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

Working Draft 08, 12 September 2005

Document identifier:
sstc-saml-tech-overview-2.0-draft-08

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

Editors:
John Hughes, Individual
Eve Maler, Sun Microsystems

Contributors:
Hal Lockhart, BEA
Thomas Wisniewski, Entrust
Prateek Mishra, Oracle
Nick Ragouzis, Individual

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 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

1 Introduction 5
2 SAML Use Cases 6
2.1 Single Sign-On Use Case 6
2.2 Identity Federation Use Cases 7
3 SAML Architecture 8
3.1 Basic Concepts 8
3.2 Summary of SAML Components 8
3.3 SAML Constructs and Examples 10
3.3.1 The Relationship of Bindings, Protocol Messages, Assertions, and Statements 11
3.3.2 Assertion, Statement, and Subject Structure 11
3.3.3 Attribute Statement Structure 12
3.3.4 Message Structure and the SOAP Binding 13
3.4 Use of SAML in other Frameworks 14
3.4.1 Web Services Security (WS-Security) 15
3.4.2 eXtensible Access Control Markup Language (XACML) 16
3.5 Security in SAML 18
4 Profiles 19
4.1 Web Browser SSO Profile 19
4.1.1 SP initiated: POST->POST binding 20
4.1.2 SP initiated: Redirect->POST binding 22
4.1.3 SP initiated: Artifact->POST binding 23
4.1.4 SP initiated: POST->Artifact binding 24
4.1.5 SP initiated: Redirect->Artifact binding 26
4.1.6 SP initiated: Artifact->Artifact binding 27
4.1.7 IdP initiated: POST binding 29
4.1.8 IdP initiated: Artifact binding 30
4.2 ECP Profile 31
4.2.1 Introduction 31
4.2.2 ECP Profile using PAOS binding 32
4.3 Identity Federation Protocols 32
4.3.1 Introduction 32
4.3.2 Single Sign-On with Out-of-band Account Linking 33
4.3.3 Attribute Federation 34
4.3.4 Persistent Federation 35
4.3.5 Transient Federation 38
4.3.6 Federation Termination 40
4.4 Single Logout 41
5 Documentation roadmap 44
6 Comparison Between SAML V2.0 and SAML V1.1 45
6.1 Differences in the Organization of the Specifications 45
6.2 Versioning Differences 45
6.3 Subject and Subject Confirmation Differences 45
6.4 Encryption-Related Differences 46

6.5 Attribute-Related Differences 46
6.6 Differences in the Request-Response Mechanism 46
6.7 Differences in the Protocols for Retrieving Assertions 46
6.8 Session-Related Differences 47
6.9 Federation-Related Differences 47
6.10 Differences in Bindings and Profiles 47
6.11 Other Differences 47
7 References 48

Table of Figures

Figure 1: Single Sign-On Use Case [@@remove attr aspect] 6
Figure 2: Account Linking Use Case [@@replace missing figure] 7
Figure 3: SAML Components 10
Figure 4: SAML Assertion Containment Overview 11
Figure 5: Example of SAML Assertion, Subject, and Authentication Statement Constructs 12
Figure @@nn: Example of Attribute Statement 13
Figure 6: SAML Protocol Message Conveyed with SOAP over HTTP 13
Figure 7: Example of SAML Authentication Request Conveyed by SOAP 14
Figure 8: Example of SAML Response 14
Figure 10: Typical use of WS-Security and SAML [@@add figure back!] 16
Figure 11: Typical use of XACML and SAML 17
Figure 12: Push and Pull Models for Web Browser SSO Profile 20
Figure 14: SP initiated: POST->POST binding 21
Figure 15: SP initiated: Redirect->POST binding 22
Figure 16: SP initiated: Artifact->POST binding 23
Figure 17: SP initiated: POST->Artifact binding 25
Figure 18: SP initiated: Redirect->Artifact binding 26
Figure 19: SP initiated: Artifact->Artifact binding 28
Figure 20: IdP initiated: POST binding 29
Figure 21: IdP initiated: Artifact binding 31
Figure 22: ECP use cases 32
Figure 23: ECP with PAOS 32
Figure 24: Single Sign-On with Out-of Band account linking 33
Figure 25: Attribute Federation 35
Figure 26: Persistent Federation – SP-initiated 36
Figure 27: Persistent Federation – IdP-initiated 37
Figure 28: Transient Federation – SP-initiated 39
Figure 29: Transient Federation – IdP-initiated 40
Figure 30: Federation Termination 41
Figure 31: Single Logout 42
Figure 32: Multiple Logouts 43
Figure 33: Multiple Logouts – identity provider initiated 43

1 Introduction

The Security Assertion Markup Language (SAML) standard defines a framework for exchanging security information between online business partners.

More precisely, SAML defines a common XML framework for exchanging security assertions between entities. As stated in the Security Services Technical Committee (SSTC) charter, the purpose of the Committee is:

> *…to define, enhance, and maintain a standard XML-based framework for creating and exchanging authentication and authorization information.*

SAML uses the approach of expressing assertions about a subject in a portable fashion that other applications across system domain boundaries can trust.

What are the entities involved in a SAML interaction? 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 system entity that makes SAML assertions is known as an ***asserting party*** or sometimes a ***SAML authority***, and a system entity that uses received assertions is known as a ***relying party***. This latter entity's willingness to rely on information from an asserting party depends on the existence of a trust relationship between them. The relying party is sometimes called a ***SAML requester***, in that it is requesting information from a SAML authority.

Typically there are a number of ***service providers*** (SPs) that can make use of assertions about a subject in order to control access and provide customized services, and accordingly they become the relying parties of an asserting party called an ***identity provider*** (IdP). For example, a typical assertion from an identity provider might convey that "This user is John Doe, he has an email address of john.doe@acompany.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 access to local resources.

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

- ***SSO interoperability:*** How different products implement Cross-Domain Single Sign-On (CDSSO) has traditionally been completely proprietary. Most pre-SAML Single Sign-On products use browser cookies to maintain state so that re-authentication is not required. Browser cookies are not transferred between DNS domains. So, if you obtain a cookie from www.abc.com, then that cookie will not be sent in any HTTP messages to www.xyz.com. This could even apply within a single organization that has separate DNS domains. SAML solves the CDSSO problem by providing a standard vendor-independent protocol for transferring information about a (browser-equipped) user from one web server to another without relying on cookies.
- ***Federated identity:*** Federated identity deals with the sharing of information about user identities across organizational boundaries while maintaining privacy protection. From an administrative perspective, this type of sharing can help reduce identity management costs as multiple organizations do not need to independently collect and maintain identity-related data (e.g. passwords). From a user-centric viewpoint, as explained under SSO interoperability, this also results in an enhanced user experience with fewer sign-ons. In addition, administrators do not have to maintain the mappings; rather control can reside with the user.
- ***Web services:*** SAML allows its security assertion format to be used outside of a "native" SAML-based protocol context, and this modularity has proven useful within the web services environment. The WS-Security effort has defined how to use a SAML assertion as a security token. The SAML assertion format (and associated security token definition) provides a means by which security assertions about messages and service requesters can be exchanged between communicating service endpoints. 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, not easily contained within other WS-Security token formats.

2 SAML Use Cases

- Prior to examining the details of the SAML standard, it's useful to describe some of its high-level use cases. (Later on, more detailed use cases are described based on specific SAML profiles.)

2.1 Single Sign-On Use Case

Single sign-on is the classic use case supported in SAML V1.0 and V1.1. 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 another web site (typically in a different DNS domain). The identity provider site (*AirlineInc.com*) asserts to the service provider site (*CarRentalInc.com*) that the user is known to it and provides the user's name and possibly additional session attributes (e.g. “Gold member”). The user's identity is federated between *AirlineInc.com* and *CarRentalInc.com* by business agreement between the partners. As *CarRentalInc.com* trusts *AirlineInc.com*, it knows that the user is valid and creates a session for the user. This use case illustrates the fact that the user is not required to re-authenticate when directed over to the *CarRentalInc.com* site

Figure 1 illustrates the SSO high-level use case.

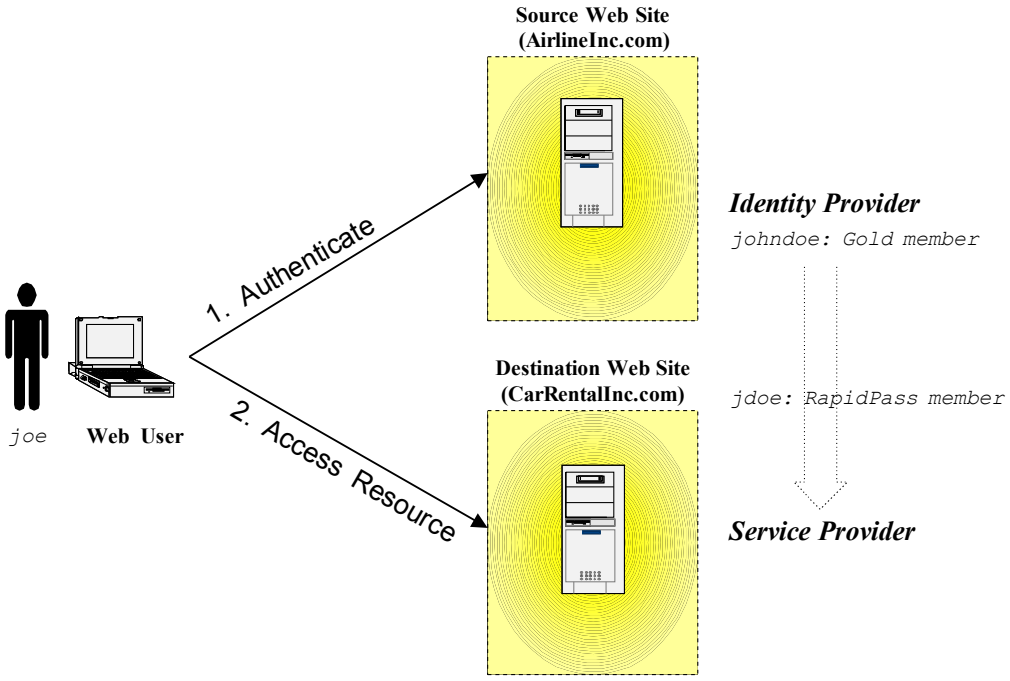


Figure 1: Single Sign-On Use Case [@@remove attr aspect]

This high-level description implies that the user always visits and is authenticated at the IdP as a first step in order to access protected resources at that site, and this is indeed sometimes the case. However, often a user starts out by visiting an SP site, accessing resources that require no special authentication or authorization, and subsequently attempts to access a protected resource at the SP. For these reasons, the SP may need to direct an authentication request to the IdP (which may ultimately require the IdP to interact with the user) in order to validate the user's access rights to the protected resource. SAML provides for both patterns, which it calls ***IdP-initiated*** and ***SP-initiated*** flows.

At a more detailed technical level, SAML provides a number of different “delivery mechanisms” (called bindings) for each request and response message to and from a SAML authority. Thus, in practice, there are several flavors of IdP-initiated and SP-initiated flows. These are described in more detail in Section 4.1.

2.2 Identity Federation Use Cases

Once it is possible to share identity information between providers, further questions arise concerning the properties of the transmitted information. These include such issues as the format and values transmitted, relevant processing rules at identity and service providers, and assumptions about contents of identity stores at the providers. These tasks fall into the general category of identity management and are more specifically described by the term identity federation.

A user's identity is said to be federated between a set of providers when there is agreement between the providers on a set of identifiers and/or attributes to use when referring to the user. There are many different techniques that may be used to implement the data flows required for such agreements between providers.

In some cases, some of the required exchanges of identity-related information may take place outside of the SAML V2.0 standard using other infrastructure. For example, providers may choose to share information about newly registered or de-registered users via batch “identity feeds” that are driven by identity sources (for example, human resources databases) at the identity provider and propagated to service providers. Subsequently, the user name may be placed in a SAML assertion and propagated between providers to implement single sign-on. 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

A typical use case for achieving identity federation using SAML is “account linking.” Figure 2 illustrates one scenario. Two service providers exist, one for car rentals, the other for hotel bookings. In addition to *AirlineInc.com*, users are registered on both service provider sites, but using different names. At *AirlineInc.com*, user **joe** may be registered as **johndoe**, on *CarRentalInc.com* as **jdoe**, and on *HotelBookings.com* as **johnd**. SAML V2.0 supports a model for federated identity based on pseudonyms. A pseudonym is a privacy-preserving identifier shared between entities. In this use case, *AirlineInc.com* describes the user to *CarRentalInc.com* and *HotelBooking.com* using (distinct) pseudonyms. Each of *CarRentalInc.com* and *HotelBooking.com* can link the pseudonym to the existing user account once user consent has been obtained. In subsequent access, the user will only need to log in once to *AirlineInc.com* before beginning to conduct business at *CarRentalInc.com* and *HotelBooking.com* using account information available at these sites.

Figure 2: Account Linking Use Case [@@replace missing figure]

SAML V2.0 supports several features that are desirable when working with federated identity. For example, confidentiality is supported by permitting various SAML constructs to be encrypted. In addition, providers can capture information about user consent and transmit it within SAML messages.

3 SAML Architecture

This section provides a brief description of the concepts that underlie SAML and the component pieces 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. Primarily the components permit transfer of identity, authentication, attribute, and authorization information to be exchanged between autonomous organizations. The "core" SAML specification defines the structure and content of ***Assertions*** – which carry statements about a Principal as asserted by an Asserting Party. These are defined by an XML schema.

Assertions are either requested or just "pushed" out to the service provider. How and which assertions are requested is defined by the ***SAML Protocols***, which have their own XML schema. The lower-level communication or messaging protocols (such as HTTP or SOAP) over which the SAML protocol messages can be transported are defined by ***Bindings***. SAML Protocols and Bindings, together with the structure of Assertions, can be combined together to create a ***Profile*** for greater interoperability. In general Profiles can be thought of a satisfying a particular use case, for example the Web Browser SSO profile. There are also Attribute Profiles (for example, LDAP and DCE profiles), which define how to interpret attribute information carried within an Assertion using common attribute/directory technologies.

Two other SAML components can be used in building a system:

- **Metadata:** Metadata defines a way to express and share configuration information between two communicating providers. For instance, an entity's support for given SAML bindings, identifier information, and PKI information can be defined. Metadata is defined by an XML schema. The location of Metadata is defined using DNS records.
- **Authentication Context:** In a number of situations the service provider may wish to have additional information in determining the authenticity and confidence they have in the information within an assertion. Authentication Context permits the augmentation of Assertions with additional information pertaining to the authentication of the principal at the identity provider. For instance, details of multi-factor authentication can be included.

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

3.2 Summary of SAML Components

The SAML components and their individual parts are as follows:

- **Assertions:** SAML allows for one party to assert characteristics and attributes of a subject. For instance, a SAML assertion could state that the subject is "John Doe", has "Gold" status, has an email address of john.doe@example.com, and is a member of the "engineering" group. SAML assertions are encoded in an XML schema. SAML defines three kinds of statements that can be carried within an assertion:
  - **Authentication statements:** These are issued by the party that successfully authenticated the user. They describe who issued the assertion, the authenticated subject, and the validity period, plus other authentication-related information.
  - **Attribute statements: These** contain specific details about the subject (for example, that user "John Doe" has "Gold" status).
  - **Authorization decision statements:** These define something the subject is entitled to do (for example, whether "John Doe" is permitted to buy a specified item).
- **Protocols:** SAML defines a number of request/response protocols, which are encoded in an XML schema as a set of request-response pairs. The protocols defined are:
  - **Assertion Query and Request Protocol:** Defines a set of queries by which existing SAML

assertions may be obtained. The query can be on the basis of a SAML message reference, the subject, or the statement type.

- **Authentication Request Protocol:** Defines a protocol by which a service provider or principal can request assertions from an identity provider tailored to the requirements of a particular SAML profile, for example the Web Browser SSO Profile.
- **Artifact Resolution Protocol:** Provides a mechanism by which protocol messages may be passed by reference using a small, fixed-length value called an ***artifact***. The artifact receiver uses the Artifact Protocol to dereference the actual protocol message.
- **Name Identifier Management Protocol:** Provides mechanisms to change the value of the principal's name identifier. 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 between an identity provider and service provider.
- **Single Logout Protocol:** Defines a request that allows near-simultaneous logout of all sessions associated by a principal. The logout can be directly initiated by the principal, due to a session timeout or because a user access rights have been revoked. Logout can also be initiated by a provider site.
- **Name Identifier Mapping Protocol:** Provides a mechanism to programmatically map one SAML name identifier into another, subject to appropriate policy controls.

In addition to request and response messages comprising the protocols listed here, SAML provides for a special representation of any type of protocol message, called an ***artifact***, which can be dereferenced to obtain the full message. An artifact takes the form of a base-64 encoded string.

- **Bindings:** These detail exactly how the SAML protocols map onto underlying transport protocols. For instance, the SAML specification provides a binding of how SAML requests and responses are carried with SOAP exchange messages. The bindings defined are:
  - **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.
  - **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:** Any SAML protocol message can be represented by an artifact, which has a compact base-64 format and allows for the real message to be “pulled” (dereferenced). This binding defines how an artifact is transported by HTTP using one of two mechanisms: either an HTML form control or a query string in the URL.
  - **SAML URI Binding:** Defines a means for retrieving a SAML assertion by resolving a URI (uniform resource identifier).
- **Profiles:** Profiles define how the SAML assertions, protocols, and bindings are combined for interoperability in particular usage scenarios. Some of these are described in detail later on in the document. In summary they are:
  - **Web Browser SSO Profile:** Defines a mechanism for single sign-on by unmodified web browsers to multiple service providers using the Authentication Request protocol in combination with the HTTP Redirect, HTTP POST, and Artifact bindings.
  - **Enhanced Client and Proxy (ECP) Profile:** Defines a profile of the Authentication Request protocol in conjunction with the Reverse-SOAP and SOAP bindings suited to clients or gateway

devices with knowledge of one or more identity providers.

- **Identity Provider Discovery Profile:** Defines one possible mechanism for a set of cooperating Identity and service providers to obtain the identity providers used by a principal.
- **Single Logout Profile:** A profile of the SAML Single Logout protocol is defined. Defines how SOAP, HTTP Redirect, HTTP POST, and HTTP Artifact bindings may be used.
- **Name Identifier Management Profile:** Defines how the Name Identifier Management protocol may be used with SOAP, HTTP Redirect, HTTP POST, and HTTP Artifact bindings.
- **Artifact Resolution Profile:** Defines how the Artifact Resolution protocol uses a synchronous binding, for example the SOAP binding.
- **Assertion Query/Request Profile:** Defines how the SAML query protocols (used for obtaining SAML assertions) use a synchronous binding such as the SOAP binding.
- **Name Identifier Mapping Profile:** Defines how the Name Identifier Mapping protocol uses a synchronous binding such as the SOAP binding.

Figure 3 illustrates the relationship between the components:

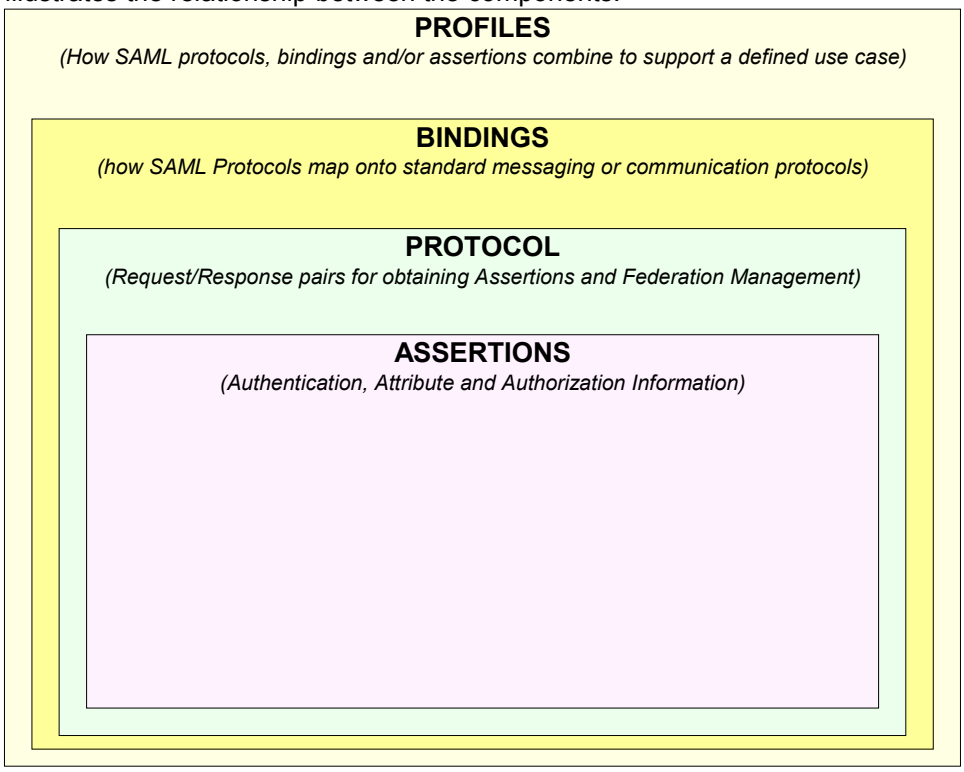


Figure 3: SAML Components

It should be noted that the story of SAML does not end with its published set of assertions, protocols, bindings, and profiles. It is designed to be highly flexible, and thus 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.

3.3 SAML Constructs and Examples

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

3.3.1 The Relationship of Bindings, Protocol Messages, Assertions, and Statements

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 conveyed by a SAML protocol response message, which itself must be carried by some sort of transport or messaging protocol.

Figure 4 shows a typical example of containment: a SAML Assertion containing a series of statements, the whole being carried within a SAML response, which itself is within a SOAP body.

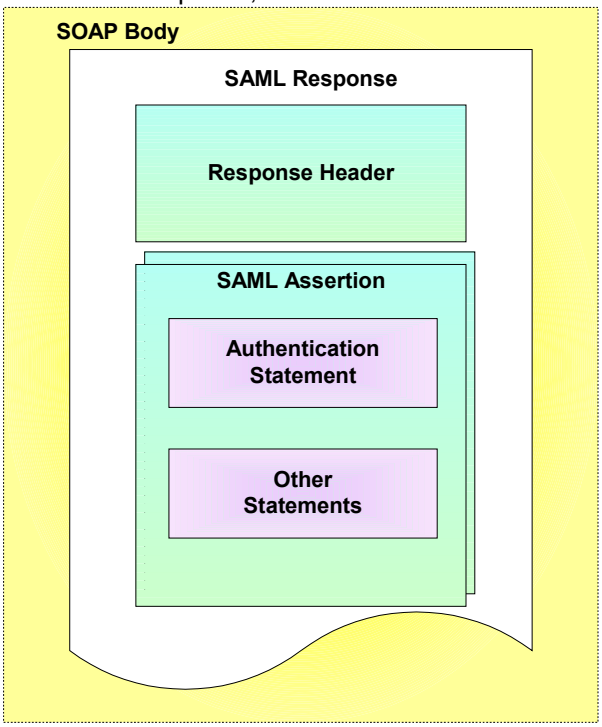


Figure 4: SAML Assertion Containment Overview

3.3.2 Assertion, Statement, and Subject Structure

Figure 5 shows an XML fragment containing an example assertion with a single authentication statement. Note the following:

- Line 1 contains the declaration of the SAML assertion namespace, which is conventionally represented in the specifications with the `saml:` prefix.
- Lines 3 through 6 provide information about the nature of the assertion: when it was issued and who issued it.
- Lines 7 through 12 provide information about the subject of the assertion, to which all of the contained statements (in this case only a single authentication statement) apply. The subject has a name identifier (line 10) whose value is “j.doe@acompany.com”, provided in a format whose label is given on line 9. SAML predefines a number of 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.
- The authentication statement appearing on lines 17 through 24 shows that this subject was originally authenticated using a password-protected transport mechanism at the time and date

shown. SAML predefines some dozens of labels for various authentication mechanisms, and you can also define your own.

```
 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>
 5:     www.acompany.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@acompany.com
11:     </saml:NameID>
12:   </saml:Subject>
13:   <saml:Conditions
14:     NotBefore="2005-01-31T12:00:00Z"
15:     NotOnOrAfter="2005-01-31T12:00:00Z">
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 5: Example of SAML Assertion, Subject, and Authentication Statement Constructs

The `<NameID>` element within `<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 identifiers require further discussion. ***Transient identifiers*** support “anonymity” at the SP since they correspond to a “one-time use” identifier created at the IdP. ***Persistent identifiers*** support pseudonymity at SPs; they are privacy-preserving and their use is restricted to an IdP-SP pair. The concept of ***affiliation*** permits a group of SPs to consume a single shared persistent identifier used to describe a principal. Affiliations are indicated by the `SPNameQualifier` attribute on the `<NameID>` and `<NameIDPolicy>` elements.

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 even in lieu of authentication information in scenarios such as “attribute federation.” SAML's attribute structure does not presume that any particular type of data store is being used for the attributes; it has an agnostic structure.

Figure @@nn shows an XML fragment containing an attribute statement. Note the following:

- A single statement can contain multiple attributes. In this case, there are two attributes (starting on line 2 and line 9) within the statement.
- Similarly to name identifier formats, attribute names are also qualified with a format label which indicates how the attribute name is to be interpreted. In both of the cases here (lines 3 and 10), the name format is not one of those predefined by SAML, but is rather defined by a third party, SmithCo. This is a fairly artificial example, and interoperability would increased by either the use of one of SAML's ***attribute profiles*** or the formal definition of a third-party attribute profile. See the

SAML Profiles specification [@@add bibref] for more information and examples.

- The value of an attribute can be plain text, as on line 6, or can be structured XML, as on lines 13 through 15.

```
 1: <saml:AttributeStatement xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
 2:   <saml:Attribute
 3:     NameFormat="http://smithco.com"
 4:     Name="PaidStatus">
 5:     <saml:AttributeValue>
 6:       PaidUp
 7:     </saml:AttributeValue>
 8:   </saml:Attribute>
 9:   <saml:Attribute
10:     NameFormat="http://smithco.com"
11:     Name="CreditLimit">
12:     <saml:AttributeValue xsi:type="smithco:type">
13:       <smithco:amount currency="USD">
14:         500.00
15:       </my:amount>
16:     </saml:AttributeValue>
17:   </saml:Attribute>
18: </saml:AttributeStatement>
```

Figure @@nn: Example of Attribute Statement

3.3.4 Message Structure and the SOAP Binding

In environments where the two communicating endpoints are SOAP-enabled, then the SOAP-over-HTTP binding can be used to exchange SAML request/query and response protocol messages. Figure 6 provides an overview of the structure. The SAML request or SAML response being carried within the SOAP body, which itself has an HTTP response wrapper.

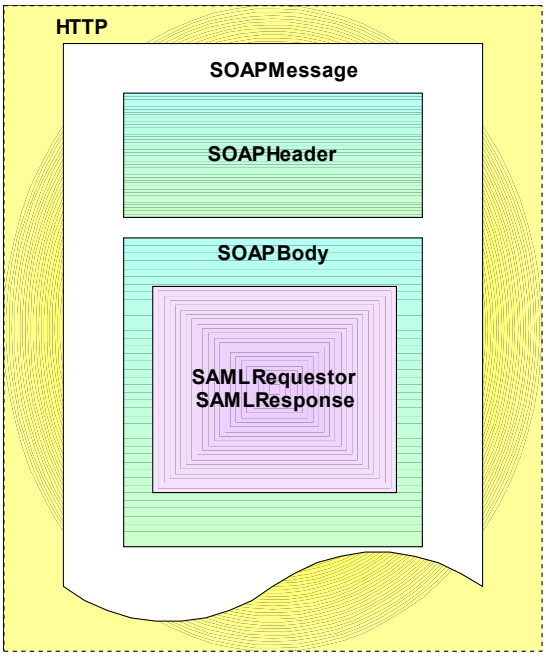


Figure 6: SAML Protocol Message Conveyed with SOAP over HTTP

Figure 7 shows a complete XML document containing an example of a SAML authentication request message being transported within a SOAP message. Note the following:

- The authentication request starting on line 5 is embedded in a SOAP body element starting on line 4. The request contains, on line 6, a declaration of the SAML protocol namespace, which is conventionally represented in the specifications with the `samlp:` prefix. The protocol namespace

and the assertion namespace are separate (and SAML defines a number of adjunct vocabularies with their own namespaces as well).

- The request element provides a number of parameters that govern the type of assertion it expects back, for example, the requested subject (lines 14 through 20) and whether to force fresh authentication of the subject (line 7).

```
 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:       ForceAuthn="true"
 8:       AssertionConsumerServiceURL="https://www.sp.example.com/SSO"
 9:       AttributeConsumingServiceIndex="0" ProviderName="CarRentalInc.com"
10:       ID="abe567de6"
11:       Version="2.0"
12:       IssueInstant="2005-01-31T12:00:00Z"
13:       Destination="https://www.idp.example.com/" >
14:       <saml:Subject
15:         xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
16:         <saml:NameID
17:           Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
18:             j.doe@acompany.com
19:         </saml:NameID>
20:       </saml:Subject>
21:     </samlp:AuthnRequest>
22:   </env:Body>
23: </env:Envelope>
```

Figure 7: Example of SAML Authentication Request Conveyed by SOAP

Figure 8 shows an XML fragment containing a SAML response. Note the following:

- On line 6, the response references the request to which it is responding, and additional information is provided in the response header (lines 4 through 13), including status information. SAML predefines a number of status codes and, in many cases, dictates the circumstances under which each must be used.
- Within the response (line 14; detail elided) is a SAML assertion, typically containing one or more statements as shown above.

```
 1: <env:Envelope  xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
 2:   <env:Body>
 3:     <samlp:Response xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
 4:       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 5:       ID="abe567de6"
 6:       InResponseTo="example-ncname" Version="2.0"
 7:       IssueInstant="2005-01-31T12:00:00Z"
 8:       Destination="https://www.idp.example.com/">
 9:       <samlp:Status>
10:         <samlp:StatusCode Value="samlp:Success"/>
11:         <samlp:StatusMessage>Success</samlp:StatusMessage>
12:         <samlp:StatusDetail/>
13:       </samlp:Status>
14:                   ...SAML assertion...
15:     </samlp:Response>
16:   </env:Body>
17: </env:Envelope>
```

Figure 8: Example of SAML Response

3.4 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.

3.4.1 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 the use of SAML by Web Services Security (WS-Security) [@@add bibref], which is a set of specifications that define means for providing security protection of SOAP messages. The primary services provided WS-Security are authentication, data integrity, and confidentiality.

WS-Security defines a `<Security>` element that may be included in the SOAP header. This element contains information that specifies how the message is protected. WS-Security makes use of mechanisms defined in the XML Digital Signature and XML Encryption specifications to sign and encrypt message data in both the header and the body. The information in the `<Security>` element specifies what operations were performed and in what order, what keys were used for the 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 keys and attributes are specified using tokens. Tokens can either be binary or 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 be used to refer to a token in one of a number of ways.

WS-Security consists of a Core Specification which describes the mechanisms independent of the type of token being usedand 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 [@add bibref].

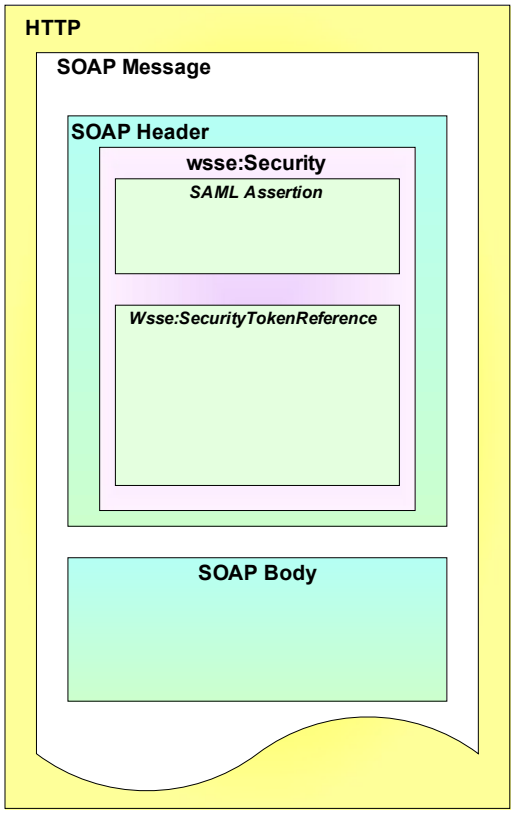
Because the SAML protocols have a binding to SOAP, it is easy to get confused between that and the use of SAML Assertions by WS-Security. They can be distinguished by their purpose, the message format, and the parties involved.

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

- It is used to obtain SAML Assertions for future use; they play no role in protecting the message.
- The SAML Assertions are contained within a SAML Response, which is carried in the SOAP body.
- The SAML Assertions are provided by a trusted authority or repository 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 usually play a role in the protection of the message they are carried in, typically they contain a key used for digital signatures.
- The SAML Assertions are carried in a `<Security>` element within the SOAP header.
- The SAML Assertions will have been obtained previously and typically pertain to the identity of the sender. Figure 9: *WS-Security* and SAML Relationship



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.

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

1. Sender obtains SAML Assertion by means of SAML Request/Response or other SAML Profile. Assertion contains attribute statement and Subject Confirmation Method of Holder of Key.
2. Sender constructs SOAP message, including Security header. SAML Assertion is included in Security header. Key referred to by SAML Assertion is used to construct digital signature over data in message body. Signature information is also included in Security header.
3. Receiver verifies digital signature.
4. The information in the SAML Assertion is used for purposes such as Access Control and Audit logging.

Figure 10 illustrates this usage scenario.

Figure 10: Typical use of WS-Security and SAML [@@add figure back!]

3.4.2 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. 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.

Figure 11 illustrates the typical use of SAML with XACML.

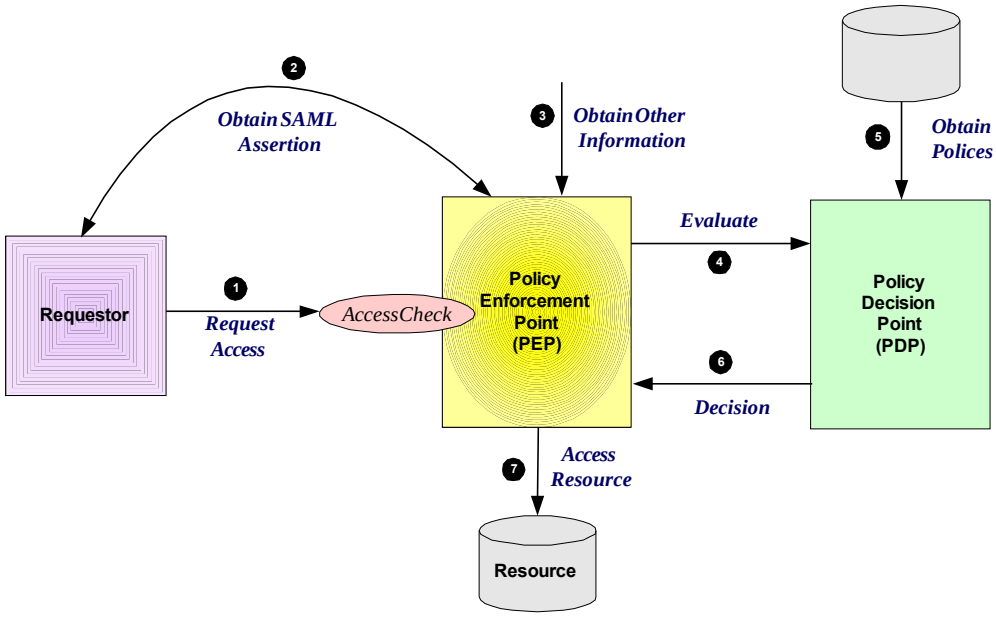


Figure 11: Typical use of XACML and SAML

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

The XACML Attribute Profile, which can be found in the SAML Profiles specification, defines how SAML attributes may be mapped to XACML Attributes. A schema is provided by SAML to facilitate this.

One of the XACML specifications, SAML V2.0 profile of XACML, provides additional information on mapping SAML Attributes to XACML Attributes.

The SAML V2.0 profile of XACML 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.

The 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.

3.5 Security in SAML

[@@add SecConsider bibref]Just providing assertions from an asserting party to a relying party may not be adequate for 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 grabs assertions to be illicitly “replayed” at a later date? SAML defines a number of security mechanisms that prevent or detect such attacks. The primary mechanism is for the relying party and asserting party to have a pre-existing trust relationship, typically involving a Public Key Infrastructure (PKI). While use of a PKI is not mandated, it is recommended. Use of particular mechanisms is described for each profile; however, an 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 then bi-lateral authentication is required and the use of SSL 3.0 or TLS 1.0 using server and client authentication is recommended.
- When an assertion or request “pushed” to a relying party (for example using the HTTP POST binding), then it is mandated that the response message be digitally signed using the XML digital signature standard.

4 Profiles

SAML supports a number of use cases and profiles. The purpose of this section is to describe a number of the more important ones. The following are described:

- Web Browser SSO Profile
- Enhanced Client and Proxy (ECP) Profile
- Federation Use Cases
- Single Logout

4.1 Web Browser SSO Profile

The Web Browser SSO profile supports a variety of options, having to do with two dimensions of choice: whether the message flows are IdP-initiated or SP-initiated (as discussed in Section 2.1) and whether the IdP pushes SAML assertions to the SP or the SP pulls them from the IdP. The push approach involves using either HTTP redirects or HTTP POST messages to deliver a SAML message. The pull approach involves sending a artifact to the receiver, which then uses the artifact to dereference and obtain the related SAML message. A combination of message flow and binding techniques gives rise to eight different combinations, all of which are described in this section:

- Message flow initiated by the SP:
  - POST binding used for both the request and the response
  - Redirect binding used for the request and POST binding used for the response
  - Artifact binding used for the request and POST binding used for the response
  - POST binding used for the request and artifact binding used for the response
  - Redirect binding used for the request and artifact binding used for the response
  - Artifact binding used for both the request and the response
- Message flow initiated by the IdP:
  - POST binding used for the (soliary) response
  - Artifact binding used for the response

Figure 12 compares the push and pull approaches.

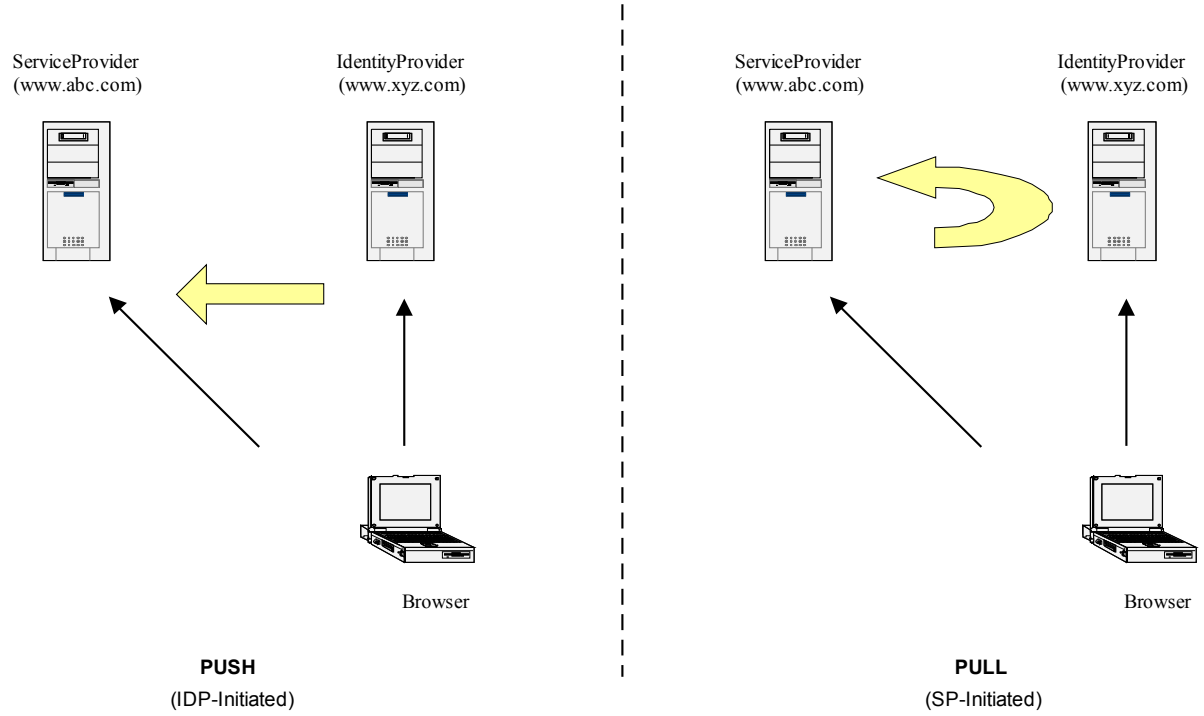


Figure 12: Push and Pull Models for Web Browser SSO Profile

656

657 4.1.1 SP initiated: POST->POST binding

658 In this use case the user attempts to access a resource on www.abc.com. However they do not have
659 current an existing session context on this site (e.g. logged on) and their identity is managed by
660 www.xyz.com. A SAML `<AuthnRequest>` is sent to their identity provider so that the identity provider
661 can provide back a SAML assertion concerning the user. HTTP POST messages are used to deliver the
662 SAML `<AuthnRequest>` to the identity provider as well as receive back the SAML response.

663 Figure 14 illustrates the message flow:

664

665

666

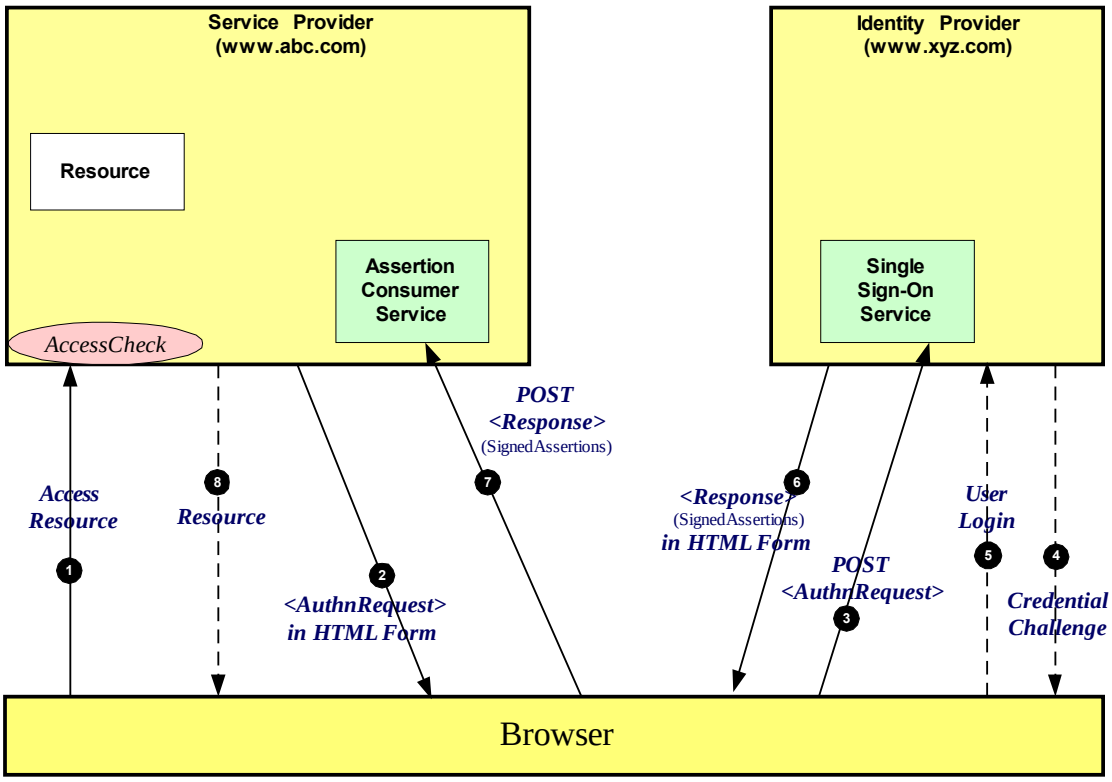


Figure 14: SP initiated: POST->POST binding

667 The processing is as follows:

668 1. The user attempts to access a resource on www.abc.com. The user does not have any current logon
669 session (i.e. security context) on this site, and is unknown to it.

670 2. The SP sends a HTML form back to the browser. The HTML FORM contains a SAML
671 `<AuthnRequest>` defining the user for which authentication and authorization information is
672 required. Typically the HTML FORM will contain an input or submit action that will result in an HTTP
673 POST.

674 3. The browser, either due to a user action or via an "auto-submit", issues an HTTP POST containing
675 the SAML `<AuthnRequest>` to the identity provider's Single Sign-On service.

676 4. If the user does not have an existing security context on the identity provider, or the policy defines
677 that authentication is required, they user will be challenged to provide valid credentials.

678 5. The user provides valid credentials and a security context is created for the user.

679 6. The Single Sign-On Service sends a HTML form back to the browser. The HTML FORM contains a
680 SAML response, within which is a SAML assertion. The SAML specification mandates that the
681 response must be digitally signed. Typically the HTML FORM will contain an input or submit action
682 that will result in an HTTP POST.

683 7. The browser, either due to a user action or via an "auto-submit", issues an HTTP POST containing
684 the SAML response to be sent to the service provider's Assertion Consumer service.

685 8. The service provider's Assertion Consumer service validates the digital signature on the SAML
686 Response. If this, and the Assertion validate correctly, it sends an HTTP redirect to the browser
687 causing it to access the TARGET resource, with a cookie that identifies the local session (use of a
688 cookie is implementation specific, other techniques to maintain the security context at the SP can be
689 used). An access check is then made to establish whether the user has the correct authorization to
690 access the www.abc.com web site and the TARGET resource. If the access check passes, the
691 TARGET resource is then returned to the browser.

4.1.2 SP initiated: Redirect->POST binding

In this use case the user attempts to access a resource on www.abc.com. However they do not have current logon session on this site and their identity is managed by www.xyz.com. A SAML `<AuthnRequest>` is sent to their identity provider so that the identity provider can provide back a SAML assertion concerning the user. An HTTP redirect message is used to deliver the SAML `<AuthnRequest>` to the identity provider and an HTTP POST is used to return the SAML response.

Figure 15 illustrates the message flow:

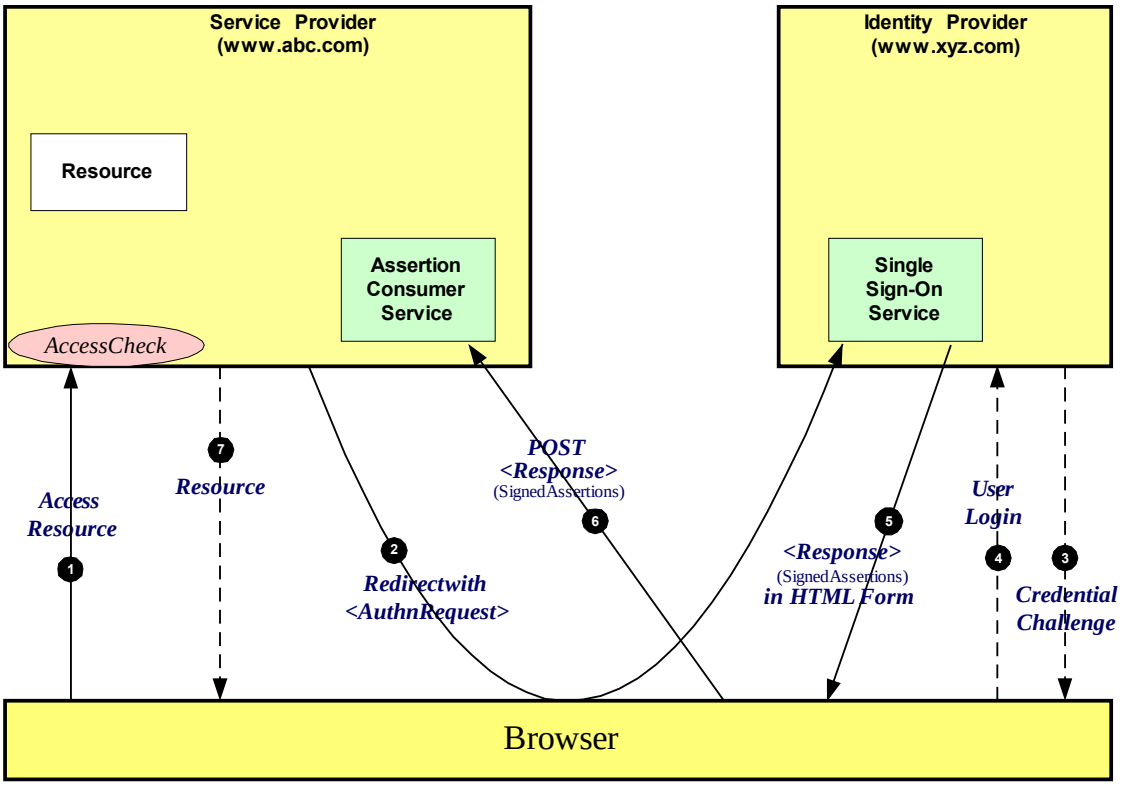


Figure 15: SP initiated: Redirect->POST binding

The processing is as follows:

1. The user attempt to access a resource on www.abc.com. The user does not have any current logon session (i.e. security context) on this site, and is unknown to it.
2. The SP sends a redirect message to the browser with HTTP status code of either 302 or 303. The Location HTTP header contains the destination URI of the Sign-On Service of the identity provider together with the `<AuthnRequest>` as a query variable named `SAMLRequest`. The query string is encoded using the DEFLATE encoding. The browser processes the redirect message and issues a GET to the Sign-on Service with the `SAMLRequest` query parameter.
3. The Sign-on Service determines whether the user has an existing security context on the identity provider, or that the policy defines that authentication is required. If the user requires to be authenticated he will be challenged to provide valid credentials.
4. The user provides valid credentials and a security context is created for the user.
5. The Single Sign-On Service sends a HTML form back to the browser. The HTML FORM contains a SAML response, within which is a SAML assertion. The SAML specification mandates that the response must be digitally signed. Typically the HTML FORM will contain an input or submit action that will result in an HTTP POST.
6. 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 service provider's Assertion Consumer service.

7. The service provider's Assertion Consumer service validates the digital signature on the SAML Response. If this, and the Assertion validate correctly, it sends an HTTP redirect to the browser causing it to access the TARGET resource, with a cookie that identifies the local session (use of a cookie is implementation specific, other techniques to maintain the security context at the SP can be used). An access check is then made to establish whether the user has the correct authorization to access the www.abc.com web site and the TARGET resource. If the access check passes, the TARGET resource is then returned to the browser.

4.1.3 SP initiated: Artifact->POST binding

In this use case the user attempts to access a resource on www.abc.com. However they do not have a current logon session on this site and their identity is managed by www.xyz.com. A SAML artifact is sent to the identity provider (using an HTTP redirect), which it uses to obtain a SAML `<AuthnRequest>` from the service provider's SAML Responder. When the identity provider obtains the SAML `<AuthnRequest>` it provides back to the service provider the SAML response using the HTTP POST binding mechanism.

Figure 16 illustrates the message flow:

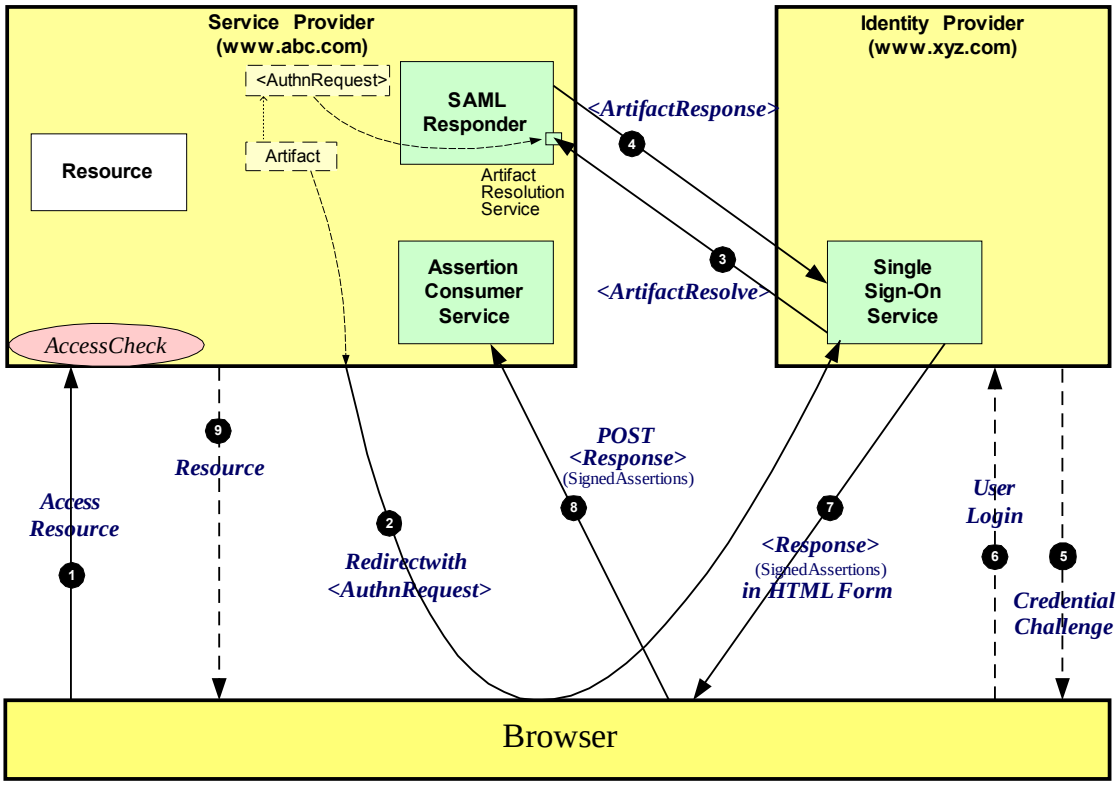


Figure 16: SP initiated: Artifact->POST binding

The processing is as follows:

1. The user attempt to access a resource on www.abc.com. The user does not have any current logon session (i.e. security context) on this site, and is unknown to it.
2. The SP generates the `<AuthnRequest>` while also creating an artifact. The artifact contains the source ID of the www.abc.com SAML responder together with a reference to the assertion (the MessageHandle). The HTTP Artifact binding allows the choice of either HTTP redirection or a HTML form as the delivery mechanism to the service provider. The figure shows the use of the HTML form mechanism. The service provider sends a HTML form back to the browser. The HTML FORM

contains the SAML artifact, the control name being `SAMLart`. Typically the HTML FORM will contain an input or submit action that will result in an HTTP POST.

3. On receiving the HTTP message, the Single Sign-On Service, extracts the SourceID from the SAML artifact. A mapping between source IDs and remote Responders will already have been established administratively. The Assertion Consumer service will therefore know that it has to contact the www.abc.com SAML responder at the prescribed URL. It sends the SAML `<ArtifactResolve>` message to the service provider's SAML responder containing the artifact supplied by its service provider.
4. The SAML responder supplies back a SAML `<ArtifactResponse>` message containing the `<Authn Request>` previously generated.
5. The Sign-on Service determines whether the user, for which the `<AuthnRequest>` pertains, has an existing security context on the identity provider, or that the policy defines that authentication is required. If the user requires to be authenticated he will be challenged to provide valid credentials.
6. The user provides valid credentials and a security context is created for the user.
7. The Single Sign-On Service sends a HTML form back to the browser. The HTML FORM contains a SAML response, within which is a SAML assertion. The SAML specification mandates that the response must be digitally signed. Typically the HTML FORM will contain an input or submit action that will result in an HTTP POST.
8. 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 service provider's Assertion Consumer service.
9. The service provider's Assertion Consumer service validates the digital signature on the SAML Response. If this, and the Assertion validate correctly, it sends an HTTP redirect to the browser causing it to access the TARGET resource, with a cookie that identifies the local session (use of a cookie is implementation specific, other techniques to maintain the security context at the SP can be used). An access check is then made to establish whether the user has the correct authorization to access the www.abc.com web site and the TARGET resource. If the access check passes, the TARGET resource is then returned to the browser.

4.1.4 SP initiated: POST->Artifact binding

In this use case the user attempts to access a resource on www.abc.com. However they do not have current logon session on this site and their identity is managed by www.xyz.com. A SAML `<AuthnRequest>` is sent to their identity provider so that the identity provider can provide back a SAML assertion concerning the user. A HTTP POST message is used to deliver the SAML `<AuthRequest>` to the identity provider. The response is in the form of a SAML Artifact. In this example the SAML Artifact is provided back within an HTTP POST message. The service provider uses the SAML artifact to obtain the SAML response (containing the SAML assertion) from the identity provider's SAML Responder.

Figure 17 illustrates the message flow:

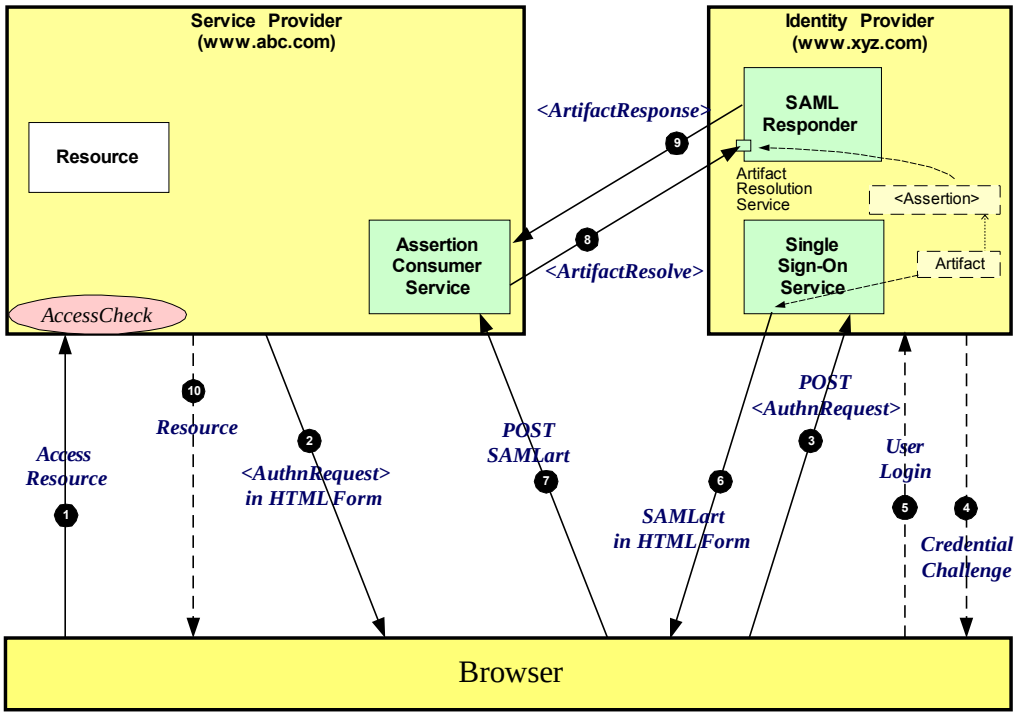


Figure 17: SP initiated: POST->Artifact binding

The processing is as follows:

1. The user attempt to access a resource on www.abc.com. The user does not have any current logon session (i.e. security context) on this site, and is unknown to it.
2. The SP sends a HTML form back to the browser. The HTML FORM contains a SAML `<AuthnRequest>` defining the user for which authentication and authorization information is required. Typically the HTML FORM will contain an input or submit action that will result in an HTTP POST.
3. The browser, either due to a user action or via an “auto-submit”, issues an HTTP POST containing the SAML `<AuthnRequest>` to the identity provider's Single Sign-On service.
4. If the user does not have an existing security context on the identity provider, or the policy defines that authentication is required, they user will be challenged to provide valid credentials.
5. The user provides valid credentials and a security context is created for the user.
6. The Single Sign-On Service generates an assertion for the user while also creating an artifact. The artifact contains the source ID of the www.xyz.com SAML responder together with a reference to the assertion (the MessageHandle). The HTTP Artifact binding allows the choice of either HTTP redirection or a HTML form as the delivery mechanism to the service provider. The figure shows the use of the HTML form mechanism. The Single Sign-On Service sends a HTML form back to the browser. The HTML FORM contains the SAML artifact, the control name being `SAMLart`. Typically the HTML FORM will contain an input or submit action that will result in an HTTP POST.
7. On receiving the HTTP message, the Assertion Consumer service, extracts the SourceID from the SAML artifact. A mapping between source IDs and remote Responders will already have been established administratively. The Assertion Consumer service will therefore know that it has to contact the www.xyz.com SAML responder at the prescribed URL.
8. The www.abc.com Assertion Consumer service will send a SAML `<ArtifactResolve>` message to the identity provider's SAML responder containing the artifact supplied by the identity provider.
9. The SAML responder supplies back a SAML `<ArtifactResponse>` message containing the assertion previously generated. In most implementations, if a valid assertion is received back, then a session on www.abc.com is established for the user (the relying party) at this point.

10. Typically the Assertion Consumer service then sends a redirection message containing a cookie back to the browser (use of a cookie is implementation specific, other techniques to maintain the security context at the SP can be used). The cookie identifies the session. The browser then processes the redirect message and issues an HTTP GET to the TARGET resource on www.abc.com. The GET message contains the cookie supplied back by the Assertion Consumer service. An access check is then back to established whether the user has the correct authorization to access the www.abc.com web site and the index.asp resource.

4.1.5 SP initiated: Redirect->Artifact binding

In this use case the user attempts to access a resource on www.abc.com. However they do not have current logon session on this site and their identity is managed by www.xyz.com. A SAML `<AuthnRequest>` is sent to their identity provider so that the identity provider can provide back a SAML assertion concerning the user. A HTTP redirect message is used to deliver the SAML `<AuthRequest>` to the identity provider. The response is in the form of a SAML Artifact. In this example the SAML Artifact is provided back within an HTTP POST message. The service provider uses the SAML artifact to obtain the SAML response (containing the SAML assertion) from the identity provider's SAML Responder.

Figure 18 illustrates the message flow:

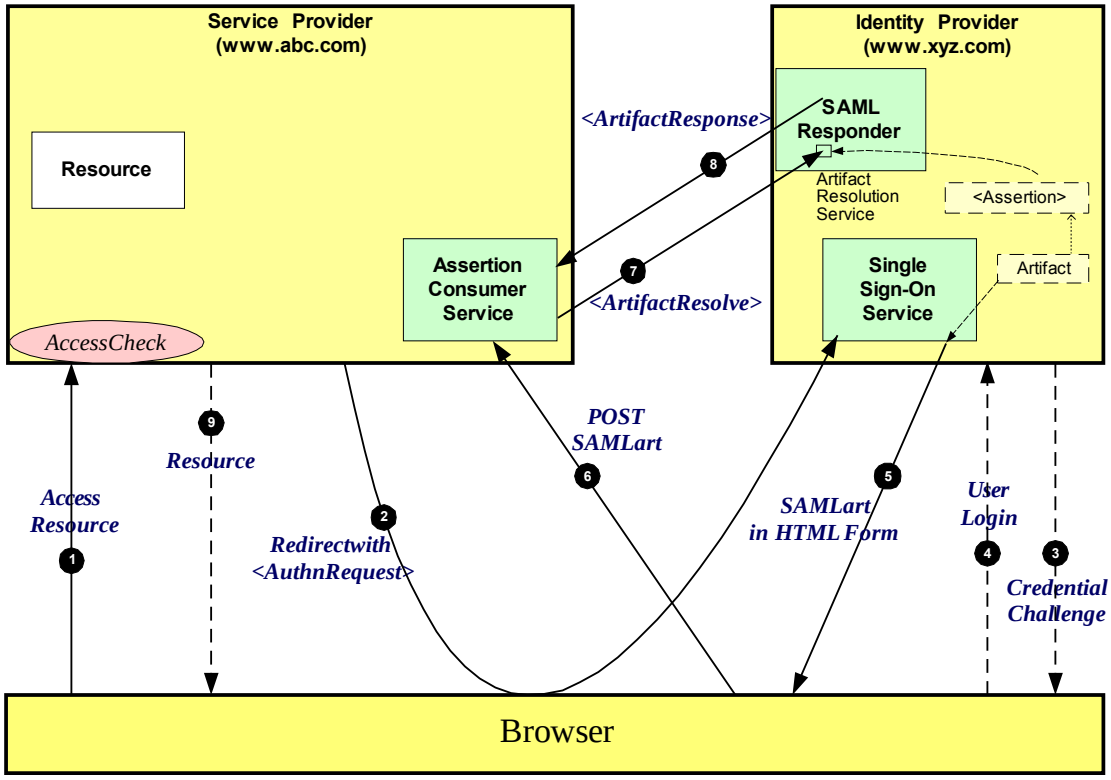


Figure 18: SP initiated: Redirect->Artifact binding

The processing is as follows:

1. The user attempt to access a resource on www.abc.com. The user does not have any current logon session (i.e. security context) on this site, and is unknown to it.
2. The SP sends a redirect message to the browser with HTTP status code of either 302 or 303. The Location HTTP header contains the destination URI of the Sign-On Service of the identity provider together with the `<AuthnRequest>` as a query variable named `SAMLRequest`. The query string is encoded using the DEFLATE encoding. The browser processes the redirect message and issues a

GET to the Sign-on Service with the `SAMLRequest` query parameter.

3. The Sign-on Service determines whether the user has an existing security context on the identity provider, or that the policy defines that authentication is required. If the user requires to be authenticated he will be challenged to provide valid credentials.
4. The user provides valid credentials and a security context is created for the user.
5. The Single Sign-On Service generates an assertion for the user while also creating an artifact. The artifact contains the source ID of the www.xyz.com SAML responder together with a reference to the assertion (the MessageHandle). The HTTP Artifact binding allows the choice of either HTTP redirection or a HTML form as the delivery mechanism to the service provider. The figure shows the use of the HTML form mechanism. The Single Sign-On Service sends a HTML form back to the browser. The HTML FORM contains the SAML artifact, the control name being `SAMLart`. Typically the HTML FORM will contain an input or submit action that will result in an HTTP POST.
6. On receiving the HTTP message, the Assertion Consumer service, extracts the SourceID from the SAML artifact. A mapping between source IDs and remote Responders will already have been established administratively. The Assertion Consumer service will therefore know that it has to contact the www.xyz.com SAML responder at the prescribed URL.
7. The www.abc.com Assertion Consumer service will send a SAML `<ArtifactResolve>` message to the identity provider's SAML responder containing the artifact supplied by the identity provider.
8. The SAML responder supplies back a SAML `<ArtifactResponse>` message containing the assertion previously generated. In most implementations, if a valid assertion is received back, then a session on www.abc.com is established for the user (the relying party) at this point.
9. Typically the Assertion Consumer service then sends a redirection message containing a cookie back to the browser. The cookie identifies the session (use of a cookie is implementation specific, other techniques to maintain the security context at the SP can be used). The browser then processes the redirect message and issues an HTTP GET to the TARGET resource on www.abc.com. The GET message contains the cookie supplied back by the Assertion Consumer service. An access check is then back to established whether the user has the correct authorization to access the www.abc.com web site and the index.asp resource.

4.1.6 SP initiated: Artifact->Artifact binding

In this use case the user attempts to access a resource on www.abc.com. However they do not have a current logon session on this site and their identity is managed by www.xyz.com. A SAML artifact is sent to the identity provider (using an HTTP redirect), which it uses to obtain a SAML `<AuthnRequest>` from the service provider's SAML Responder. When the identity provider obtains the SAML `<AuthnRequest>` it provides back to the service provider another SAML Artifact. In this example the SAML Artifact is provided back within an HTTP POST message. The service provider uses the SAML artifact to obtain the SAML response (containing the SAML assertion) from the identity provider's SAML Responder.

Figure 19 illustrates the message flow:

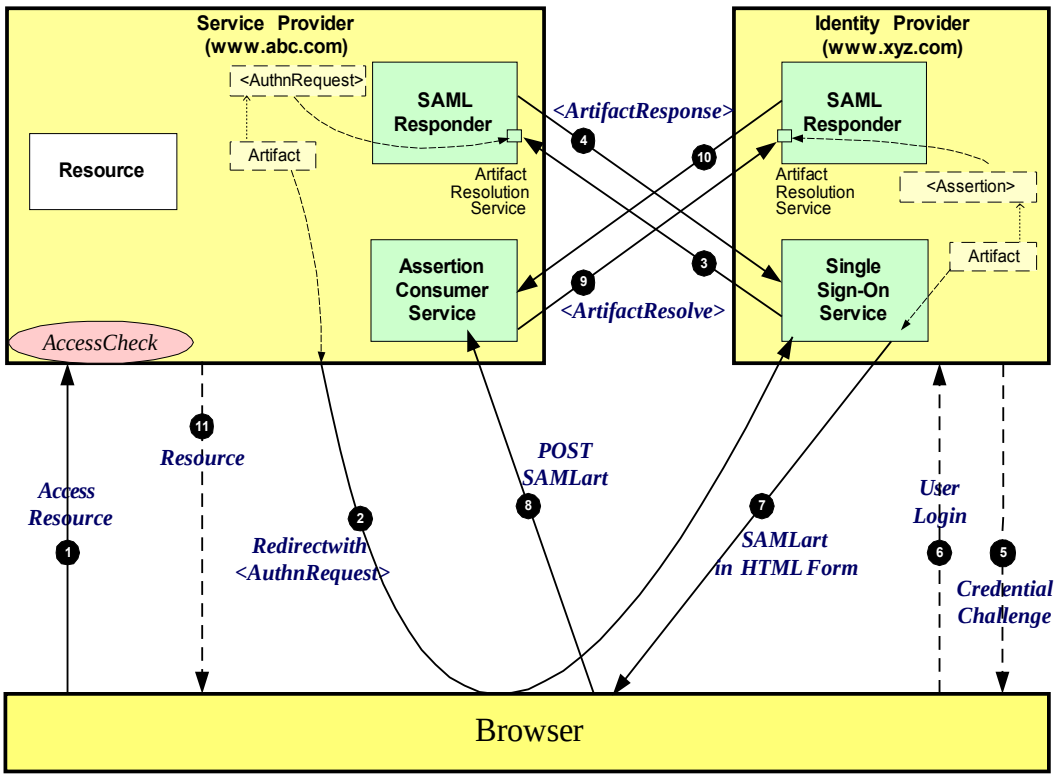


Figure 19: SP initiated: Artifact->Artifact binding

The processing is as follows:

1. The user attempt to access a resource on www.abc.com. The user does not have any current logon session (i.e. security context) on this site, and is unknown to it.
2. The SP generates the `<AuthnRequest>` while also creating an artifact. The artifact contains the source ID of the www.abc.com SAML responder together with a reference to the assertion (the MessageHandle). The HTTP Artifact binding allows the choice of either HTTP redirection or a HTML form as the delivery mechanism to the service provider. The figure shows the use of the HTML form mechanism. The service provider sends a HTML form back to the browser. The HTML FORM contains the SAML artifact, the control name being `SAMLart`. Typically the HTML FORM will contain an input or submit action that will result in an HTTP POST.
3. On receiving the HTTP message, the Single Sign-On Service, extracts the SourceID from the SAML artifact. A mapping between source IDs and remote Responders will already have been established administratively. The Assertion Consumer service will therefore know that it has to contact the www.abc.com SAML responder at the prescribed URL. It sends the SAML `<ArtifactResolve>` message to the service provider's SAML responder containing the artifact supplied by its service provider.
4. The SAML responder supplies back a SAML `<ArtifactResponse>` message containing the `<Authn Request>` previously generated.
5. The Sign-on Service determines whether the user, for which the `<AuthnRequest>` pertains, has an existing security context on the identity provider, or that the policy defines that authentication is required. If the user requires to be authenticated he will be challenged to provide valid credentials.
6. The user provides valid credentials and a security context is created for the user.
7. The Single Sign-On Service generates an assertion for the user while also creating an artifact. The artifact contains the source ID of the www.xyz.com SAML responder together with a reference to the assertion (the MessageHandle). The HTTP Artifact binding allows the choice of either HTTP redirection or a HTML form as the delivery mechanism to the service provider. The figure shows the use of the HTML form mechanism. The Single Sign-On Service sends a HTML form back to the

browser. The HTML FORM contains the SAML artifact, the control name being `SAMLart`. Typically the HTML FORM will contain an input or submit action that will result in an HTTP POST.

8. On receiving the HTTP message, the Assertion Consumer service, extracts the SourceID from the SAML artifact. A mapping between source IDs and remote Responders will already have been established administratively. The Assertion Consumer service will therefore know that it has to contact the www.xyz.com SAML responder at the prescribed URL.
9. The www.abc.com Assertion Consumer service will send a SAML `<ArtifactResolve>` message to the identity provider's SAML responder containing the artifact supplied by the identity provider.
10. The SAML responder supplies back a SAML `<ArtifactResponse>` message containing the assertion previously generated. In most implementations, if a valid assertion is received back, then a session on www.abc.com is established for the user (the relying party) at this point.
11. Typically the Assertion Consumer service then sends a redirection message containing a cookie back to the browser. The cookie identifies the session (use of a cookie is implementation specific, other techniques to maintain the security context at the SP can be used). The browser then processes the redirect message and issues an HTTP GET to the TARGET resource on www.abc.com. The GET message contains the cookie supplied back by the Assertion Consumer service. An access check is then back to established whether the user has the correct authorization to access the www.abc.com web site and the index.asp resource.

4.1.7 IdP initiated: POST binding

In this use case the user has a security context on the identity provider (www.xyz.com) and wishes to access a resource at a service provider (www.abc.com). The SAML assertion is transported to the service provider using the HTTP POST binding.

Figure 20 shows the process flow:

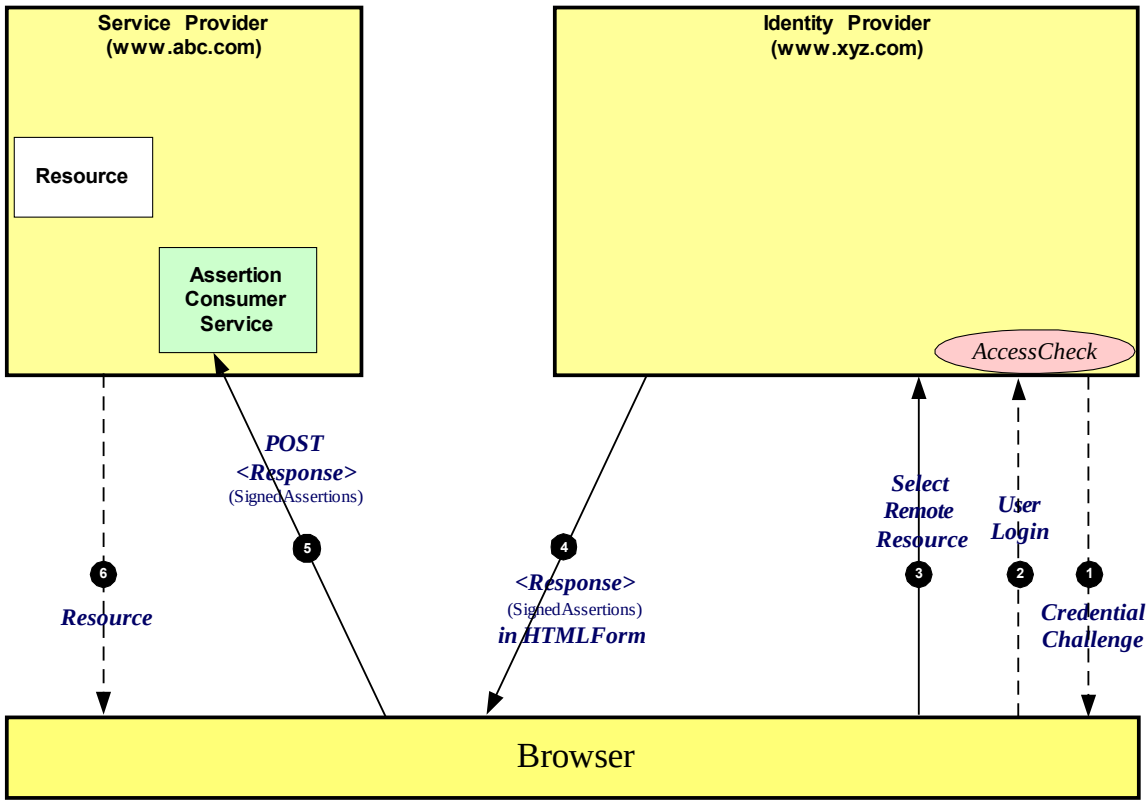


Figure 20: IdP initiated: POST binding

The processing is as follows:

1. At some point the user will have been challenged to supply their credentials to the site www.xyz.com.
2. The user successfully provides their credentials and has a security context with the identity provider.
3. The user selects a menu option (or function) on the displayed screen that means the user wants to access a resource or application on another web site www.xyz.com.
4. The SP sends a HTML form back to the browser. The HTML FORM contains a SAML response, within which is a SAML assertion. The SAML specification mandates that the response must be digitally signed. Typically the HTML FORM will contain an input or submit action that will result in an HTTP POST.
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 Service provider's Assertion Consumer service.
6. The service provider's Assertion Consumer service validates the digital signature on the SAML Response. If this, and the Assertion validate correctly, it sends an HTTP redirect to the browser causing it to access the TARGET resource, withing with a cookie that identifies the local session (use of a cookie is implementation specific, other techniques to maintain the security context at the SP can be used). An access check is then made to establish whether the user has the correct authorization to access the www.abc.com web site and the TARGET resource. If the access check passes, the TARGET resource is then returned to the browser.

4.1.8 IdP initiated: Artifact binding

In this use case the user has a security context on the identity provider (www.xyz.com) and wishes to access a resource at a service provider (www.abc.com). An artifact is provided to the service provider, which it can use (e.g. "de-reference") to obtain the associated SAML response from the identity provider.

Figure 21 shows the process flow:

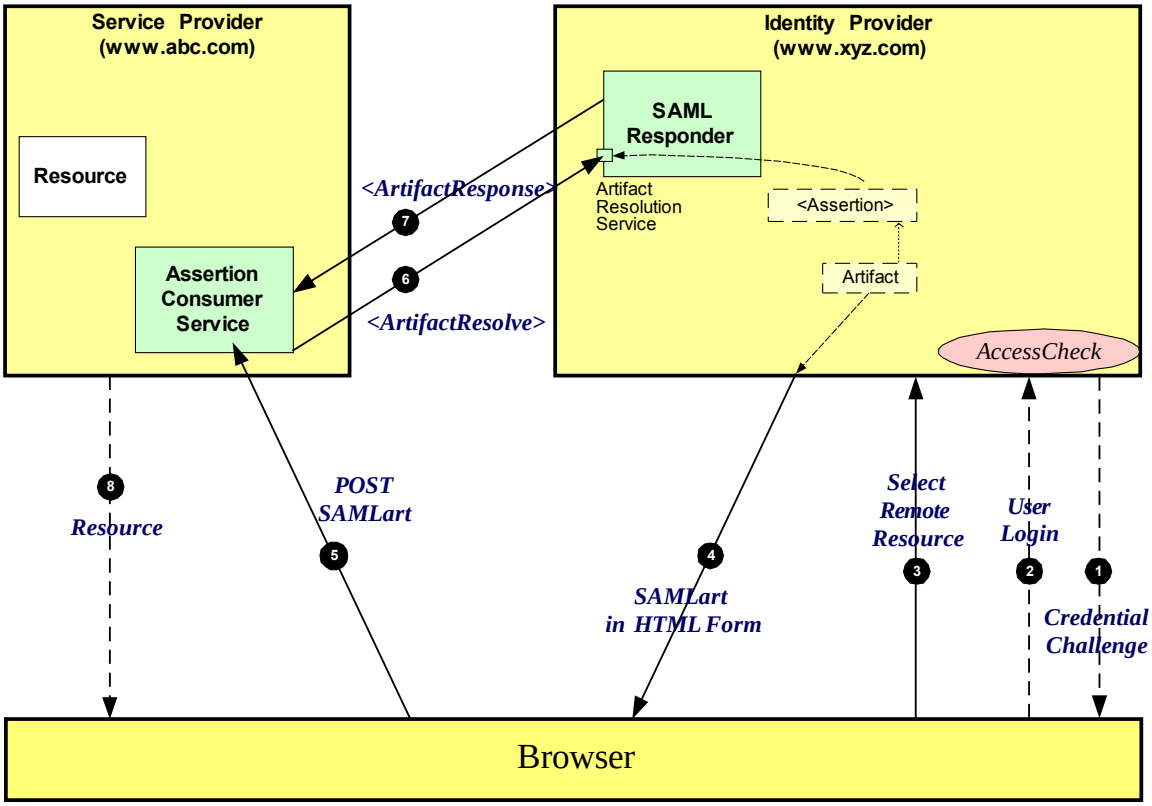


Figure 21: IdP initiated: Artifact binding

The processing is as follows:

1. At some point the user will have been challenged to supply their credentials to the site www.xyz.com.
2. The user successfully provides their credentials and has a security context with the identity provider.
3. The user selects a menu option (or function) on the displayed screen that means the user wants to access a resource or application on a destination web site www.abc.com.
4. The SP generates an assertion for the user while also creating an artifact. The artifact contains the source ID of the www.xyz.com SAML responder together with a reference to the assertion (the MessageHandle). The HTTP Artifact binding allows the choice of either HTTP redirection or a HTML form as the delivery mechanism to the service provider. The figure shows the use of the HTML form mechanism. The service provider sends a HTML form back to the browser. The HTML FORM contains the SAML artifact, the control name being `SAMLart`. Typically the HTML FORM will contain an input or submit action that will result in an HTTP POST.
5. On receiving the HTTP message, the Assertion Consumer service, extracts the SourceID from the SAML artifact. A mapping between source IDs and remote Responders will already have been established administratively. The Assertion Consumer service will therefore know that it has to contact the www.xyz.com SAML responder at the prescribed URL.
6. The www.abc.com Assertion Consumer service will send a SAML `<ArtifactResolve>` message to the identity provider's SAML responder containing the artifact supplied by its service provider.
7. The SAML responder supplies back a SAML `<ArtifactResponse>` message containing the assertion previously generated. In most implementations, if a valid assertion is received back, then a session on www.abc.com is established for the user (the relying party) at this point.
8. Typically the Assertion Consumer service then sends a redirection message containing a cookie back to the browser. The cookie identifies the session (use of a cookie is implementation specific, other techniques to maintain the security context at the SP can be used). The browser then processes the redirect message and issues an HTTP GET to the TARGET resource on www.abc.com. The GET message contains the cookie supplied back by the Assertion Consumer service. An access check is then back to established whether the user has the correct authorization to access the www.abc.com web site and the index.asp resource.

4.2 ECP Profile

4.2.1 Introduction

The Enhanced Client and Proxy (ECP) Profile supports several 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 22 illustrates two use cases for using the ECP Profile.
-
-

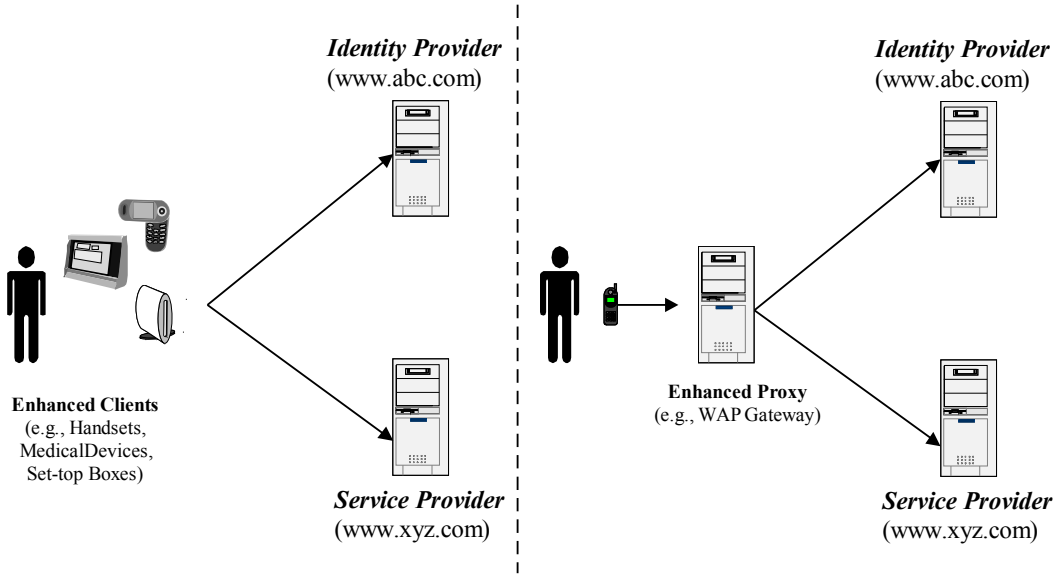


Figure 22: ECP use cases

The ECP profile defines a single binding – PAOS (Reserve 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 23 shows the message flows between the ECP, service provider and identity provider. The ECP is shown as a single logical entity.

Figure 23: ECP with PAOS

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 Identity Federation Protocols

4.3.1 Introduction

This section provides details of a number of use cases when identities are federated. The following use cases are described:

- ***Single Sign-on with Out-of-Band Account Linking:*** Not a true example of federation but a worth while example of what is required to be established if only the Single Sign-On features of SAML are used.
- ***Attribute Federation:*** Attributes of the principal, as defined by the identity provider, are used to link to the account used at the service provider.
- ***Persistent Federation:*** an identity provider federates the identity provider's principal with the principal's identity at the service provider using a persistent ID.
- ***Transient Federation:*** a transient ID is used to federate between the IdP and the SP.
- ***Federation Termination:*** termination of a Federation

To simplify the examples not each permutation of the bindings are illustrated.

All the examples are based on the use case scenarios originally defined in section 2, with *AirlineInc.com* being the identity provider.

4.3.2 Single Sign-On with Out-of-band Account Linking

In this example the same user, joe, has accounts on both *AirlineInc.com* and *CarRentalInc.com* each with the same user name (***joe***). The identity stores at both sites are synchronized by some means, for example either via database synchronization or off-line batch updates. This example purely illustrates the support for Single Sign-On by SAML. This form of account linking uses persistent identifiers.

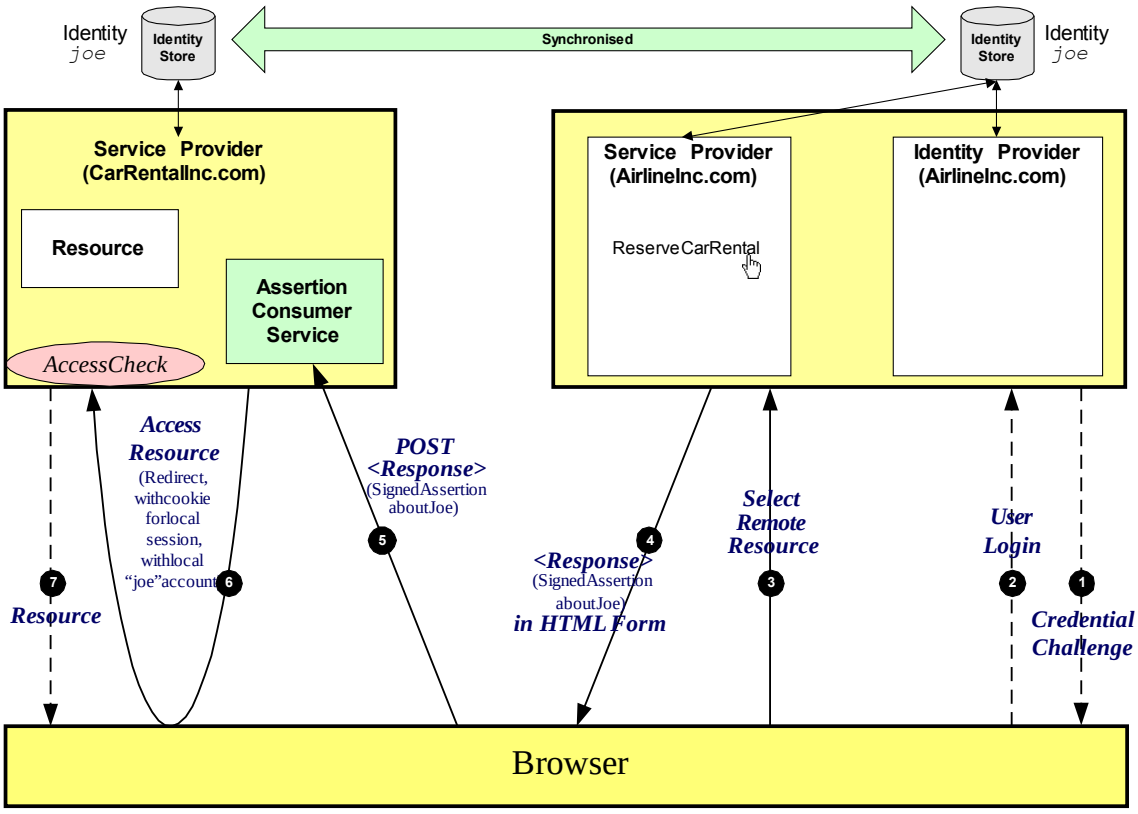


Figure 24: Single Sign-On with Out-of Band account linking

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*.
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 ***joe***.
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 for user ***joe***, based on the local joe 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 ***joe*** has the correct authorization to access the *CarRentalInc.com* web site and the TARGET resource. The TARGET resource is then returned to the browser.

4.3.3 Attribute Federation

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.

[@@Add 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 25]

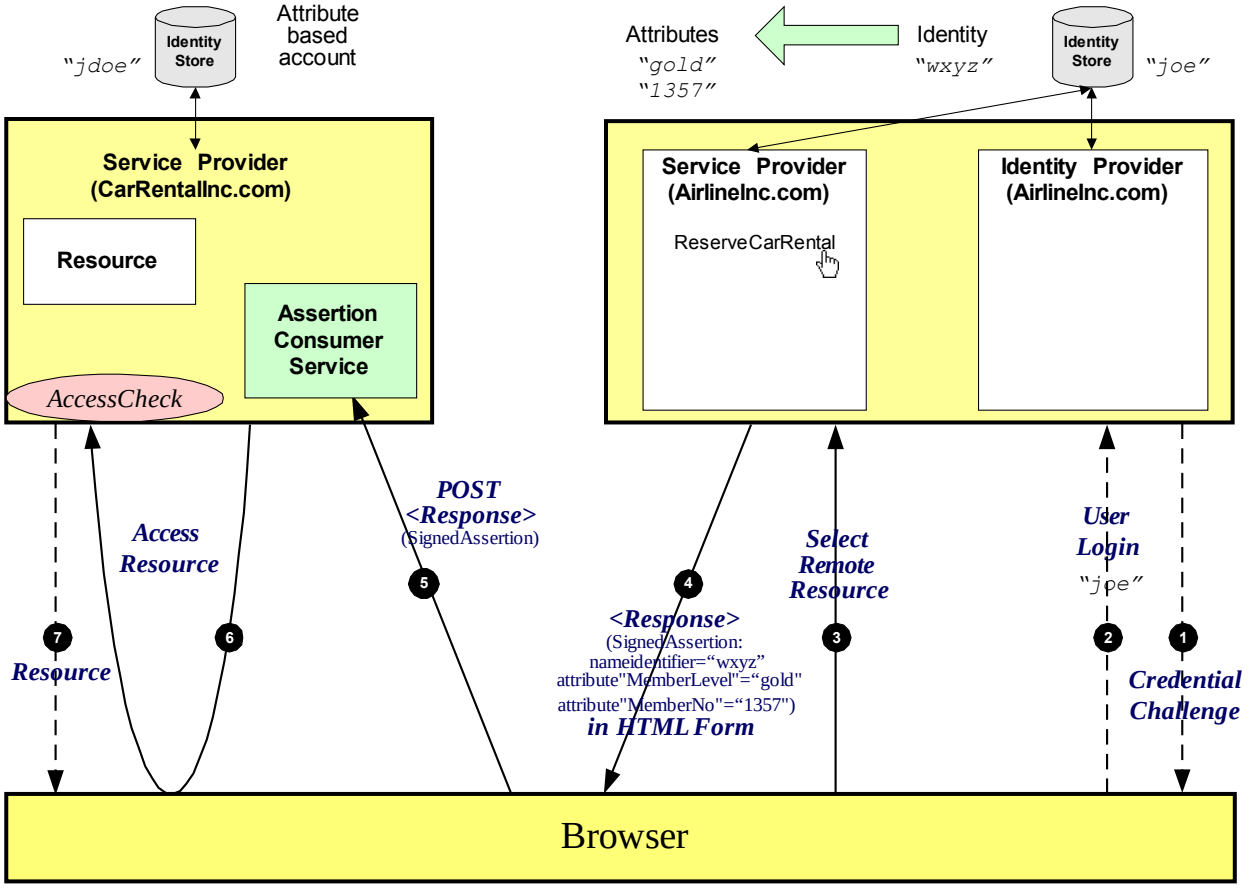


Figure 25: Attribute Federation

In this example 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, the user named supplied is ***joe***.
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 ***joe***. 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 ***joe*** 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.

4.3.4 Persistent Federation

This Federation example is similar to the previous one, except in this case the identity provider provides to the service provider an assertion with a persistent name identifier using a `<Response>`. For the following set of examples will shall illustrate the information maintained at both the IdP and the SP. In all cases the user ***joe*** on *AirlineInc.com* wishes to federate this account with his ***jdoe*** account on *CarRentalInc.com* . There are two use cases that could occur, firstly the user accesses a resource on *CarRentalInc.com* and is then redirected to *AirlineInc.com,* secondly the user is accessing a resource on *AirlineInc.com* and is directed to *CarRentalInc.com*. The former we refer to as being SP-initiated the later IdP-initiated.

Figure 26 illustrates Persistent Federation when it is SP-initiated.

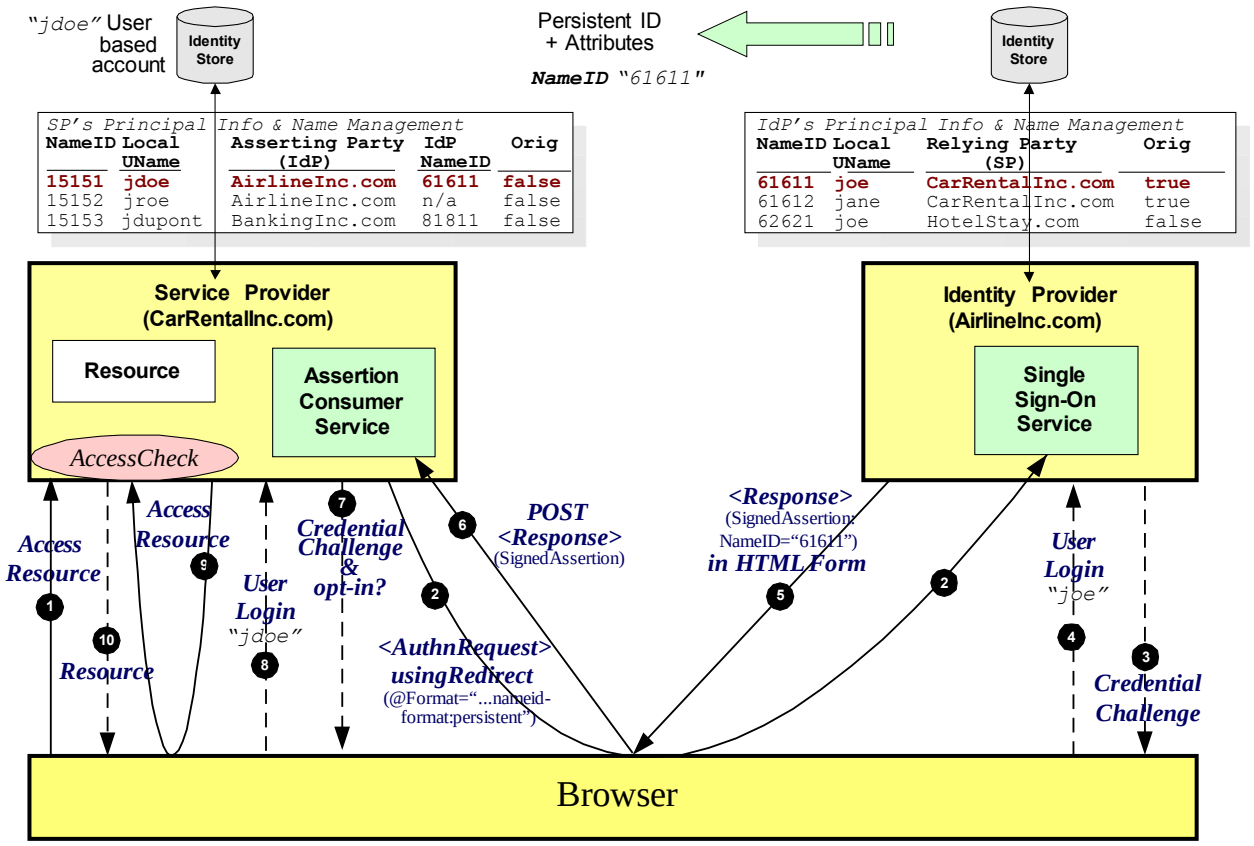


Figure 26: Persistent Federation – SP-initiated

The processing is as follows:

1. The user attempt 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.
2. The service provider sends a HTTP redirect to the identity provider (*AirlineInc.com*). The HTTP redirect contains a SAML `<AuthnRequest>` requesting that the identity provider provide an assertion about the requesting user. The request asks that the identity provider sends back a persistent identifier.
3. The user will be challenged to provide valid credentials.
4. The user provides valid credentials and a security context is created for the user. The user identifies themselves as ***joe***. The identity provider looks up user ***joe*** in its identity store and determines the persistent name identifier to be used for this federation (`61611`).
5. The Single Sign-On Service sends a HTML form back to the browser. The HTML FORM contains a SAML response, within which is a SAML assertion. The name identifier used in the assertion is a persistent identifier. The attribute "gold member" and a membership number attribute are provided. The name ***joe*** is not contained anywhere in the assertion.
6. 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 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 look up to establish 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 is in existence then the user will be challenged to provide valid credentials. Optionally the user could be asked whether he would like to federate the two accounts.
8. The user provides valid credentials and identifies themselves as ***jdoe***. The persistent name identifier is then stored and registered with the ***jdoe*** account along with who the identity provider is.

9. A session created is for user ***jdoe*** and 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
10. If the access check passes, the TARGET resource is then returned to the browser.

The second use case is when a user accesses a resource on the identity provider (*AirlineInc.com*) that points to a resource on *CarRentalInc.com.* Figure 27 illustrates this use case.

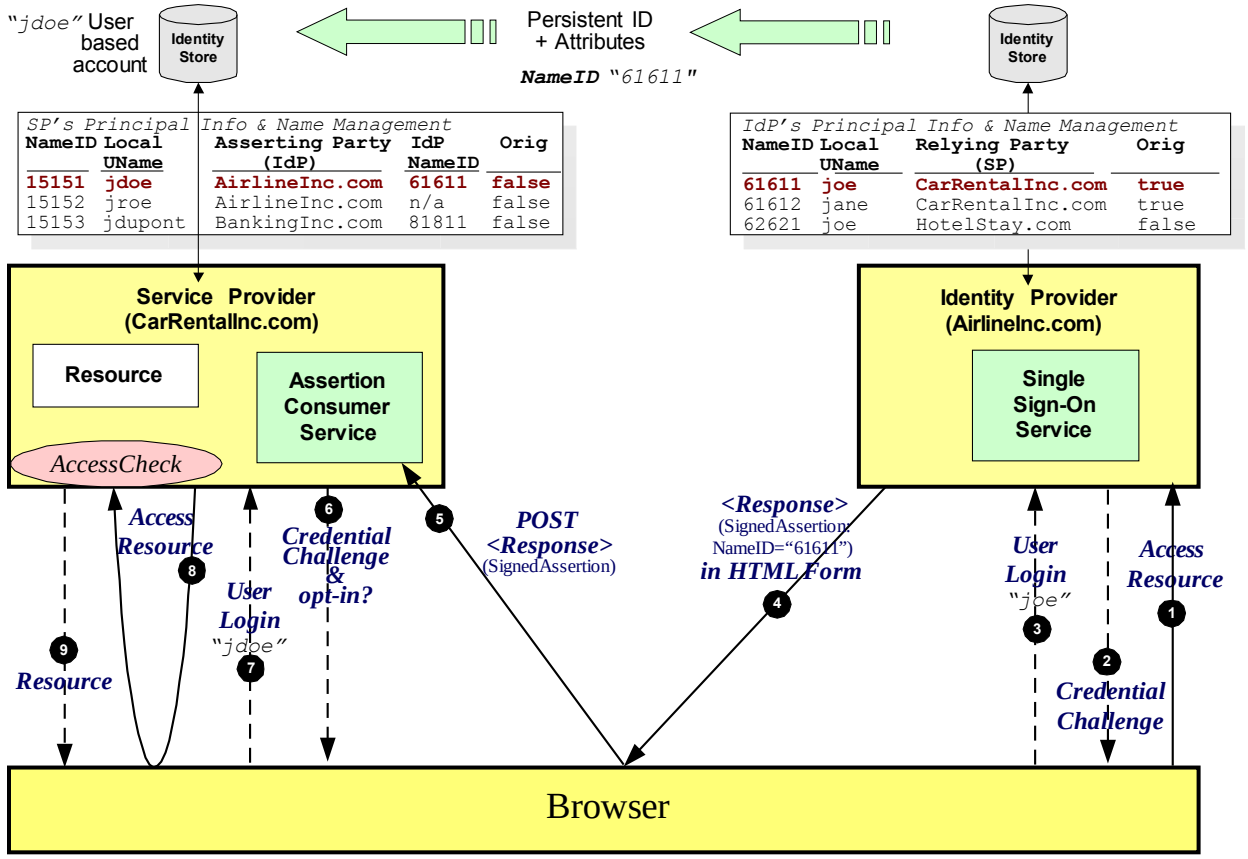


Figure 27: Persistent Federation – IdP-initiated

The processing is as follows:

1. The user attempt to access a resource on *AirlineInc.com* which eventually will pass them over to a resource on *CarRentalInc.com* .
2. If the user does not have a current security context they will be challenged to provide valid credentials.
3. The user provides valid credentials and a security context is created for the user. The user identifies themselves as ***joe***. The identity provider looks up user ***joe*** in its identity store and determines the persistent name identifier to be used for this federation (`61611`).
4. The Single Sign-On Service sends a HTML form back to the browser. The HTML FORM contains a SAML response, within which is a SAML assertion. The name identifier used in the assertion is a persistent identifier. The attribute "gold member" and a membership number attribute are provided. The name ***joe*** 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 service provider's Assertion Consumer service.
6. 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 look up to establish 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 is in existence then the user will be challenged to provide valid credentials. Optionally the user could be asked whether he would like to federate the two accounts.

7. The user provides valid credentials and identifies themselves as ***jdoe***. The persistent name identifier is then stored and registered with the ***jdoe*** account along with who the identity provider is.
8. A session created is for user ***jdoe*** and 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
9. If the access check passes, the TARGET resource is then returned to the browser.

4.3.5 Transient Federation

The previous use cases showed the use of persistent identifiers, what if you do want to establish a permanent federation. This is where the use of transient identifiers are useful. Transient identifiers allow you to:

- avoid having to manage userids and passwords at the service provider. Therefore all authentication is performed at the identity 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 case one can have SP and IdP-initiated variations. Figure 28 shows the SP-initiated use case

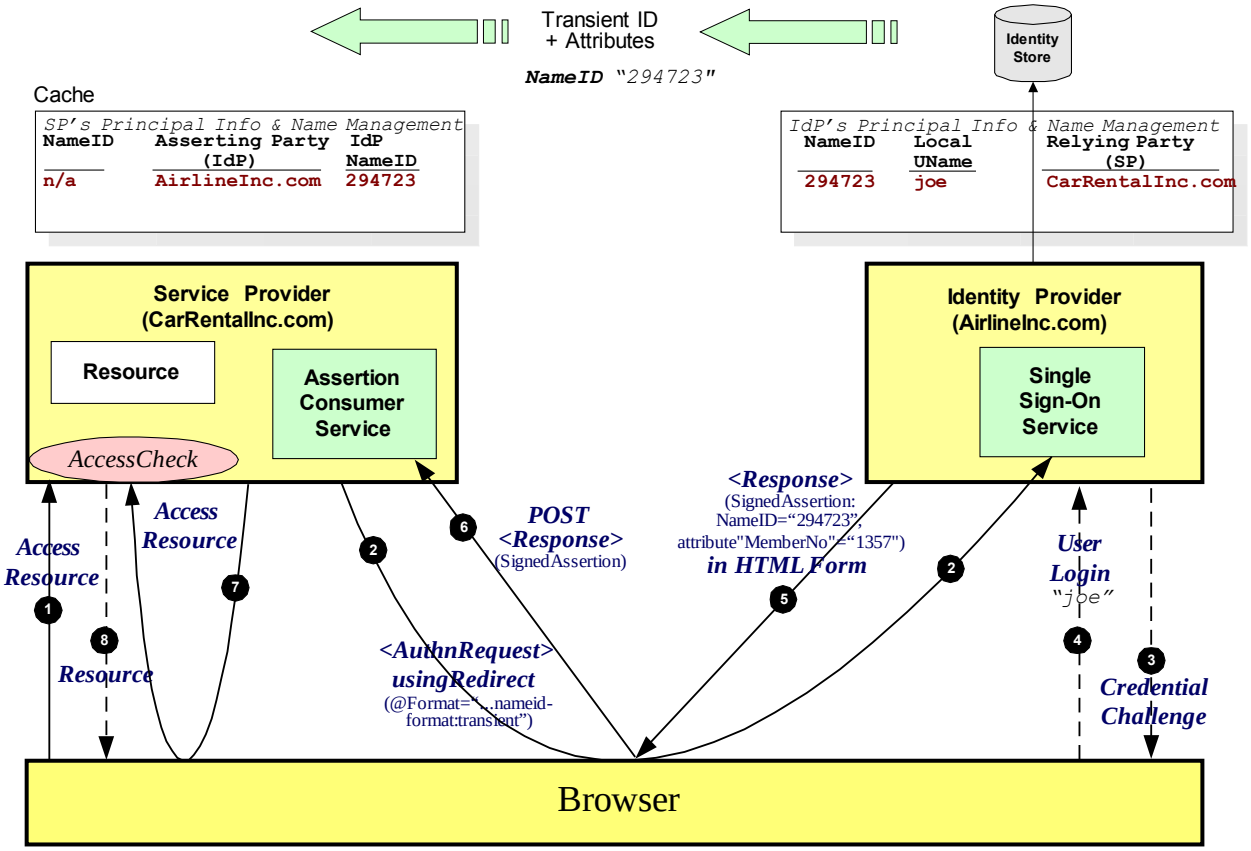


Figure 28: Transient Federation – SP-initiated

The processing is as follows:

1. The user attempt 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.
2. The service provider sends a HTTP redirect to the identity provider (*AirlineInc.com*). The HTTP redirect contains a SAML `<AuthnRequest>` requesting that the identity provider provide an assertion about the requesting user. The request asks that the identity provider sends back a transient identifier.
3. The user will be challenged to provide valid credentials.
4. The user provides valid credentials and a security context is created for the user. The user identifies themselves as ***joe***. The identity provider looks up user ***joe*** in its identity store and creates a transient name identifier to be used for this federation (`294723`).
5. The Single Sign-On Service sends a HTML form back to the browser. The HTML FORM contains a SAML response, within which is a SAML assertion. The name identifier used in the assertion is a transients identifier. The attribute “gold member” and a membership number attribute (`1357`) are provided. The name ***joe*** is not contained anywhere in the assertion.
6. 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 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 dynamically create a session based in the received assertion. In this example it could be the membership number attribute which maps to the ***jdoe*** account. A session created is for user ***jdoe*** and 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.
8. If the access check passes, the TARGET resource is then returned to the browser.

The IdP-initiated use case is shown in figure 29.

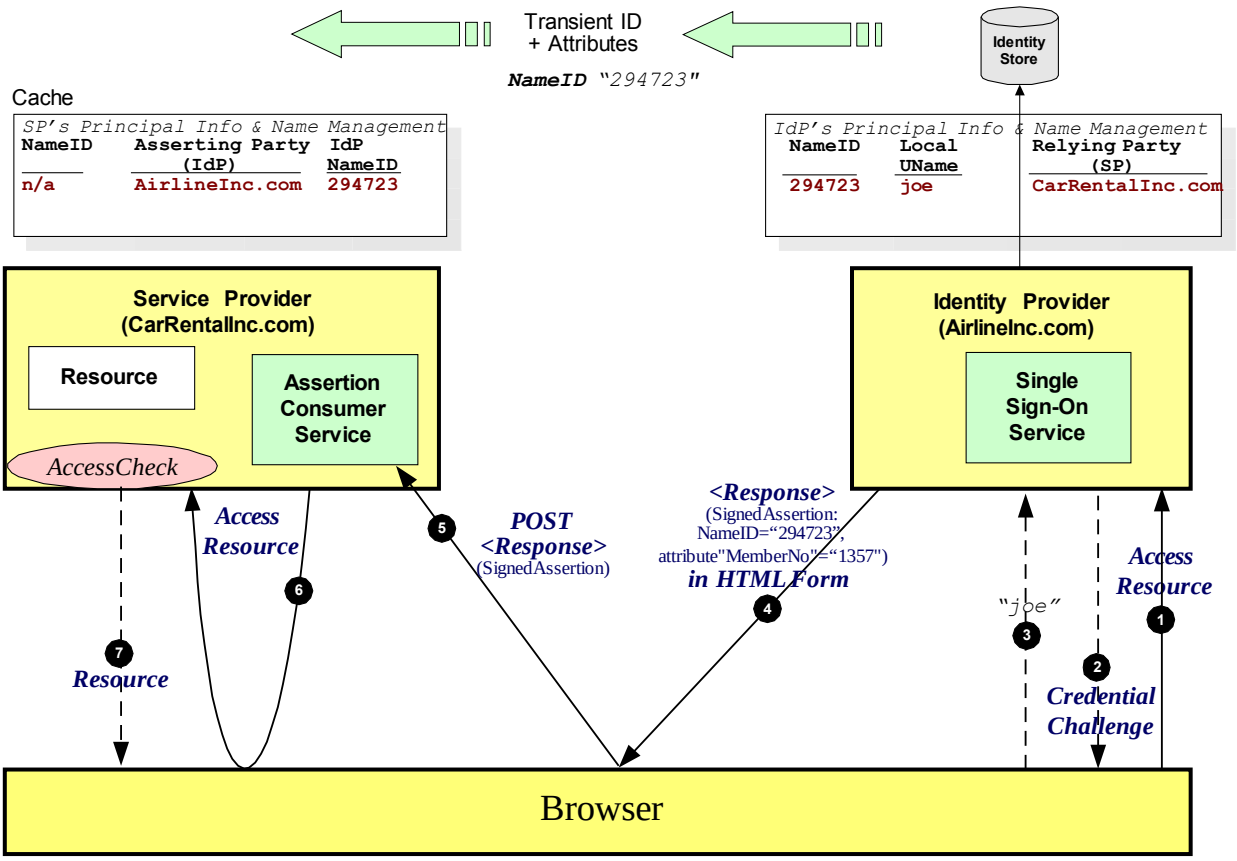


Figure 29: Transient Federation – IdP-initiated

The processing is as follows:

1. The user attempt to access a resource on *AirlineInc.com* which eventually will pass them over to a resource on *CarRentalInc.com* .
2. If the user does not have a current security context they will be challenged to provide valid credentials.
3. The user provides valid credentials and a security context is created for the user. The user identifies themselves as ***joe***. The identity provider looks up user ***joe*** in its identity store and creates a transient name identifier to be used for this federation (`294723`).
4. The Single Sign-On Service sends a HTML form back to the browser. The HTML FORM contains a SAML response, within which is a SAML assertion. The name identifier used in the assertion is a transient identifier. The attribute "gold member" and a membership number attribute (`1357`) are provided. The name ***joe*** 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 service provider's Assertion Consumer service.
6. 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 dynamically create a session based in the received assertion. In this example it could be the membership number attribute which maps to the ***jdoe*** account. A session created is for user ***jdoe*** and 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.
7. If the access check passes, the TARGET resource is then returned to the browser.

4.3.6 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 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.

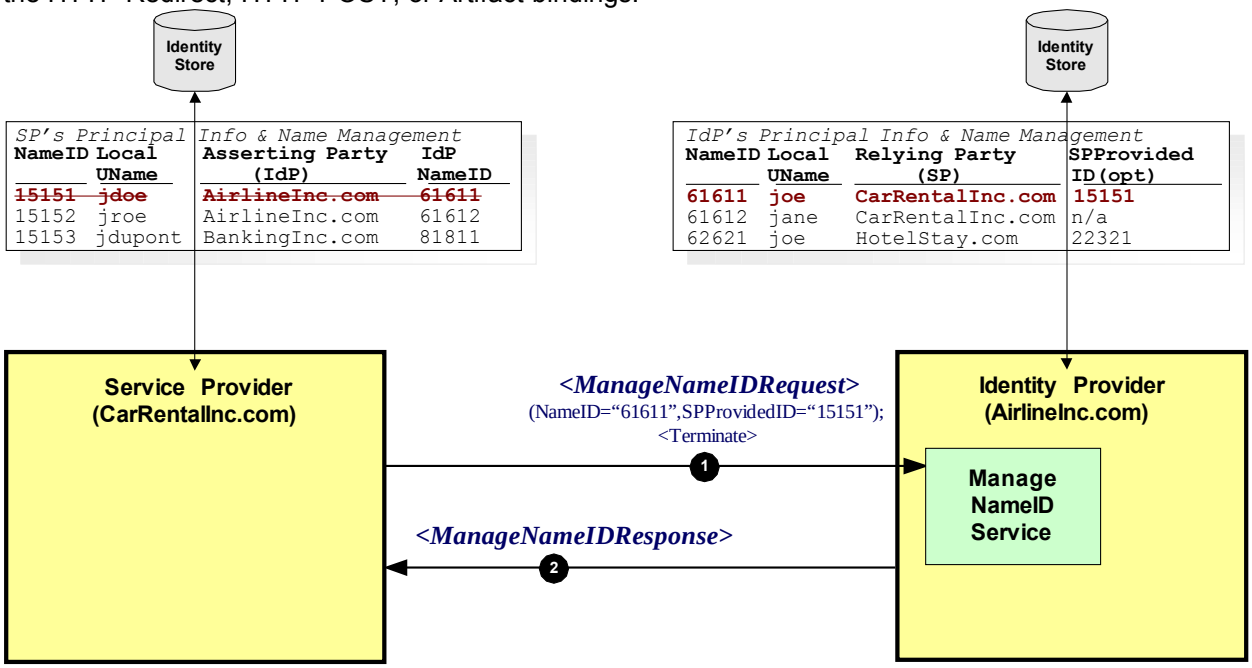


Figure 30: Federation Termination

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.

4.4 Single Logout

Single Logout permits near real-time session logout of all participants in a session. A request can be issued by any session participant to request that the session is to be finished. In this example a user on the *CarRentalInc.com* service provider decides that they wish to logout out of the session.

The example shows the use of the SOAP over HTTP binding, however asynchronous front-channel bindings can also be used.

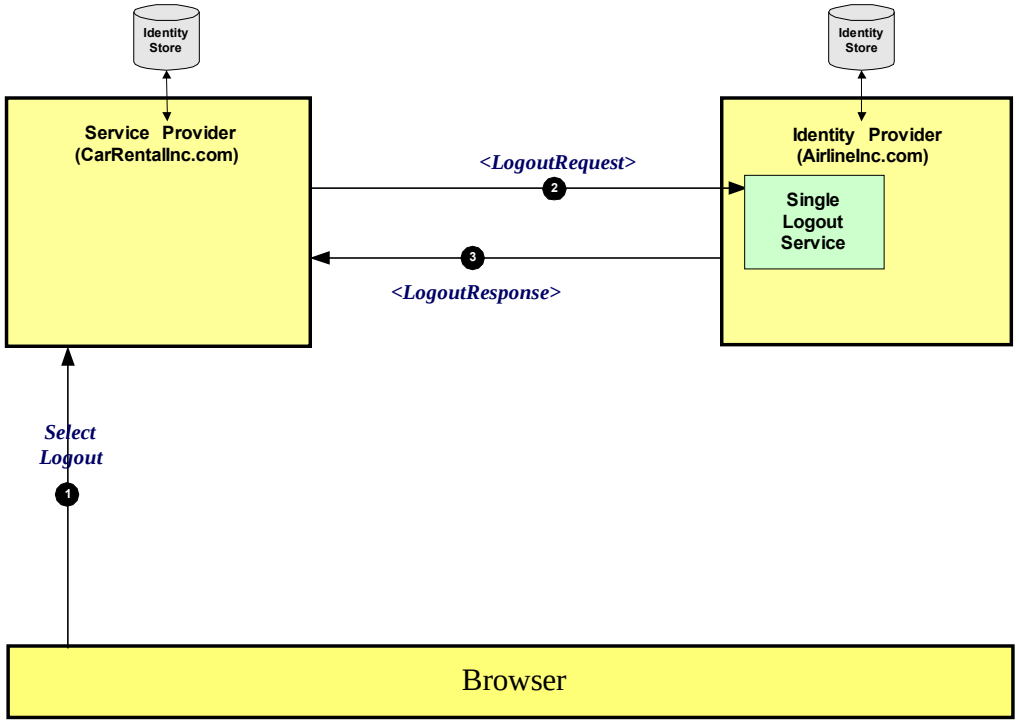


Figure 31: Single Logout

1237

1238 The processing is as follows:

1239 1. A user was previously authenticated by the *AirlineInc.com* identity provider and is interacting with the
1240 *CarRentalInc.com* service provider. The user decides to terminate their session and logout.

1241 2. The service provider, sends to the *AirlineInc.com* identity provider a `<LogoutRequest>` defining
1242 that the session is to be logged out. The request identifies the principal to be logged out, by using the
1243 `<NameID>` element, as well as providing a `<SessionIndex>` element to uniquely identify the
1244 session being closed. The request is carried in a SOAP message which is transported using HTTP,
1245 as defined by the SAML SOAP binding. The request is also digitally signed by the service provider.

1246 3. The identity provider verifies the digital signature ensuring that the `<LogoutRequest>` originated
1247 from a known and trusted service provider. The identity Provider processes the request and returns
1248 a `<LogoutResponse>` containing a suitable status code response. The response is carried within a
1249 SOAP over HTTP message and is digitally signed.

1250

1251 If in step 3 the identity provider determines that other service providers are participants in the session,
1252 then the identity provider will send `<LogoutRequest>` messages to them. Figure 32 illustrates this
1253 processing. Notice that different bindings are used between the two different exchanges with the service
1254 providers. One using the redirect binding the other using a back channel, illustrating the point that
1255 different combinations of bindings can be used.

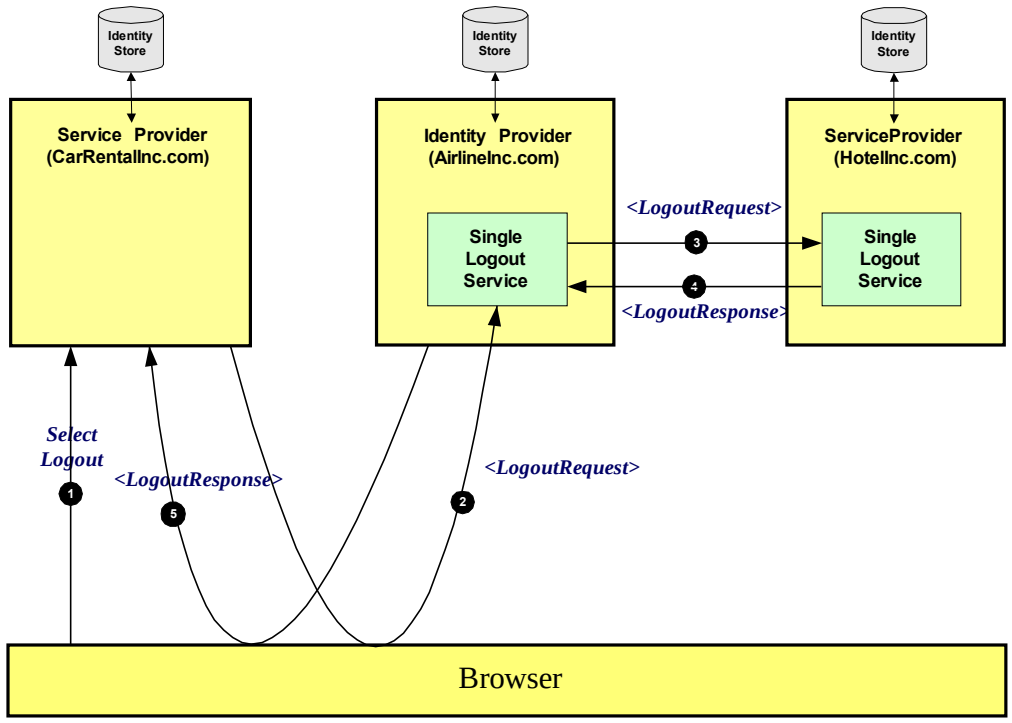


Figure 32: Multiple Logouts

1256 The two previous examples showed the user instigating the logout. Of course the service provider itself
1257 could initiate the logout, and in that case step 1 would not occur. There is one other use case possible,
1258 and that is when the identity provider initiates the logout. Figure 33 illustrates this example.

1259

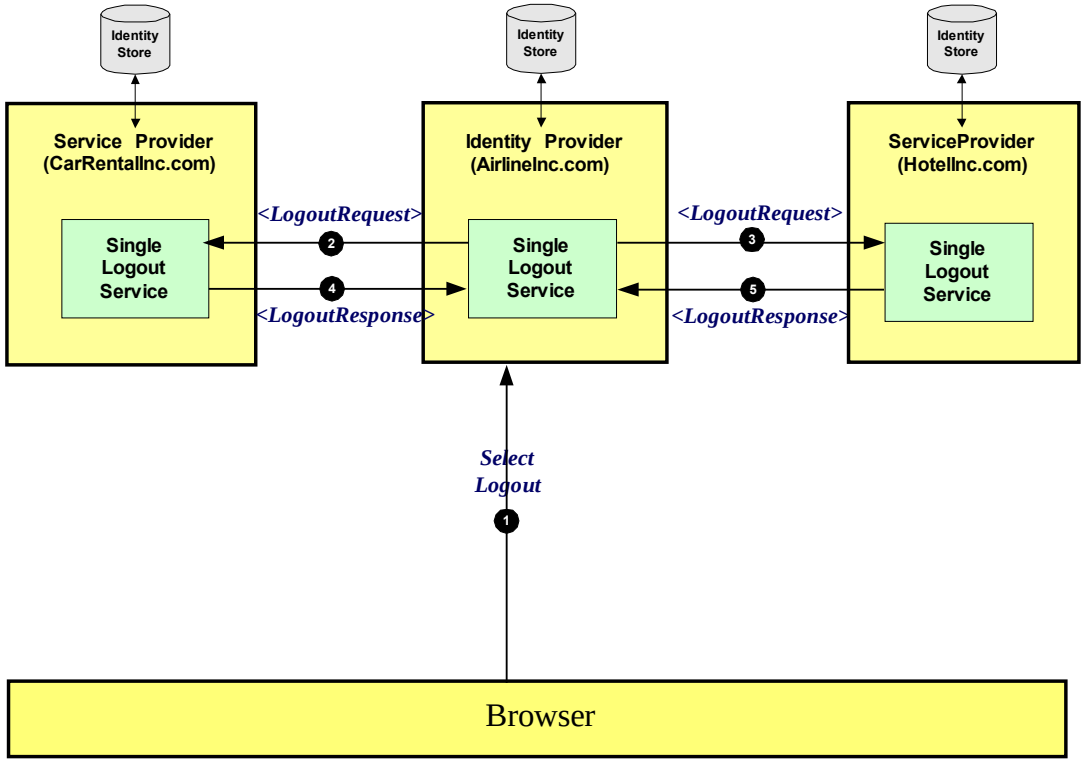


Figure 33: Multiple Logouts – identity provider initiated

5 Documentation roadmap

- **Security Assertion Markup Language (SAML) V2.0 Executive Overview**. (sstc-saml-exec-overview-2.0) Provides a brief overview of SAML and describes its primary benefits.
- **Security Assertion Markup Language (SAML) V2.0 Technical Overview**. (sstc-saml-tech-overview-2.0). This document.
- **Assertions and Protocol for the OASIS Security Assertions Markup Language (SAML) V2.0** (sstc-saml-core-2.0). Defines the syntax and semantics for XML-encoded assertions about authentication, attributes and authorization, and for the protocol that conveys this information.
- **Security and Privacy Considerations for the OASIS Security Assertions Markup Language (SAML) V2.0** (sstc-saml-sec-consider-2.0). Describes and analyzes the security and privacy properties of SAML
- **Bindings for the OASIS Security Assertions Markup Language (SAML) V2.0 (**sstc-saml-bindings-2.0). Defines protocol bindings for the use of SAML assertions and request-response messages in communications protocols and frameworks.
- **Profiles for the OASIS Security Assertions Markup Language (SAML) V2.0 (**sstc-saml-profiles-2.0). Defines how the assertions, protocols and bindings combine to define specific profiles.
- **Conformance Program Specification for the OASIS Security Assertions Markup Language (SAML) V2.0** (sstc-saml-conform-2.0). Describes the program and technical requirements for SAML conformance.
- **Metadata for the OASIS Security Assertions Markup Language (SAML) V2.0** (sstc-saml-metadata-2.0). Describes metadata format to enable configuration data to be shared in a standardized format.
- **Glossary for the OASIS Security Assertions Markup Language (SAML) V2.0** (sstc-saml-glossary-2.0). Defines terms used throughout the OASIS Security Assertion Markup Language (SAML) specifications.
- **Authentication Context for the OASIS Security Assertions Markup Language (SAML) V2.0** (sstc-saml-authn-context-2.0). Defines a syntax for the definition of authentication context declarations.

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

Note that this appendix contains information that is known to be out of date; it only covers differences through about core-10 in most cases. To be updated soon with other differences.

SAML V2.0 constitutes a large-scale realization of features derived from the Liberty Alliance Identity Federation Framework (ID-FF) V1.2 specifications that were contributed to the SSTC in 2003, along with other requested features, improvements, and streamlining.

The on-the-wire representations of SAML V2.0 assertions and 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. However, most such incompatibility is syntactic in nature; the expressiveness of SAML has increased rather than markedly changed.

The differences are described in the sections below. Note that these descriptions may not be complete; for a full accounting of precise differences to SAML V1.1 specification text, see [some change-bar version of specs that doesn't exist yet].

6.1 Differences in the Organization of the Specifications

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

6.2 Versioning Differences

- The SAML assertions namespace (known by its convention 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 changed to a single `Version` attribute that must have the value "2.0".
- A series of changes planned during SAML the 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 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.

6.3 Subject and Subject Confirmation Differences

- The `<SubjectStatement>` element and its type have been removed.
- The `<Subject>` element has been moved up to appear on the `<Assertion>` element, where the

subject so specified applies to all enclosed statements. The <Subject> element is now optional for extensibility reasons, but is required for all SAML-specified statement types.

- The new **BaseID** complex type is an extension point that permits non-string identification of subjects.
- The <SubjectConfirmation> element is now repeatable, with the formerly repeatable <ConfirmationMethod> element now an attribute within that element.
- The <ds:KeyInfo> element is now allowed only inside <SubjectConfirmationData>. Further, the usage of <ds:KeyInfo> within <SubjectConformationData> has been clarified to more clearly allow for impersonation.
- A set of generic attributes in <SubjectConfirmationData> have been defined for use in constraining the bearer method or other confirmation methods. Overall assertion validity is more flexible within profiles that use bearer as a result.

6.4 Encryption-Related Differences

- The name identifier structure, the attribute structure, and the assertion structure have all been refactored to allow encryption.

6.5 Attribute-Related Differences

- The AttributeNamespace field has been removed in favor of NameFormat, and two new URI-based identifiers of attribute name format types have been defined for use in this field. This field can be left blank, as a default has been defined.
- The name of the AttributeName field has been changed to just Name.
- 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 attributes specified using common attribute/directory technologies.

6.6 Differences in the Request-Response Mechanism

- 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.
- The Issuer field is now an element and is based on the same datatype that underlies name identifiers, for more unified treatment. Also, in addition to appearing on assertions, it now appears on requests and responses as well.
- 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 statuses when using the new protocols. Status codes are now URIs instead of QNames.

6.7 Differences in the Protocols for Retrieving Assertions

- Instead of the <AssertionIDReference> in <Request>, the <AssertionIDRequest> element is now used to get an assertion by means of its ID.
- Instead of the <AssertionArtifact> element to retrieve assertions in a response message, now a special <ArtifactResolve> protocol is used to get SAML protocol messages by means of an artifact. All types of protocol messages can theoretically be retrieved in this fashion, but in practice only some kinds will appear in profiles.

6.8 Session-Related Differences

- The <AuthnStatement> element can now contain a SessionIndex attribute in support of single logout and other session management requirements.
- There is a new single logout protocol for near-simultaneous logout from multiple related sessions.

6.9 Federation-Related Differences

- There is a new protocol for requesting that authentication be performed and a new assertion with an authentication statement returned. As part of this, the policy for the desired form of name identifier can be specified.
- In such an assertion, it is now possible to specify many more details about the authentication that was performed using the new authentication context schemas; the old AuthenticationMethod field has been removed.
- There is a new federated name management (registration and deregistration) protocol.
- There is a new name identifier mapping protocol.

6.10 Differences in Bindings and Profiles

- A lot of profile detail has been refactored out to become new, more generic bindings; the profiles are much thinner. For example, there's now an HTTP redirect/HTTP POST binding.
- There is a new HTTP-based binding added for retrieval of assertions by means of URIs.
- A PAOS (reverse SOAP) binding has been added.
- An enhanced client profile has been added.
- The two original browser profiles (browser/artifact and browser/POST) have become a single web SSO profile.
- A set of mechanisms for relaying state have been added to most of the bindings.
- As noted above, a series of attribute profiles has now been defined.

6.11 Other Differences

- A number of elements, attributes, and types have been renamed for brevity and consistency. List them
- 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 attributes.
- The notion of special “SAML namespaces” (attribute namespaces and action namespaces) has been deemphasized, with attribute namespaces being removed entirely in favor of URIs as attribute format identifiers.
- The <ds:Signature> that allows for the digital signing of assertions and messages has been positioned earlier in the respective content models.
- The authorization decision feature (statement and query) has been frozen; if more functionality is desired, it is suggested that XACML [XACML] be used.
- Two new conditions, <ProxyRestriction> element and <OneTimeUse>, have been added. The relationship of the latter to the NotBefore and NotOnOrAfter conditions has been delineated.
- 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.

TBS: validity period semantics and syntax extended, removal of QNames in content, etc.

7 References

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

[SAMLBind] S. Cantor et al., *Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC. Document ID sstc-saml-bindings-2.0-os. See http://www.oasis-open.org/committees/security/.

[SAMLConform] P. Mishra et al. *Conformance Requirements for the OASIS Security Assertion Markup Language (SAML) V2.0.* OASIS SSTC. Document ID sstc-saml-conformance-2.0-os. http://www.oasis-open.org/committees/security/.

[SAMLCore] S. Cantor et al., *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC. Document ID sstc-saml-core-2.0-os. 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. Document ID sstc-saml-glossary-2.0-os. 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. Document ID sstc-saml-metadata-2.0-os. 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. Document ID sstc-saml-sec-consider-2.0-os. See http://www.oasis-open.org/committees/security/.

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

[WSS] A, Nadalin, ed., *OASIS Web Services Security: SOAP Message Security 1.0.* Available on the OASIS WS-Security web page at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

[WSSSAML] R. Monzillo, ed. OASIS Web Services Security: SAML 2.0 Token Profile 1.0. wss-saml-2.0-token-profile-1.0. Available on the OASIS XACML TC web page at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

[XACML] T. Moses, ed., *OASIS eXtensible Access Control Markup Language (XACML) Version 2.0*. Available on the OASIS XACML TC web page at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.

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

1459 # 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)

1460

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.