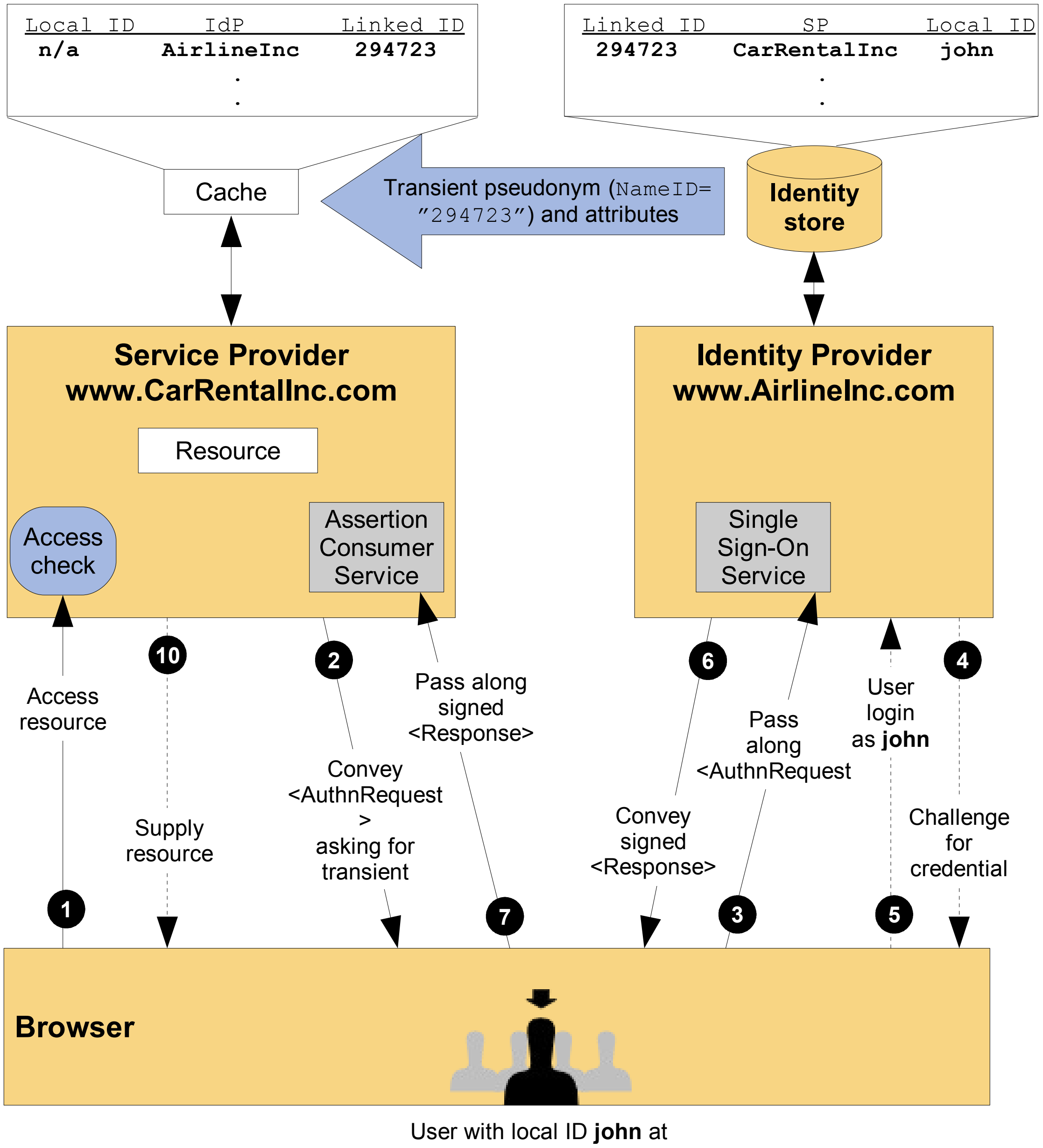


Figure 25: SP-Initiated Identity Federation with Transient Pseudonym

The processing is as follows:

1. The user attempts to access a resource on *CarRentalInc.com*. The user does not have any current logon session (i.e. security context) on this site, and is unknown to it. The resource that the user attempted to access is saved as `RelayState` information.
2. The service provider uses the HTTP Redirect Binding to send the user to the Single Sign-On Service at the identity provider (*AirlineInc.com*). The HTTP redirect includes a SAML `<AuthnRequest>` message requesting that the identity provider provide an assertion using a transient name identifier for the user.
3. The user will be challenged to provide valid credentials at the identity provider.
4. The user provides valid credentials identifying himself as ***john*** and a local security context is created for the user at the IdP.
5. The Single Sign-On Service looks up user ***john*** in its identity store and creates a transient name

identifier (294723) to be used for the session at the service provider. It then builds a signed SAML web SSO assertion where the subject uses a transient name identifier format. The name ***john*** is not contained anywhere in the assertion. The assertion also contains an attribute statement with a membership number attribute (1357) provided [@@rsp: this isn't typical for the transient case, is it? Why would an IdP be holding the member numbers of users at an SP. A more typical scneario (at least to me) would perhaps send an attribute such as “member level” for which many users might have the same value. That gives the user access to the generic service level at the SP without specifically identifying them]. The assertion is placed in a SAML response message and the IdP uses the HTTP POST Binding to send the Response message to the service provider.

6. The browser, due either to a user action or execution of an “auto-submit” script, issues an HTTP POST request to send the form to the service provider's Assertion Consumer Service.
7. The *CarRentalInc.com* service provider's Assertion Consumer service validates the SAML Response and SAML assertion. The supplied transient name identifier is then used to dynamically create a session for the user at the SP. The member number attribute [@@rsp: membership level?] might be used to perform an access check on the requested resource and customize the content provided to the user.
8. If the access check passes, the requested resource is then returned to the browser.

While not shown in the diagram, the transient identifier remains active for the life of the user authentication session. If needed, the SP could use the identifier to make SAML attribute queries back to an attribute authority at *AirlineInc.com* to obtain other identity attributes about the user in order to customize their service provider content, etc.

4.4.5 Federation Using Identity Attributes

Attribute Federation is when the identity provider sends an assertion to the service provider where the supplied NameID is not used to map or create a session on the SP, rather an attribute (or possibly several attributes) are used to define the account to be used. This scenario is shown in Figure 26.

[@@rsp: I haven't reviewed this example in detail. Add a high-level use case attribute federation figure and explanation here, based on original Figure 1, but with attribute aspect emphasized and with details changed to match figure nn?]

In this example the processing is as follows: @@change joe->john in figure

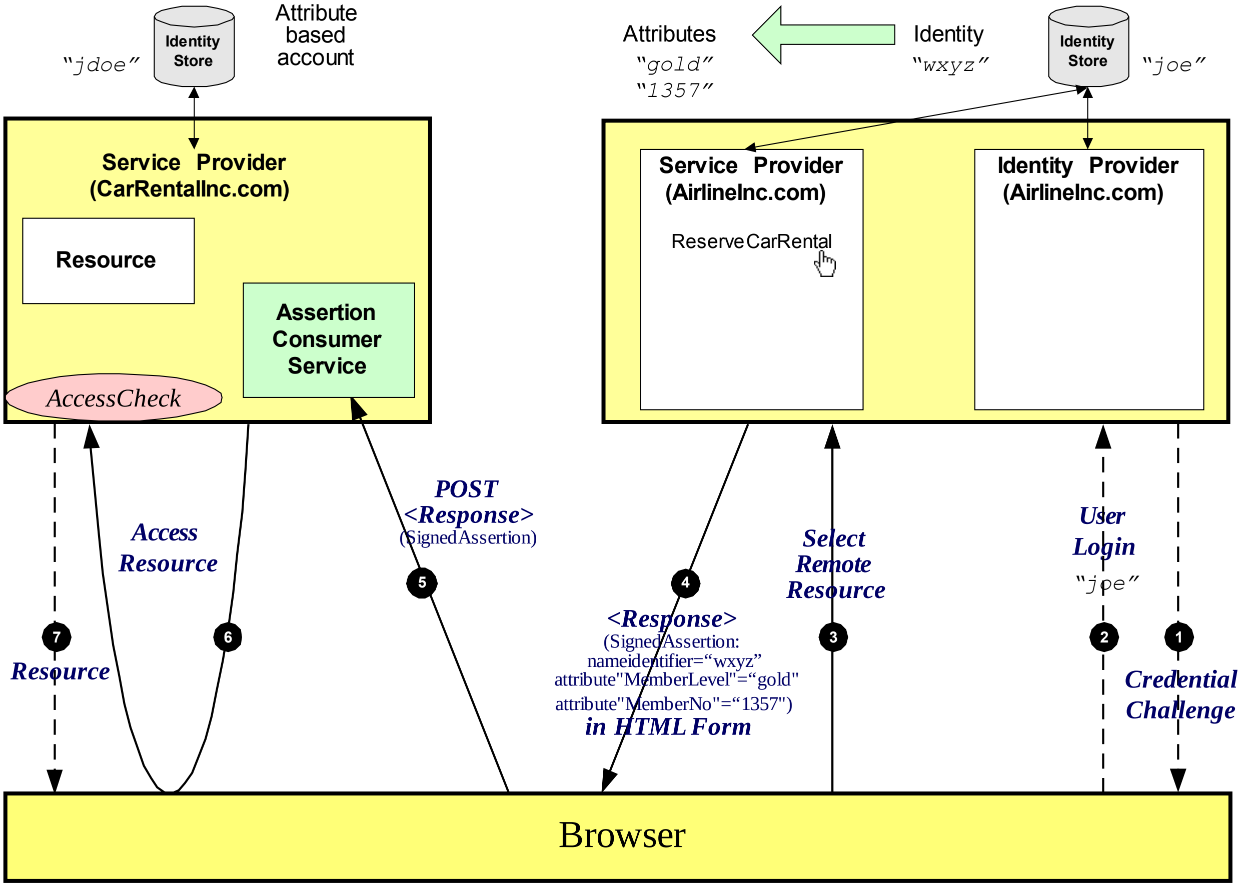


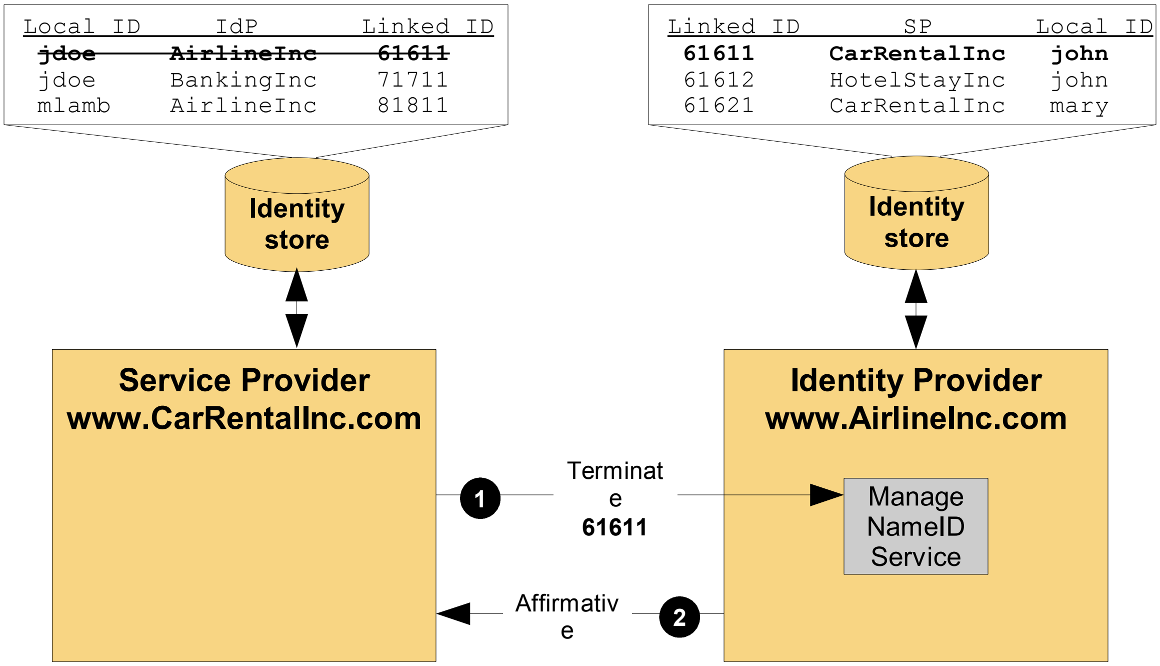
Figure 26: Identity Federation: Identity Attributes

1. The user is challenged to supply their credentials to the site *AirlineInc.com*.
2. The user successfully provides their credentials and has a security context with the *AirlineInc.com* identity provider, the user named supplied is ***john***.
3. The user selects a menu option (or function) on the *AirlineInc.com* application that means the user wants to access a resource or application on *CarRentalInc.com*.
4. The *AirlineInc.com* service provider sends a HTML form back to the browser. The HTML FORM contains a SAML response, within which is a SAML assertion about user ***john***. The name identifier used in the assertion is an arbitrary value ("wxyz"). The attributes "gold member" and a membership number attribute ("1357") are provided. The name ***john*** is not contained anywhere in the assertion.
5. The browser, either due to a user action or via an "auto-submit", issues an HTTP POST containing the SAML response to be sent to the *CarRentalInc.com* Service provider.
6. The *CarRentalInc.com* service provider's Assertion Consumer service validates the digital signature on the SAML Response. If this, and the assertion validate correctly it creates a local session. The session created is for user ***jdoe.*** It determines this from a combination of the gold member and membership number attributes. It then sends an HTTP redirect to the browser causing it to access the TARGET resource, with a cookie that identifies the local session. An access check is then made to establish whether the user ***jdoe*** has the correct authorization to access the *CarRentalInc.com* web site and the TARGET resource. If the access check passes, the TARGET resource is then returned to the browser.

4.4.6 Federation Termination

This example builds upon the previous example and shows how a federation can be terminated. In this case the ***jdoe*** account on *CarRentalInc.com* service provider has been deleted, hence it wishes to terminate the federation with *AirlineInc.com* for this user.

The Terminate request is sent to the identity provider using the Name Identifier Management Protocol, specifically using the `<ManageNameIDRequest>`. The example shown in Figure 27 uses the SOAP over HTTP binding which demonstrates a use of the back channel. Bindings are also defined that permit the request (and response) to be sent via the browser using asynchronous "front-channel" bindings, such as the HTTP Redirect, HTTP POST, or Artifact bindings.



Local ID	IdP	Linked ID
~~jdoe~~	~~AirlineInc~~	~~61611~~
jdoe	BankingInc	71711
mlamb	AirlineInc	81811

Linked ID	SP	Local ID
61611	**CarRentalInc**	**john**
61612	HotelStayInc	john
61621	CarRentalInc	mary

Figure 27: Identity Federation Termination

In this example the processing is as follows:

1. The service provider, *CarRentalInc.com,* determines that the local account, j***doe***, should no longer be federated. An example of this could be that the account has been deleted. The service provider sends to the *AirlineInc.com* identity provider a `<ManageIDNameRequest>` defining that the persistent identifier (previously established) must no longer be used. The request is carried in a SOAP message which is transported using HTTP, as defined by the SAML SOAP binding. The request is also digitally signed by the service provider.
2. The identity provider verifies the digital signature ensuring that the `<ManageIDNameRequest>` originated from a known and trusted service provider. The identity Provider processes the request and returns a `<ManageIDNameResponse>` containing a suitable status code response. The response is carried within a SOAP over HTTP message and is digitally signed.

4.5 Use of Attributes

As explained in Section 2.2, in describing the web single sign-on use case, the SAML assertion transferred from an identity provider to a service provider may include attributes describing the user. The ability to transfer attributes within an assertion is a powerful SAML feature and it may also be combined with the forms of identity federation described above.

The following are some typical use patterns:

- Transfer of profile information

  Attributes may be used to convey user profile information from the identity provider to the service provider.This information may be used to provide personalized services at the service provider, or to augment or even create a new account for the user at the service provider. The user should be informed about the transfer of information, and, if required, user consent explicitly obtained.

- Authorization based on attributes

  In this model, the attributes provided in the SAML assertion by the identity provider are used to authorize specific services at the service provider. The service provider and identity provider need prior agreement (out of band) on the attribute names and values included in the SAML assertion. An interesting use of this pattern which preserves user anonymity but allows for differential classes of service is found in Shibboleth [ShibReqs]: federation using transient pseudonyms combined with authorization based on attributes.

5 Comparison Between SAML V2.0 and SAML V1.1

SAML V2.0 represents a significant feature upgrade to SAML V1.1. The enhancements include features derived from the Liberty Alliance Identity Federation Framework (ID-FF) V1.2 specifications that were contributed to the SSTC in 2003, capabilities present in the Internet2's Shibboleth architecture, and enhancement requests resulting from experience with numerous deployments of SAML V1.x in the industry.

The on-the-wire representations of SAML V2.0 assertions and protocol messages are incompatible with SAML V1.x processors. As is explained in the SAML Assertions and Protocols specification [SAMLCore], only new major versions of SAML (of which this is one) typically cause this sort of incompatibility. In this release, much of the incompatibility is syntactic in nature; this was done for consistency and better component symmetry.

5.1 Specification Organization Changes

- The conformance specification now explicitly serves as the entry point for the SAML V2.0 OASIS Standard specifications.
- The assertion and protocol ("core") specification is now referred to as Assertion***s*** and Protocol***s***, specification since it now defines multiple protocols.
- Processing rules are now clearly called out in each protocol.
- The single "bindings and profiles" specification has been split into two documents, one for bindings and one for profiles, and the latter now includes "SAML attribute profiles".
- There is a new authentication context specification and several accompanying XML schemas.
- There is a new metadata specification and an accompanying XML schema.
- Bibliographic references have been divided into normative and non-normative categories.
- There is a new non-normative executive overview document and this new technical overview document.

5.2 General Changes

- The SAML assertions namespace (known by its conventional prefix `saml:`) and protocols namespace (known by its conventional prefix `samlp:`) now contain the string "2.0" in recognition of this new major version of SAML.
- The `MajorVersion` and `MinorVersion` attributes that appeared on various elements have been combined into a single `Version` attribute that has the value "2.0".
- The terminology used to describe various SAML system entities has been rationalized and enhanced to incorporate the Liberty Alliance notion of "identity providers" as opposed to "authentication authorities" and similar.
- The SAML schema extensibility mechanisms have been rationalized and, in some cases, enhanced. XSD element substitution has been blocked in favor of type extension. The `<xs:anyAttribute>` wildcard has been added selectively to structures where it has been deemed valuable to add arbitrary "foreign" attributes without having to create a schema extension; these structures include subject confirmation data and SAML attributes.
- The authorization decision feature (statement and query) has been frozen; if more functionality is desired, it is recommended that XACML [XACML] be used.
- A series of changes that were pre-announced during the SAML V1.x design cycles have been made:

  - The deprecated <AuthorityBinding> element has been removed.
  - The deprecated <RespondWith> element has been removed.
  - The deprecated name identifier and artifact URI-based identifiers [@@does this mean name ID formats?] have been removed.
  - URI references are now required to be absolute.
  - The description of appearance of the <Status> element in SOAP messages has been improved.
- TBS: validity period semantics and syntax extended, removal of QNames in content, etc.

5.3 XML Signature and XML Encryption Support

- The <ds:Signature> element that allows for the digital signing of assertions and protocol messages has been positioned earlier in the respective content models.
- SAML now supports the use of the W3C XML Encryption recommendation [XMLEnc] to satisfy privacy requirements for several important SAML constructs.
- A new <EncryptedID> element has been defined that can hold an encrypted SAML identifier. These identifiers can be encrypted <NameID> or <Assertion> elements or elements of types derived from **NameIDType, AssertionType,** or **BaseIDAbstractType**.
- A new <EncryptedAssertion> element has been defined that can hold an encrypted SAML assertion.
- A new <EncryptedAttribute> element has been defined that can hold an encrypted SAML attribute.

5.4 Name Identifier, Subject, and Subject Confirmation Changes

- The new **BaseID** complex type is an extension point used to create new types of SAML identifiers.
- Name identifiers have new attributes permitting both IdP-specific and SP-specific qualification.
- Persistent and transient name identifier formats have been introduced that utilize pseudonyms to provide privacy-preserving characteristics for federated SAML identities.
- The <SubjectConfirmation> element is now repeatable, with the formerly repeatable <ConfirmationMethod> element was renamed to Method and placed as an attribute within the <SubjectConfirmation>.
- A set of generic attributes in <SubjectConfirmationData> have been defined for use in constraining the confirmation information. Overall assertion validity is more flexible within profiles as a result.
- A <SubjectConfirmationData> element now permits the inclusion of arbitrary XML attributes and child elements.
- A new **KeyInfoConfirmationDataType** complex type is used to constrain a <SubjectConfirmationData> element to hold <ds:KeyInfo> elements. Further, the usage of <ds:KeyInfo> within <SubjectConformationData> has been clarified to more clearly allow for impersonation.

5.5 General Assertion Changes

- The AssertionID attribute has been replaced by a general XML ID attribute.
- The Issuer attribute has been replaced by the <Issuer> element allowing the use of a

generalized name identifier.

- The <Subject> element has been moved up to be a child of the <Assertion> element rather than appearing as a child of a <SubjectStatement> element. All statements of the assertion must apply to the specified <Subject> element. The <Subject> element is now optional for extensibility reasons, although it is required for all assertions with SAML-specified statement types.
- The <SubjectStatement> element and its type have been removed.
- The <Conditions> element has been extended and restructured to permit more flexible conditions to be defined.
- The <DoNotCacheCondition> element has been replaced by a <OneTimeUse> element as a child of a <Conditions> element. The relationship of this condition to the NotBefore and NotOnOrAfter conditions has been delineated.
- A new <ProxyRestriction> element has been defined as a child of a <Conditions> element.

5.6 Authentication Statement Changes

- The <AuthenticationStatement> element has been renamed to <AuthnStatement>.
- The <AuthnStatement> element now supports the concept of a session in support of single logout and other session management requirements.
- The AuthenticationMethod attribute has been replaced by the new structured <AuthnContext> element permitting the expression of new, very fine-grained authentication methods.

5.7 Attribute Statement Changes

- The <AttributeStatement> element can now hold both encrypted and unencrypted SAML attributes.
- The name of the AttributeName field has been changed to just Name.
- The AttributeNamespace field has been removed in favor of NameFormat, and two new URI-based identifiers for attribute name format types have been defined for use in this field. This field can be left blank, as a default has been defined.
- Arbitrary XML attributes can now appear on the <Attribute> element without a supporting extension schema.
- Clearer instructions have been provided for how to represent null and multi-valued attributes.
- A series of attribute profiles has now been defined. They provide for proper interpretation of SAML attributes specified using common attribute/directory technologies.

5.8 General Request-Response Protocol Changes

- The RequestID and ResponseID attributes have been replaced by general XML ID attributes.
- The request datatype hierarchy has been reorganized; all queries are now kinds of requests, not inside requests, and the plain <Query> has been removed.
- Consent and <Extensions> constructs have been added to all requests and responses.
- An <Issuer> element can now be present on requests and responses (in addition to appearing on assertions).

- The response type hierarchy has been reorganized; most response elements in the various protocols are simply of **StatusResponseType**.
- New status codes have been added to reflect possible status values for the new protocols. Status codes are now URIs instead of Qnames.
- The `<AssertionIDRequest>` element is now used to obtain an assertion by means of its ID instead of using a `<Request>` with an `<AssertionIDReference>` element.
- SAML artifacts can no longer be used to refer to specific SAML assertions to be exchanged as described in the SAML v1 Browser/Artifact Profile. Artifacts are now used only to refer to SAML protocol messages. Once in possession of an artifact from a partner, an entity can retrieve the actual message from the partner through use of the new SAML Artifact Resolution Protocol. All types of protocol messages can theoretically be retrieved in this fashion.

5.9 Changes to SAML Queries

- An authentication query now supports the concept of sessions.
- In an authentication query, the `AuthenticationMethod` attribute has been replaced by the new structured `<AuthnContext>` element permitting queries for the new, very fine-grained authentication methods.
- In an attribute query, semantics have been defined to support the specification of attribute values as part of the query to limit the set of attribute values which may be returned.

5.10 New SAML Protocols

- The Authentication Request Protocol provides support for SP-initiated web SSO exchanges. This protocol allows the SP to make requests to an IdP and potentially control various aspects of the user authentication at the IdP, the binding to be used to return the response message, the set of SAML attributes to be included in the resulting assertion, etc. As part of this request, the SP can also indicate the desire to dynamically establish a new federated identity for the user.
- The Single Logout Protocol supports near-simultaneous logout of sessions at web SSO participants.
- The Artifact Resolution Protocol is used to retrieve SAML protocol messages through an artifact reference.
- The NameID Management Protocol provides the ability to modify federated name identifiers or to terminate their use.
- The NameID Mapping Protocol allows an SP that shares an identifier for a principal with an IdP to obtain a name identifier for the same principal in another format or that is in another federation namespace (i.e. Is shared between the IdP and another SP.

5.11 Bindings Changes

- Generalized bindings have been created to support protocol message transfer between SAML parties using HTTP via a user agent (e.g. A browser). These bindings are known as the HTTP Redirect and the HTTP POST bindings.
- The HTTP Artifact Binding describes the means by which a SAML artifact can be transferred from one party to another. Once in possession of an artifact, an entity utilizes the SAML Artifact Resolution Protocol to retrieve the referenced protocol message.
- A PAOS (reverse SOAP) binding has been added.
- A set of mechanisms for relaying state have been added to most of the bindings.

- There is a new HTTP-based binding added for retrieval of assertions by means of URIs.

5.12 Profiles Changes

- A great deal of binding-specific detail has been factored out of the profiles. The resulting profiles are much shorter.
- The two original web browser profiles (Browser/Artifact and Browser/POST) have been consolidated into a single web browser SSO profile.
- An enhanced client and proxy (ECP) SSO profile has been added.
- An Identity Provider Discovery Profile has been added that relies on the technique of creating common domain cookies.
- The new Artifact Resolution Profile describes how the Artifact Resolution Protocol is specifically used with the SOAP over HTTP Binding to retrieve SAML protocol messages referred to by an artifact.
- The new Name Identifier Mapping Profile describes how the Name Identifier Mapping Protocol is specifically used with the SOAP over HTTP Binding.
- As noted earlier, a series of attribute profiles has now been defined.

6 Comparison Between SAML V2.0 and Liberty ID-FF V1.2

SAML V2.0 represents a significant feature upgrade to SAML V1.1. The enhancements include features derived from the Liberty Alliance Identity Federation Framework (ID-FF) V1.2 specifications that were contributed to the SSTC in 2003, capabilities present in the Internet2's Shibboleth architecture, and enhancement requests resulting from experience with numerous deployments of SAML V1.x in the industry.

The on-the-wire representations of SAML V2.0 assertions and protocol messages are incompatible with Liberty ID-FF processors. As is explained in the SAML Assertions and Protocols specification [SAMLCore], only new major versions of SAML (of which this is one) typically cause this sort of incompatibility. In this release, much of the incompatibility is syntactic in nature; this was done for consistency and better component symmetry.

The following sections analyze the differences between SAML V2.0 and ID-FF V1.2 and provide guidance and other commentary, in order to aid developers and deployers who are undergoing an upgrade or need to support multiple versions at once. The analysis results are stated in the following (somewhat subjective) terms, which are not mutually exclusive:

- **Same:** SAML's approach is largely identical to Liberty's approach, including close similarity in specification text and even syntax to a large degree (though it cannot be assumed to be identical; at the very least, the markup resides in a different namespace).
- **Equivalent:** SAML's approach is functionally equivalent, even if achieved in a different manner structurally.
- **More functional:** SAML has generalized the Liberty functionality to account for more options or use cases.
- **Different:** SAML has significant structural differences from Liberty due to the refactoring activity done as part of the design and convergence effort for SAML V2.0.

6.1 Representation of Principals

6.1.1 <Subject> and <NameID>

Different, More Functional

The `<NameID>` element has been substantially enhanced to combine the information carried in SAML V1.1, Shibboleth, and the extensions added to `<saml:Subject>` in ID-FF. As in ID-FF and Shibboleth, specific `Format` values are used to connote identifiers designed with privacy-preserving properties in mind. Both persistent and so-called transient identifiers can be used, corresponding to the ID-FF federated and onetime identifiers, respectively.

ID-FF overloaded all non-transient identifiers into a single `Format` value, regardless of their privacy characteristics. For example, an employee ID number might be used instead of a pseudonym. SAML reserves the use of the persistent `Format` URN for pseudonyms having the pair-wise characteristics of ID-FF federated identifiers. Another URN MUST be used when violating the privacy or pair-wise semantics. Deployers can choose specific kinds of identifiers and enable and disable their use as needed.

When using any form of identifier (whether privacy-preserving or not), SAML V2.0 also incorporates the ability from ID-FF to qualify the identifier both in terms of the asserting party and the relying party by adding an `SPNameQualifier` to the original `NameQualifer` attribute. Also, any identifier can carry a second string identifier established by the relying party as an alias, termed the `SPProvidedID`. This eliminates the two-part subject structure created in ID-FF.

Commentary:

- Implementation or deployment guidance should specify use of the privacy-preserving formats when privacy is an issue.
- As with ID-FF V1.2, attention will be needed to address interoperation of SAML V2.0 with ID-FF V1.1 and V1.2 in terms of representing a single principal in all three message types. However, a direct mapping from ID-FF V1.2 to SAML V2.0 should be possible in most cases.

6.1.2 Encrypted Identifiers

Different, Equivalent

ID-FF defined a mechanism to hide encrypted identifiers inside standard ID-FF identifiers by encoding the XML Encryption content. SAML V2.0 permits direct use of XML Encryption in various places, including an `<EncryptedID>` element that can replace the usual `<NameID>` element. ID-FF's confusing rules for using `NameQualifier` in different ways in the encrypted case are gone.

6.2 Single Sign-On Profiles

6.2.1 <AuthnRequest>

Different, More Functional

The `<AuthnRequest>` protocol message in SAML is somewhat revised, but is a superset of the ID-FF message. The main difference relevant to ID-FF use cases is a revised `<NameIDPolicy>` element that addresses the ability to request specific principal representations. The other enhancements are intended for advanced use cases in the future in which assertions need to be tailored by the relying party for their intended use by including additional subject confirmations or conditions.

Commentary:

- Some of the advanced capabilities supported by the `<AuthnRequest>` message are permitted in the SAML SSO profiles, such as the ability to specify arbitrary `<Conditions>` in the assertion, that were not permitted in ID-FF.

6.2.2 Browser SSO

Different, Equivalent

All of the existing SAML V1.1 and ID-FF profiles for browser SSO have been merged into a single basic profile with support for different bindings. The general characteristics of the profile align well with ID-FF assumptions. For example, when using the new POST binding, the assertion is signed, rather than the response (a change to SAML, but not to ID-FF). All of the device accommodations in the ID-FF profiles are captured in the defined Redirect, POST, and Artifact bindings.

Commentary:

- You should check on the level of support for your required bindings among your identity federation partners.

6.2.3 Enhanced Client SSO

Different, Equivalent

The LECP use case in ID-FF is rended in a redesigned profile called ECP that uses SOAP and PAOS. It is functionally the same, but uses SOAP and SOAP header blocks to carry the information the ID-FF profile places inside custom XML envelopes. The most significant difference is that the interaction with the SP is via PAOS and not POST. This is a change, but an ID-FF SP could not support LECP before without explicit changes anyway.

6.2.4 Proxying

Same

Much of the specification language about proxying SSO is very similar to ID-FF's text. Some additional policy controls on proxying are supported in SAML, but the overall approach is about the same. Note that the reliance on the Authentication Context specification to carry the list of providers is removed.

6.3 Single Logout Profiles

Same

Logout is largely unchanged from ID-FF. All of the Liberty-generated errata around logout request expiration, `SessionIndex`, proxy failure, and so on have been incorporated into SAML. The distinct profiles have been combined into a single binding-independent profile, but the overall functionality is basically the same.

6.4 Name Identifier Registration and Federation Termination Profiles

Different, More Functional

SAML V2.0 combines the two protocols in ID-FF V1.2 into a single protocol for updating or terminating use of identifiers. Any kind of identifier (as opposed to just the federated variety) can be updated or terminated. The distinct profiles have been combined into a single binding-independent profile, but the overall functionality is basically the same.

6.5 Name Identifier Mapping

Different, More Functional

SAML V2.0 generalizes the mapping protocol by using the `<NameIDPolicy>` element to describe the properties of the identifier to be returned. This allows for arbitrary mappings between any two formats, even allowing “create” or “don't create” semantics that match the behavior during SSO. It's therefore possible to programmatically “federate” a principal explicitly using this protocol.

6.6 URL Encoding of Messages

Different, Equivalent

The per-message piecemeal encoding of XML into URL parameters was replaced in SAML V2.0 with a scheme based on compressing the actual XML with the DEFLATE algorithm (used by gzip). Signing is still permitted in a similar fashion, with somewhat simpler processing rules.

6.7 Metadata

Different, More Functional

SAML V2.0 metadata is substantially different in format from ID-FF, but is a functional superset in most respects. Support for SAML profiles unsupported by ID-FF are captured, such as SAML attribute exchange and authorization decisions. Problems with schema consistency, encryption metadata, and multi-endpoint support for a single protocol have been corrected. In the common case, a single SOAP endpoint might be duplicated in many metadata elements if it is used for many profiles, so a "space for clarity" trade-off was chosen to support cases in which the endpoint is not shared.

Commentary:

- All language about use of DNS, zone signatures, etc. is tightly bound to the DNS-based exchange profile, and it is not called out as a preferred or primary means of exchange. Check to see if it is supported in your chosen implementation.
- Since Liberty ID-WSF carries some of its bootstrapping information in SAML attributes, use of

SAML attribute-oriented metadata might be useful when integrating SAML deployments with ID-WSF

6.8 Authentication Context

Different, More Functional

Though mostly intact from ID-FF, some key changes include the removal of SAML `AuthenticationMethod` in favor of the `<AuthnContext>` element; support for including only context classes in an assertion, and omitting specific context declarations (formerly statements); and the ability to determine class conformance by a declaration instance using schema validation, enabling machine-processable conformance checking.

7 References

[SAMLAuthnCxt] J. Kemp et al. *Authentication Context for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-authn-context-2.0-os. See http://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf.

[SAMLBind] 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.

[SAMLConform] P. Mishra et al. *Conformance Requirements for the OASIS Security Assertion Markup Language (SAML) V2.0.* OASIS SSTC, March 2005. Document ID saml-conformance-2.0-os. See http://docs.oasis-open.org/security/saml/v2.0/saml-conformance-2.0-os.pdf.

[SAMLCore] 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.

[SAMLErrata] 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/.

[SAMLExecOvr] 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.oasis-open.org/committees/security/.

[SAMLGloss] 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.

[SAMLMDExtQ] 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/.

[SAMLMDV1x] G. Whitehead et al. *Metadata Profile for the OASIS Security Assertion Markup Language (SAML) V1.x.* OASIS SSTC, March 2005. Document ID sstc-saml1x-metadata-cd-01. See http://www.oasis-open.org/committees/security/.

[SAMLMeta] 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.

[SAMLProf] 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.

[SAMLProt3P] 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.oasis-open.org/committees/security/.

[SAMLSec] 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-sec-consider-2.0-os. See http://docs.oasis-open.org/security/saml/v2.0/saml-sec-consider-2.0-os.pdf.

[SAMLWeb] OASIS Security Services Technical Committee web site, http://www.oasis-open.org/committees/security.

[SAMLX509Attr] R. Randall et al. *SAML Attribute Sharing Profile for X.509 Authentication-Based Systems.* OASIS SSTC, March 2006. Document ID sstc-saml-x509-authn-attrib-profile-cd-02. See http://www.oasis-open.org/committees/security/.

[SAMLXPathAttr] C. Morris et al. *SAML XPath Attribute Profile.* OASIS SSTC, August, 2005. Document ID sstc-saml-xpath-attribute-profile-cd-01. See http://www.oasis-open.org/committees/security/.

[ShibReqs] S. Carmody. *Shibboleth Overview and Requirements*. Shibboleth project of Internet2. See http://shibboleth.internet2.edu/docs/draft-internet2-shibboleth-requirements-01.html.

[WSS] 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/.

[WSSSAML] R. Monzillo et al. *Web Services Security: SAML Token Profile 1.1.* OASIS WSS-TC, February 2006. Document ID wss-v1.1-spec-os-SAMLTokenProfile. See http://www.oasis-open.org/committees/wss/.

[XACML] T. Moses, et al. *OASIS eXtensible Access Control Markup Language (XACML) Version 2.0*. OASIS XACML-TC, February 2005. Document ID oasis-access_control-xacml-2.0-core-spec-os. See http://www.oasis-open.org/committees/xacml.

[XMLEnc] D. Eastlake et al. *XML Encryption Syntax and Processing.* World Wide Web Consortium. See http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/.

A. Acknowledgments

The editors would like to acknowledge the contributions of the OASIS Security Services Technical Committee, whose voting members at the time of publication were:

- TBD

B. Revision History

Rev	Date	By Whom	What
00	Nov 6, 2003	John Hughes	Storyboard version
01	Jul 22, 2004	John Hughes	First draft
02	27 Sept 2004	John Hughes	Second Draft.General updates, limited distribution
03	Feb 20, 2005	John Hughes	DCE/Kerberos use section removed. Use of SAML in other frameworks added. SAML V2.0 XML examples included. Updated Web SSO examples to remove use of ITS
04	10 Apr 2005	Eve Maler	Edits based on comments made by myself and Scott Cantor. Fleshed out the list of 1.1->2.0 differences, but it's not complete yet. More work to come.
05	May 10, 2005	Prateek Mishra	Updated Section 2 and 3.4, Section 4.3 remains incomplete
06	Jun 3, 2005	John Hughes	Added Section 4.3 plus a few minor corrections
07	Jul 13, 2005	John Hughes	Addressed comments from SSTC, primarily re-vamping section 4.3
08	12 Sep 2005	Eve Maler	Incorporated many, though not all, of the comments that arose from the special Tech Overview review meeting (see notes sent to the SSTC list on 24 August 2005)
09	20 July 2006	Rob Philpott, Eve Maler	Major updates – reorganize material; remove misconception re: meaning of a federated identity; update doc roadmap; removed use case redundancy; updated V1-V2 differences; revised graphics; etc.
10	9 Oct 2006	Eve Maler	Added new ID-FF comparison section (closely modeled after one Scott wrote). Made Prateek's suggested changes about "use of attributes". A few other cleanup items.

C. Notices

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification, can be obtained from the OASIS Executive Director.

OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to implement this specification. Please address the information to the OASIS Executive Director.