



Security Assertion Markup Language (SAML) 2.0 Technical Overview

Working Draft 03, 20 February 2005

Document identifier:

sstc-saml-tech-overview-2.0-draft-03

Location:

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

Editors:

John Hughes, Atos Origin
Eve Maler, Sun Microsystems

Contributors:

Hal Lockhart, BEA

Abstract:

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

Status:

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

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

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

Table of Contents

1	Introduction.....	4
2	SAML Use Cases.....	5
2.1	Single Sign-On Use Case.....	5
2.2	Federation Use Case.....	6
3	SAML Architecture.....	7
3.1	Basic Concepts.....	7
3.2	Summary of SAML Components.....	7
3.3	SAML Structure and Examples.....	9
3.3.1	Assertions.....	9
3.3.2	SOAP over HTTP Binding.....	11
3.4	Single Sign-On and Federation Principals.....	12
3.5	Use of SAML in other Frameworks.....	13
3.5.1	Web Services Security (WSS).....	13
3.5.2	eXtensible Access Control Markup Language (XACML).....	15
3.6	Security in SAML.....	16
4	Profiles.....	18
4.1	Web Browser SSO Profile.....	18
4.1.1	Concept.....	18
4.1.2	SP initiated: POST->POST binding.....	20
4.1.3	SP initiated: Redirect->POST binding.....	21
4.1.4	SP initiated: Artifact->POST binding.....	22
4.1.5	SP initiated: POST->Artifact binding.....	23
4.1.6	SP initiated: Redirect->Artifact binding.....	25
4.1.7	SP initiated: Artifact->Artifact binding.....	26
4.1.8	IdP initiated: POST binding.....	28
4.1.9	IdP initiated: Artifact binding.....	29
4.2	ECP Profile.....	30
4.2.1	Introduction.....	30
4.2.2	ECP Profile using PAOS binding.....	30
4.3	Federation.....	31
4.3.1	Introduction.....	31
4.3.2	Federation during <AuthnRequest>.....	32
4.3.3	Federation Termination.....	32
4.3.4	Accounting Linking.....	32
5	Documentation roadmap	33
6	Comparison Between SAML 2.0 and SAML 1.1.....	34
6.1	Differences in the Organization of the Specifications.....	34
6.2	Versioning Differences.....	34
6.3	Subject and Subject Confirmation Differences.....	34
6.4	Encryption-Related Differences.....	35
6.5	Attribute-Related Differences.....	35
6.6	Differences in the Request-Response Mechanism.....	35
6.7	Differences in the Protocols for Retrieving Assertions.....	35
6.8	Session-Related Differences.....	35

77	6.9 Federation-Related Differences.....	35
78	6.10 Differences in Bindings and Profiles.....	36
79	6.11 Other Differences.....	36
80	7 References.....	37
81		

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 SSTC charter, the purpose of the Technical Committee is:

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

SAML is different from other security systems due to its approach of expressing assertions about a subject that other applications within a network can trust. What does this mean? To understand the answer, you need to know the following two concepts used within SAML:

Identity Provider (IdP)

The system, or administrative domain, that asserts information about a subject. For instance, the Identity Provider asserts that this user has been authenticated and has given associated attributes. For example: 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. In SAML, Identity Providers are also known as **SAML authorities** and **Asserting Parties**.

Service Provider (SP)

The system, or administrative domain, that relies on information supplied to it by the Identity Provider. It is up to the Service Provider as to whether it trusts the assertions provided to it. SAML defines a number of mechanisms that enable the Service Provider to trust the assertions provided to it. It should be noted that although a Service Provider can trust the provided assertions provided, local access policy defines whether the subject may access local resources. Therefore, although the Service Provider trusts that I'm **John Doe** – it doesn't mean I'm given carte blanche access to all resources. Service Providers are also known as **Relying Parties** – due to the fact that they “rely” on information provided by an Identity Provider (Asserting Party).

2 SAML Use Cases

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).

More precisely, SAML defines a common XML framework for creating, requesting, and exchanging security assertions between entities. As stated on the SSTC website, the purpose of the Technical Committee is:

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

But why is it required? There are four “drivers” behind the creation of the SAML standard:

- **Limitations of Browser cookies:** Most existing 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 an organization that has separate DNS domains. Therefore, to solve the Cross-Domain SSO (CDSSO) problem requires the application of different technology. All SSO products solve the CDSSO problem by different techniques.
- **SSO Interoperability:** How products implement SSO and CDSSO are completely proprietary. If you are an organization and you want to perform SSO across different DNS domains within the same organization or you want to perform CDSSO to trading partners, then you will have to use the same SSO product in all the domains.
- **Web Services:** Security within Web Services is still being defined. Most of the focus has been on how to provide confidentiality and authentication/integrity services on an end-to-end basis. The SAML standard provides the means by which authentication and authorization assertions can be exchanged between communicating parties.
- **Federation:** The need to simplify identity management across organizational boundaries, allowing users to consolidate many local identities into a single (or at least a reduced set) Federated Identity.

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

2.1 Single Sign-On Use Case

This is the original use case as supported in SAML 1.0 and 1.1. It illustrates the support for Cross Domain Single Sign-On. A user has a logon session (that is a *security context*) on a website (`AirlineInc.com`) and is accessing resources on that site. At some point either explicitly or transparently he is directed over to another web site (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 session attributes (e.g. “Gold member”). As `CarRentalInc.com` trusts `AirlineInc.com` it knows that the user is valid and creates a session for the user based on the user's name and/or the user attributes. 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.

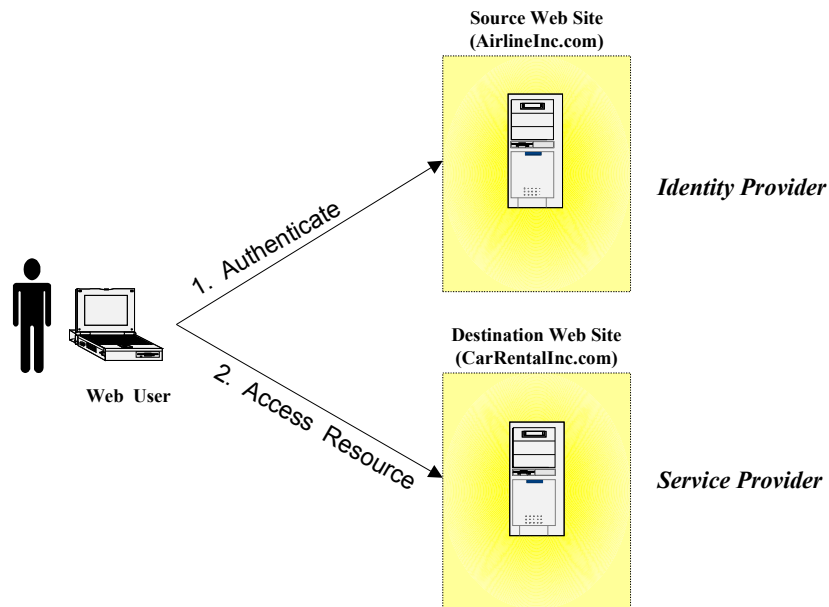


Figure 1: SSO Use Case

2.2 Federation Use Case

There are a number Federation use cases, details of which are explained later. This use case illustrates the “account linking” facet of federation. Figure 2 illustrates one scenario. Two Service Providers exist, one for car rentals the other for hotel bookings. The same user is registered on both sites, however using different names. On CarRentalInc.com he is registered as jdoes and on HotelBookings.com as johnd. Account Linking enables a pseudonym to be established that links the two accounts.

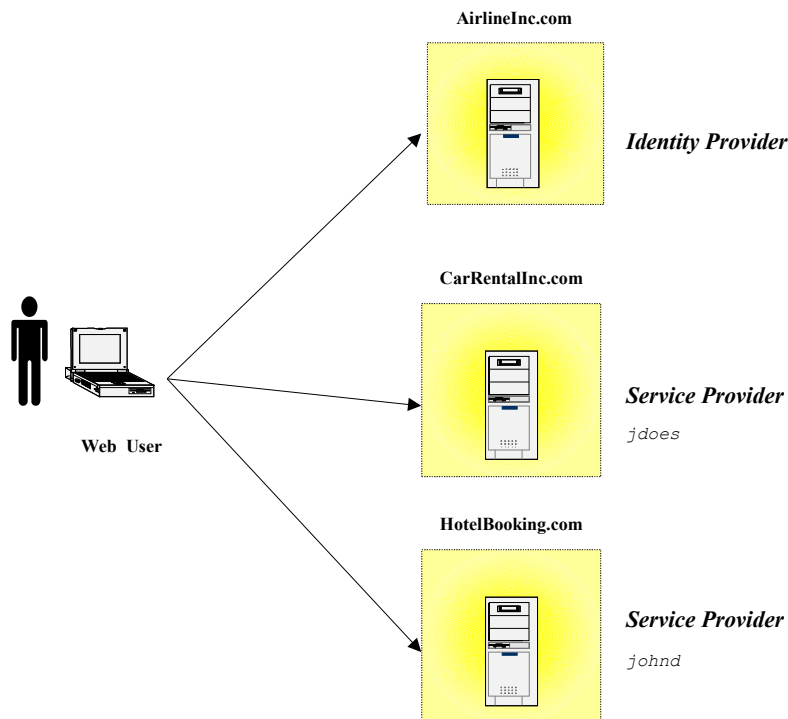


Figure 2: Federation Use Case

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 a number 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, 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) that the SAML protocols can be transported over are defined by **Bindings**. SAML Protocols and Bindings, together with the structure of Assertions, can be combined together to create a **Profile**. In general Profiles can be thought of as 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 identity and attribute information is carried within an Assertion.

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

- **Metadata:** Metadata defines how configuration information shared between two communicating entities is defined and shared. 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 an entity. For instance, a SAML assertion could state that the user is “John Doe”, the user has “Gold” status, the user’s email address is john.doe@example.com, and the user is a member of the “engineering” group. SAML assertions are encoded in a XML schema. SAML defines three kinds of statements that can be carried within an assertion:
 - **Authentication statements:** are issued by the party that successfully authenticated the user. They define who issued the assertion, the authenticated subject, validity period, plus other authentication related information.
 - **Attribute statements:** contain specific details about the user (for example, that they have “Gold” status).
 - **Authorization decision statements:** identifies what the user is entitled to do (for example, whether he is permitted to buy a specified item).
- **Protocols:** SAML defines a number of request/response protocols. The protocol is 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 reference, subject or the statement type.
- **Authentication Request Protocol:** Defines a `<AuthnRequest>` message that causes a `<Response>` to be returned containing one of more assertions pertaining to a Principal. Typically the `<AuthnRequest>` is issued by a Service Provider with the Identity Provider returning the `<Response>` message. Used to support the Web Browser SSO Profile.
- **Artifact Protocol:** Provides a mechanism to obtain a previously created assertion by providing a reference. In SAML terms the reference is called an "artifact". Thus a SAML protocol can refer to an assertion by an artifact, and then when a Service Provider obtains the artifact it can use the artifact Protocol to obtain the actual assertion using this protocol.
- **Name Identifier Management Protocol:** Provides mechanisms to change the value or format of the name of a Principal. The issuer of the request can be either the Service Provider or the Identity Provider. The protocol also provides a mechanism to terminate an association of a name 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 or due to a session timeout.
- **Name Identifier Mapping Protocol:** Provides a mechanism to enable "account linking". Refer to the subsequent sections on Federation.
- **Bindings:** This details exactly how the SAML protocol maps onto the transport protocols. For instance, the SAML specification provides a binding of how SAML request/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. In addition it also defines how the SOAP messages are transported over HTTP.
 - **Reverse SOAP (PAOS) Binding:** Defines a multi-stage SOAP/HTTP message exchange that permits a 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 (i.e. 302 status code responses)
 - **HTTP POST Binding:** Defines how SAML protocol messages can be transported within the base64-encoded content of an HTML form control
 - **HTTP Artifact Binding:** Defines how a reference to a SAML request or response (i.e. an artifact) is transported by HTTP. Defines two mechanisms, either an HTML form control, or a query string in the URL.
 - **SAML URI Binding:** ?? NOT SURE HOW TO EASILY DEFINE THIS
- **Profiles:** The core of the SAML specification defines how the SAML requests and responses are transported, however, a number of use cases have been developed that require the formulation of Profiles that define how the SAML assertions, protocols and bindings are combined. Some of these described in detail later on in the document, in summary they are:
 - **Web Browser SSO Profile:** Defines how a Web Browser supports SSO, when using `<AuthnRequest>` protocol messages in combination with HTTP Redirect, HTTP POST and HTTP Artifact bindings
 - **Enhanced Client and Proxy (ECP) Profile:** Defines how `<AuthnRequest>` protocol messages are used when combined with the Reverse-SOAP binding (PAOS). Designed to support mobile devices front-ended by a WAP gateway
 - **Identity Provider Discovery Profile:** Defines how a service provider can discover which identity providers a principal is using with the Web Server

- **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:

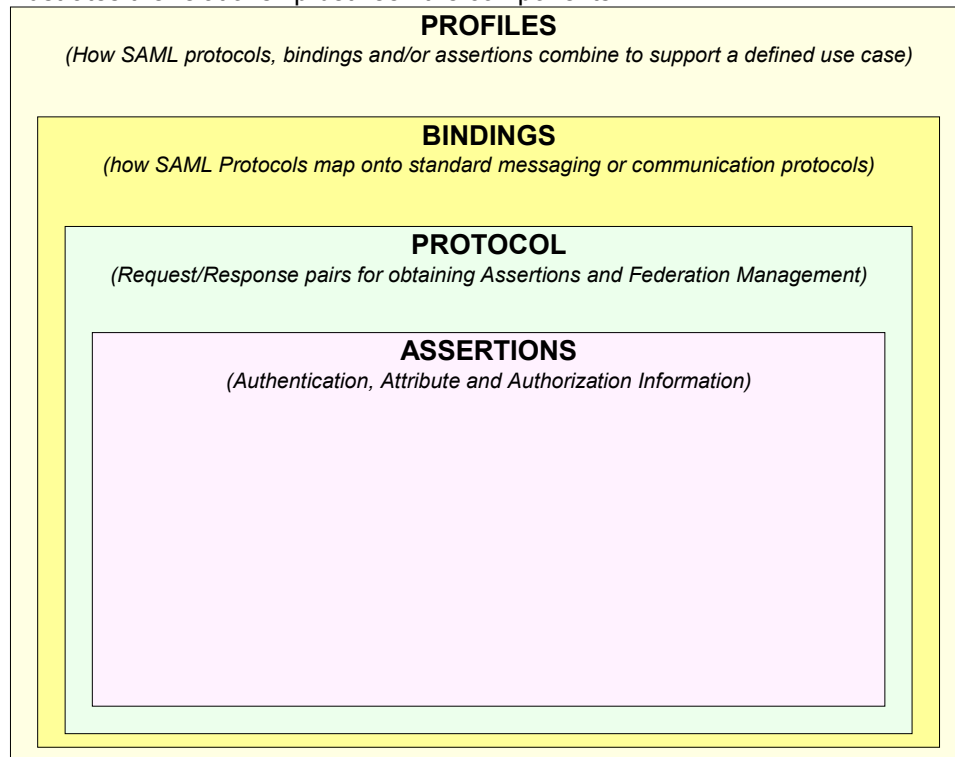


Figure 3: SAML Components

3.3 SAML Structure and Examples

In this section we provide descriptions of some of the SAML structures, bindings and profiles.

3.3.1 Assertions

An assertion consists of one or more statements. For Single Sign-On, typically a SAML assertion will contain a single authentication statement and possibly a single attribute statement. Figure 4 shows a SAML Assertion being carried within a SAML response, which itself is within a SOAP Body. Note that a SAML Response could contain multiple assertions, although its more typical to have a single assertion within a response.

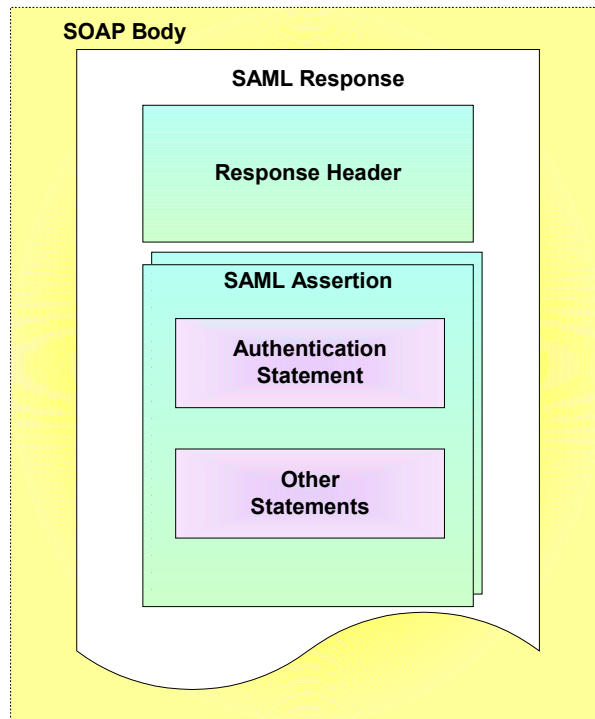


Figure 4: SAML Assertion Structure

Figure 5 shows an example assertion with a single authentication statement. The authentication statement has been highlighted. Note the following:

- The subject (e.g. user) that the authentication pertains to is "j.doe". The format of the subject has been defined. In this case its an email address, a number of predefined formats have been provided in the SAML specification, including custom formats and X.509 subject names.
- Joe was originally authenticated using a protected password mechanism at "2005-01-31T12:00:00Z"

```

<?xml version="1.0" encoding="UTF-8"?>
<saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  Version="2.0"
  IssueInstant="2005-01-31T12:00:00Z">
  <saml:Issuer>
    www.acompany.com
  </saml:Issuer>
  <saml:Subject>
    <saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
      j.doe@company.com
    </saml:NameID>
  </saml:Subject>
  <saml:Conditions NotBefore="2005-01-31T12:00:00Z"
    NotOnOrAfter="2005-01-31T12:00:00Z">
  </saml:Conditions>
  <saml:AuthnStatement
    AuthnInstant="2005-01-31T12:00:00Z" SessionIndex="67775277772">
    <saml:AuthnContext>
      <saml:AuthnContextClassRef>
        urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
      </saml:AuthnContextClassRef>
    </saml:AuthnContext>
  </saml:AuthnStatement>
</saml:Assertion>

```

Figure 5: SAML Assertion

3.3.2 SOAP over HTTP Binding

In environments where the two communicating end points 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 request or response being carried within the SOAP body.

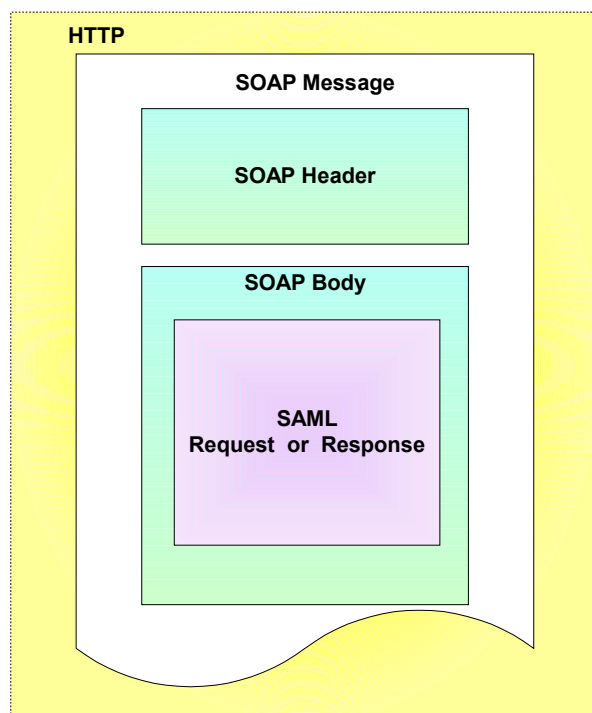


Figure 6: SOAP over HTTP binding

Figure 7 shows an example of a SAML AuthnRequest being transported within a SOAP message. In this example, a SAML assertion is being requested pertaining to the supplied subject (j.does). The SAML AuthnRequest has been highlighted.

```
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap/envelope/">
  <env:Body>
    <samlp:AuthnRequest
      xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
      ForceAuthn="true"
      AssertionConsumerServiceURL="http://www.example.com/"
      AttributeConsumingServiceIndex="0" ProviderName="string"
      ID="abe567de6"
      Version="2.0"
      IssueInstant="2005-01-31T12:00:00Z"
      Destination="http://www.example.com/"
      Consent="http://www.example.com/" >
      <saml:Subject
        xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
        <saml:NameID
          Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
            j.doe@company.com
          </saml:NameID>
        </saml:Subject>
      </samlp:AuthnRequest>
    </env:Body>
  </env:Envelope>
```

Figure 7: SAML AuthnRequest

Figure 8 shows how a SAML response is embedded within a SOAP message. The SAML response provides details as to the version of SAML being used and what request it is responding to. The ResponseID, InResponseTo, version numbers, IssueInstant and the status code represent the SAML response header. Within the response is the SAML assertion and typically one or more statements. The SAML response has been highlighted.

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Body>
    <samlp:Response xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      ID="abe567de6"
      InResponseTo="example-ncname" Version="2.0"
      IssueInstant="2005-01-31T12:00:00Z" Destination="http://www.example.com/"
      Consent="http://www.example.com/">
      <samlp:Status>
        <samlp:StatusCode Value="samlp:Success"/>
        <samlp:StatusMessage>Success</samlp:StatusMessage>
        <samlp:StatusDetail/>
      </samlp:Status>
      ..... SAML ASSERTION AND STATEMENTS
    </samlp:Response>
  </env:Body>
</env:Envelope>
```

Figure 8: SAML Response within SOAP message

3.4 Single Sign-On and Federation Principals

Whilst SAML permits transfer of identity and attribute information from an Identity Provider to a Service Provider, one has to consider what the Service Provider does with that information and how user information relates between organizations. SAML enables Single Sign-On between autonomous organizations, however it also enables different facets of **Identity Management** to be accomplished. SAML 2.0 supports interoperable use of the following approaches to SSO and federation of identity. Where appropriate the relevant SAML mechanisms or schema element(s) are highlighted.

- **Identity-based Federation:** In this environment only the Single Sign-On functionality of SAML is being utilized. For those users that can have authenticated sessions across the organizations they will be required to be registered in both organizations. Therefore if *jdoe* is registered at the Identity provider and wishes to access a resource on a Service Provider in another organization then that same identity will be registered at the Service Provider. The access rights of *jdoe* to resources on the Service Provider will be based on the *jdoe* identity. The identity of the Principal is carried in the <Subject> element of the <Assertion> header.
- **Attribute-based Federation:** Similar to Identity-based Federation, but the type of session and the access right the user has on the Service Provider is based on attribute information transported in the SAML assertion. Whilst the user name can be used for auditing purposes it is not used for access management purposes. An example of this is using a Role attribute, for example "Gold Member". Attributes are carried in the <AttributeStatement> of a SAML assertion.
- **Anonymous Federaton:** It is also possible to provide Anonymity. To support this, authentication and attribute statements containing only a "transient" <NameID> are provided to the Service Provider
- **Pseudonyms:** Users can be identified to a Service Provider during Single Sign On using pair-wise pseudonyms that preserve privacy while enabling a persistent relationship to be maintained with the user.
- **Affiliation:** Permits grouping of Service Providers to form a set. Allows organizations to form relationships able to share in the pseudonymous identification of users. Affiliations are indicated by the SPNameQualifier attribute in the <NameID> and <NameIDPolicy> elements.
- **Opt-in account linking:** Allows a user with multiple accounts at different autonomous Service Providers to link the accounts for future authentication and sign-on. As in the Federation high-level use case the accounts *jdoe* and *johnd* were linked. May use pseudonyms and affiliation mechanism to

388 preserve user privacy.

389 In section 4 a number of Federation use cases are described.

390 **3.5 Use of SAML in other Frameworks**

391 **3.5.1 Web Services Security (WSS)**

392 SAML Assertions can be conveyed by means other than the SAML Request/Response protocols or
393 Profiles defined by the SAML specification set. One example of this is the use of SAML by Web Services
394 Security (WSS). WSS is a set of specifications that define means for providing security protection of
395 SOAP messages. The primary services provided WSS by are Authentication, Data Integrity and
396 Confidentiality.

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

404 In WSS keys and attributes are specified using Tokens. WSS refers to this information as claims. Tokens
405 can either be binary or XML. Binary tokens, such as X.509 Certificates and Kerberos Tickets are carried in
406 a XML wrapper. XML Tokens, such as SAML Assertions are inserted directly as sub elements of the
407 <Security> element. Where WSS requires that the use of a particular token be indicated, a Security Token
408 Reference may be used to refer to the token in one of a number of ways.

409 WSS consists of a Core Specification which describes the mechanisms independent of the type of token
410 being used, a number of Token Profiles which describe the use of particular types of tokens and other
411 Profiles describing other features not covered in the Core Specification. Token profiles cover
412 considerations relating to that particular token type and methods of referencing the token using a Security
413 Token Reference. The use of SAML Assertions with WSS is described in the SAML Token Profile.

414 Because the SAML protocol binding is carried over SOAP, it is easy to get confused between that and the
415 use of SAML Assertions by WSS. They can be distinguished by their purpose, message format and the
416 parties involved.

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

- 418 ● It is used to obtain SAML Assertions for future use; they play no role in protecting the message.
- 419 ● The SAML Assertions are contained within a SAML Response, which is carried in the SOAP body.
- 420 ● The SAML Assertions are provided by a trusted authority or repository and may or may not pertain
- 421 to the party requesting them.

422 The characteristics of the use of SAML Assertions as defined by WSS are as follows.

- 423 ● The SAML Assertions usually play a role in the protection of the message they are carried in,
- 424 typically they contain a key used for digital signatures.
- 425 ● The SAML Assertions are carried in a <Security> element within the SOAP header.
- 426 ● The SAML Assertions will have been obtained previously and typically pertain to the identity of the
- 427 sender.

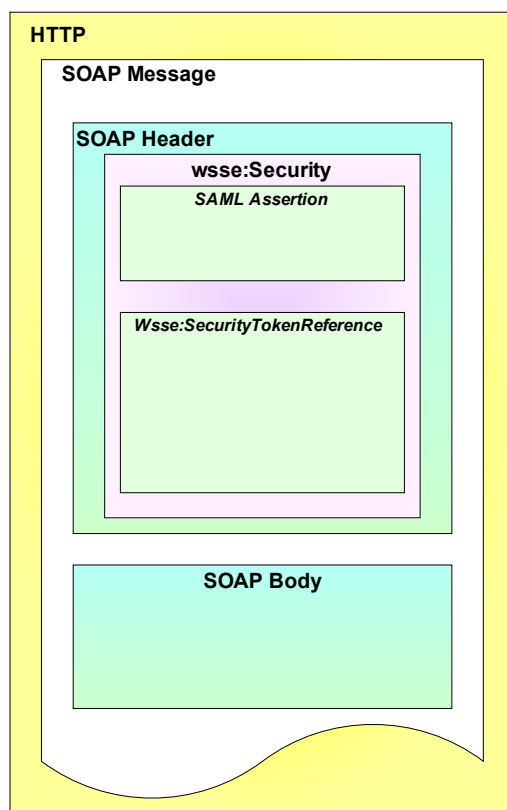


Figure 9: WSS and SAML relationship

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 WSS.

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..

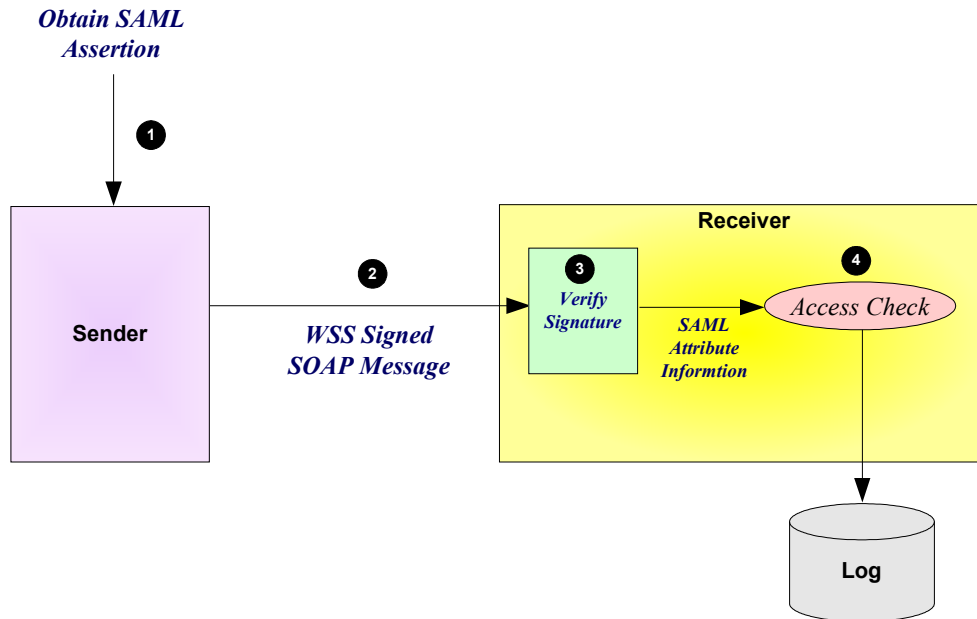


Figure 10: Typical use of WSS and SAML

3.5.2 eXtensible Access Control Markup Language (XACML)

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

The eXtensible Access Control Markup Language (XACML) is an OASIS Standard that defines the syntax and semantics of a language for expressing and evaluating access control policies. The work to define XACML was started slightly after SAML began. From the beginning they were viewed as related efforts and consideration was given to specifying both within the same Technical Committee. In the event 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.

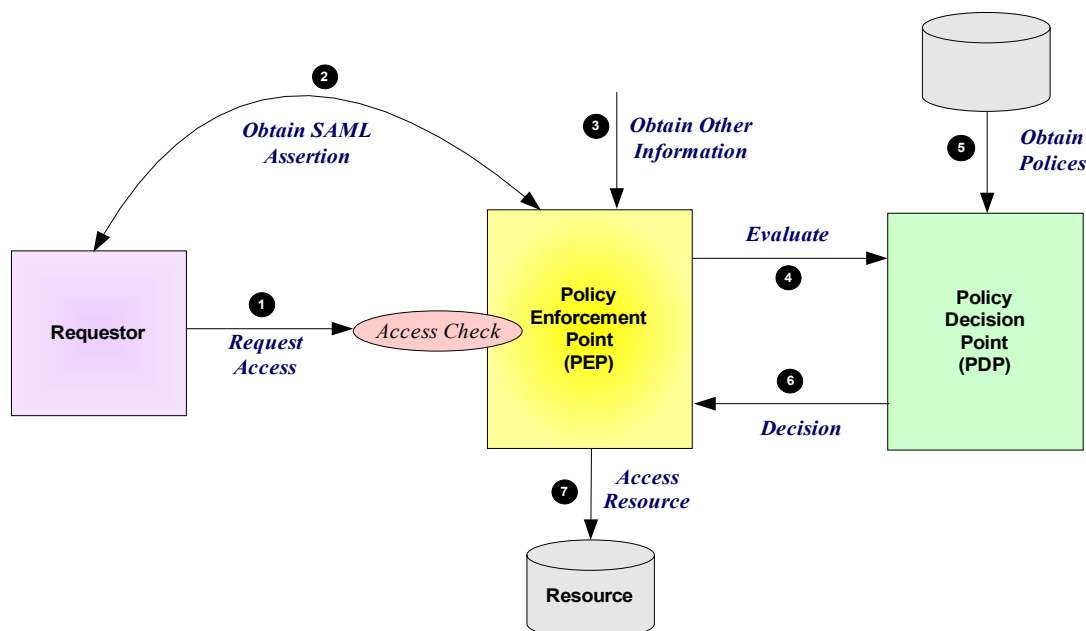


Figure 11: Typical use of XACML and SAML

472

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

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

477 The XACML specification, SAML 2.0 profile of XACML provides additional information on mapping SAML
 478 Attributes to XACML Attributes.

479 The SAML 2.0 profile of XACML also defines a new type of Authorization decision query specifically
 480 designed for use in an XACML environment. It extends the SAML protocol schema and provides a request
 481 and response that contains exactly the inputs and outputs defined by XACML.

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

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

490 3.6 Security in SAML

491 Just providing assertions from an asserting party to a relying party may not be adequate for a secure
 492 system. How does the relying party trust what is being asserted to it? In addition, what prevents a “man-
 493 in-the-middle” attack that grabs assertions to be illicitly “replayed” at a later date? SAML defines a number
 494 of security mechanisms that prevent or detect such attacks. The primary mechanism is for the relying
 495 party and asserting party to have a pre-existing trust relationship, typically involving a Public Key
 496 Infrastructure (PKI). Whilst use of a PKI is not mandated, it is recommended. Use of particular
 497 mechanisms is described for each profile; however, an overview of what is recommended is provided
 498 below:

- 499 • Where **message integrity** and **message confidentiality** are required, then HTTP over SSL 3.0 or
 500 TLS 1.0 is recommended.

- 501 • When a relying party requests an assertion from an asserting party then **bi-lateral authentication** is
502 required and the use of SSL 3.0 or TLS 1.0 using server *and* client authentication are recommended.
- 503 • When an assertion or request “pushed” to a relying party (for example using the HTTP POST binding),
504 then it is mandated that the response message be digitally signed using the XML digital signature
505 standard.
506

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) Profiles
- Federation

4.1 Web Browser SSO Profile

4.1.1 Concept

This Web Browser SSO profile supports four different types of model, two concerning how SAML assertions are provided to the Service Provider (push or pull) and two concerned with how the message flows are initiated (IdP or SP initiated). A combination of the binding techniques and how the message flow is initiated gives rise to 6 different combinations., all of which are described later. The push approach involves using either HTTP redirects or HTTP POST messages to deliver a SAML message. The pull model involves sending a artifact (a type of “reference”) to the receiver which then uses the artifact to dereference and obtain the related SAML message. An example of using artifacts is as follows:

- A user has an authenticated session on the Identity Provider
- The user wants to access a resource on the Service Provider web site and is directed there. In the HTTP message, the *artifact* carried (either as a query variable or as a control in a POST body). The artifact is a base-64 encoded string. It consists of a unique identity of the Identity Provider and a unique reference to the assertion (called the AssertionHandle). The artifact therefore enables the Service Provider to reference an assertion on the Identity Provider
- The Service Provider needs to determine the identity and entitlements of the user and sends a SAML request, containing the artifact, to the Identity Provider asking it what it can assert about the user. The assertions are transferred back in a SAML response.
- The Service Provider then can make whatever authentication and authorization decisions it needs to, based on the received assertions.

This is an example of the HTTP Artifact binding. Figure 12 compares the pull and push approaches.

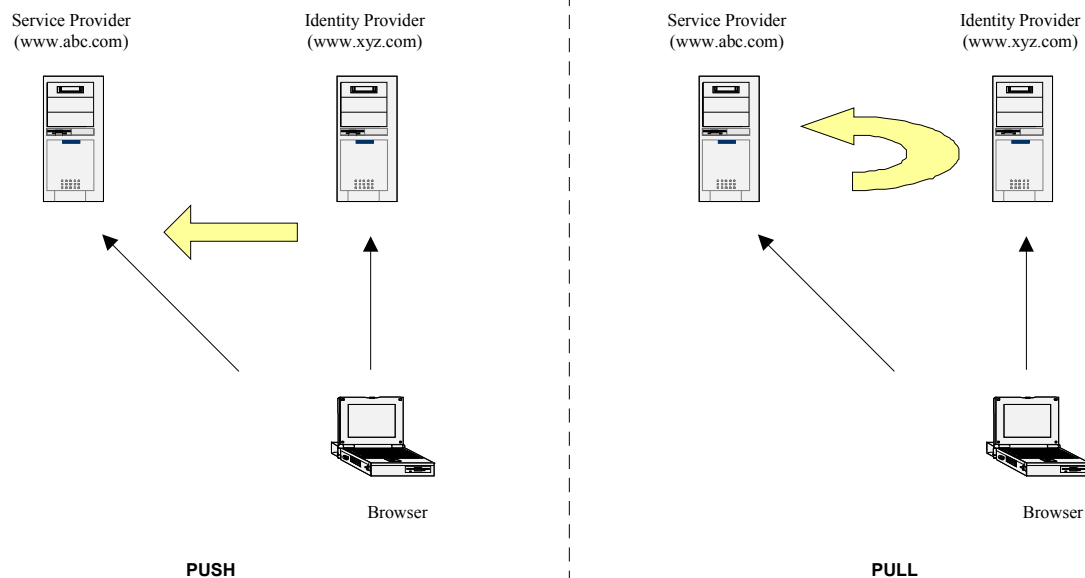


Figure 12: Push and Pull models for Web Browser SSO Profile

The Web Browser SSO Profiles supports two different use cases for situations where the user may or may not be already accessing the Service Provider. The two use cases supported are:

- **IdP Initiated:** The user is accessing resources on the Identity Provider, and wishes to access resources on another web site (the Service Provider). The user already has a current security context with the Identity Provider. A SAML assertion is provided to the Service Provider.
- **SP initiated:** The user is accessing resources on the Service Provider and attempts to access a protected resource requiring knowledge of their authentication and authorization attributes. The Service Provider directs the request to their Identity Provider so that it may provider back SAML assertion(s) in order to validate whether they have access rights to the resource.

Figure 13 compares the two approaches.

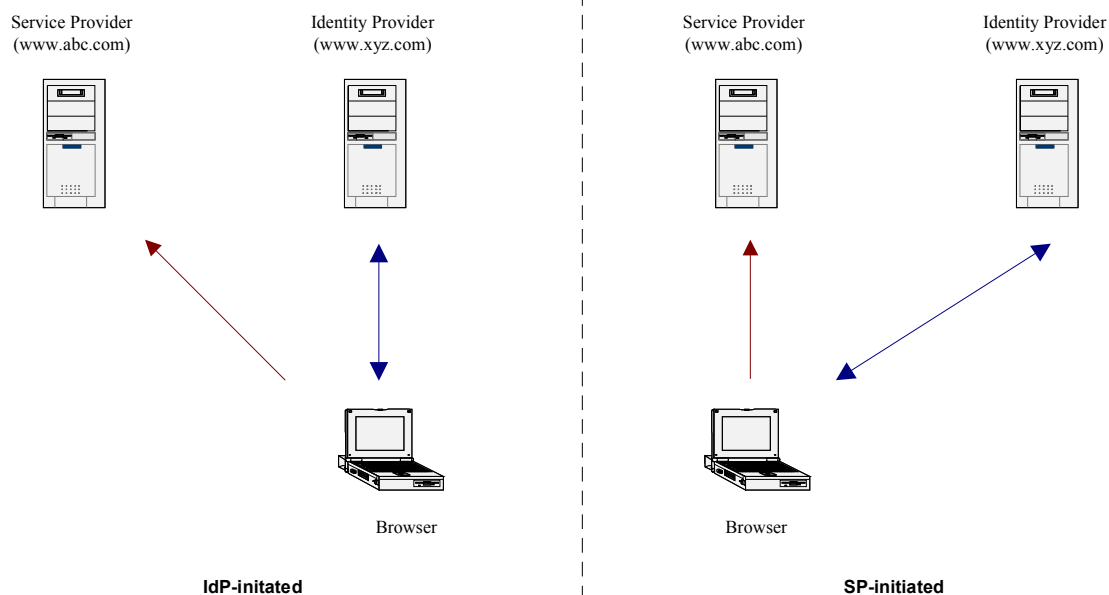


Figure 13: IdP and SP initiated approaches

4.1.2 SP initiated: POST->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. HTTP POST messages are used to deliver the SAML `<AuthnRequest>` to the Identity Provider as well as receive back the SAML response.

Figure 14 illustrates the message flow:

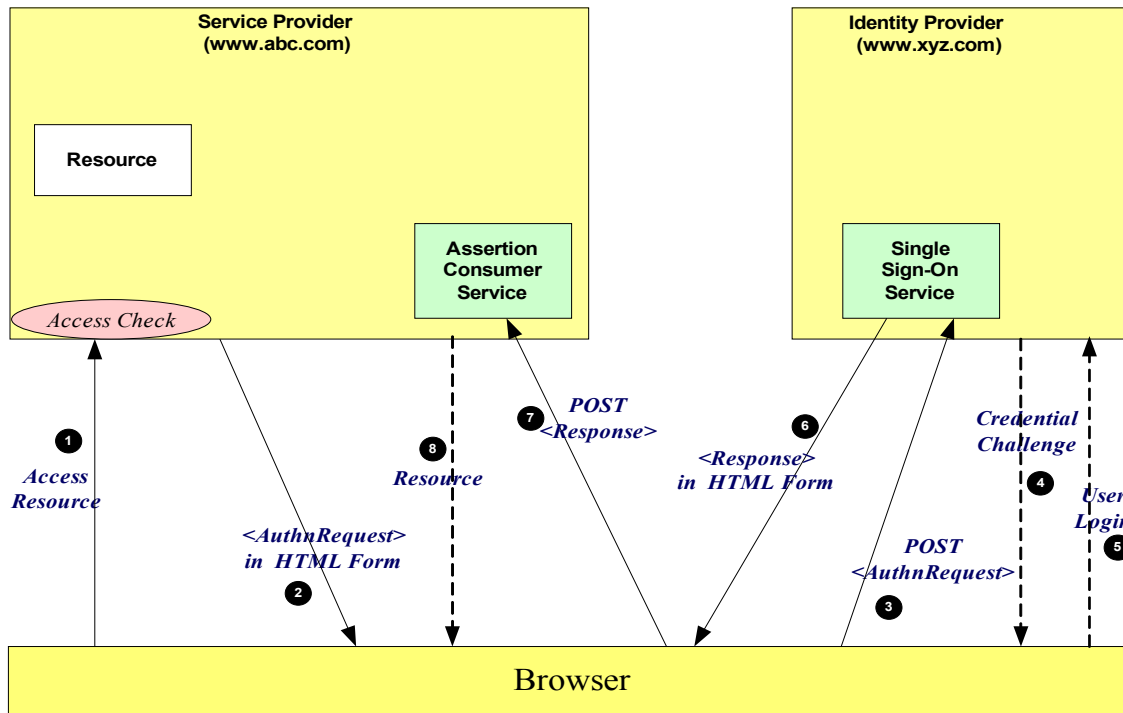


Figure 14: SP initiated: POST->POST binding

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 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 a HTTP POST.
3. The browser, either due to a user action or via an "auto-submit", issues a HTTP POST containing the SAML `<AuthnRequest>` to the Identity Provider's Single Sign-On service.
4. If the user does not have any current 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 sends a HTML form back to the browser. The HTML FORM contains a SAML response, within which is a SAML assertion. The SAML specifications mandate that the response must be digitally signed. Typically the HTML FORM will contain an input or submit action that will result in a HTTP POST.
7. The browser, either due to a user action or via an "auto-submit", issues a HTTP POST containing the SAML response to be sent to the Service Provider's Assertion Consumer service.
8. The Service Provider's Assertion Consumer validates the digital signature on the SAML Response. If this validates correctly, it sends a 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 has the correct authorization to access the www.abc.com web site and the TARGET resource. The TARGET resource is then returned to the browser.

4.1.3 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. A HTTP redirect message is used to deliver the SAML `<AuthnRequest>` to the Identity Provider and a HTTP POST is used to return the SAML response.

Figure 15 illustrates the message flow:

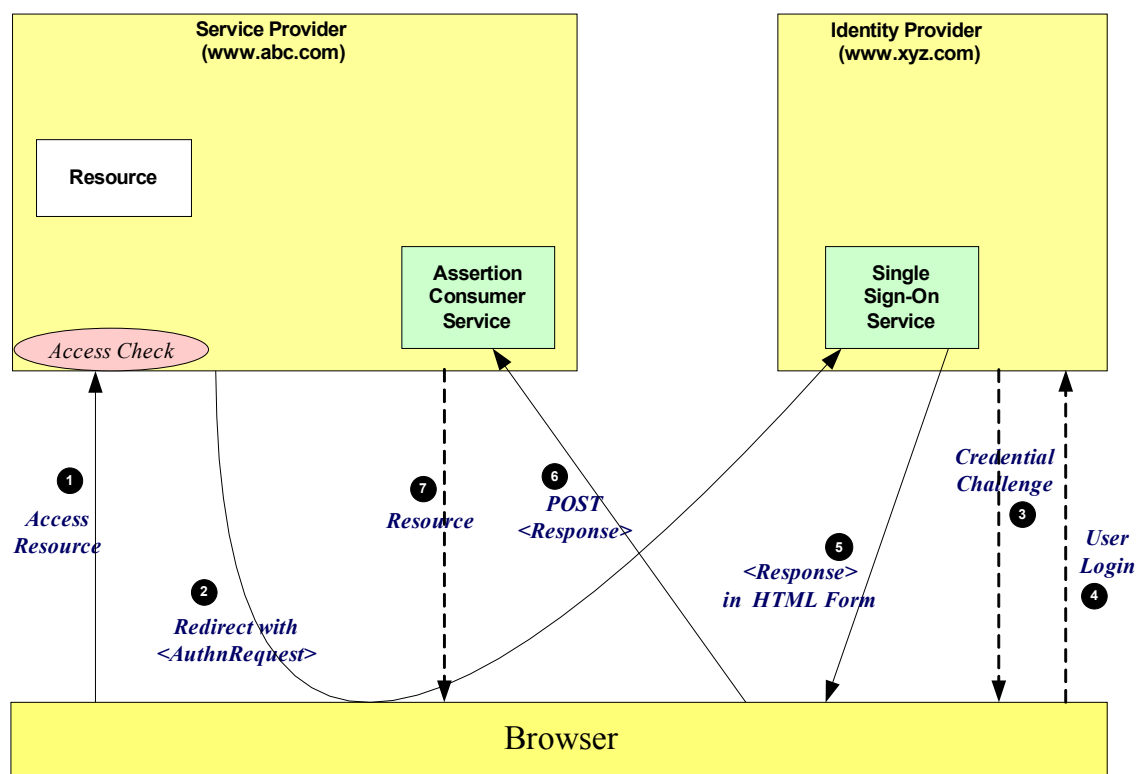


Figure 15: SP initiated: Redirect->POST binding

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 any current 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 specifications mandate that the response must be digitally signed. Typically the HTML FORM will contain an input or submit action that will result in a HTTP POST.

6. The browser, either due to a user action or via an “auto-submit”, issues a HTTP POST containing the SAML response to be sent to the Service Provider's Assertion Consumer service.
7. The Service Provider's Assertion Consumer validates the digital signature on the SAML Response. If this validates correctly, it sends a 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 has the correct authorization to access the www.abc.com web site and the TARGET resource. The TARGET resource is then returned to the browser.

4.1.4 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 a HTTP redirect), which it uses to obtain a SAML <AuthRequest> from the Service Provider's SAML Responder. When the Identity Provider obtains the SAML <AuthRequest> it provides back to the Service Provider the SAML response using the POST binding mechanism.

Figure 16 illustrates the message flow:

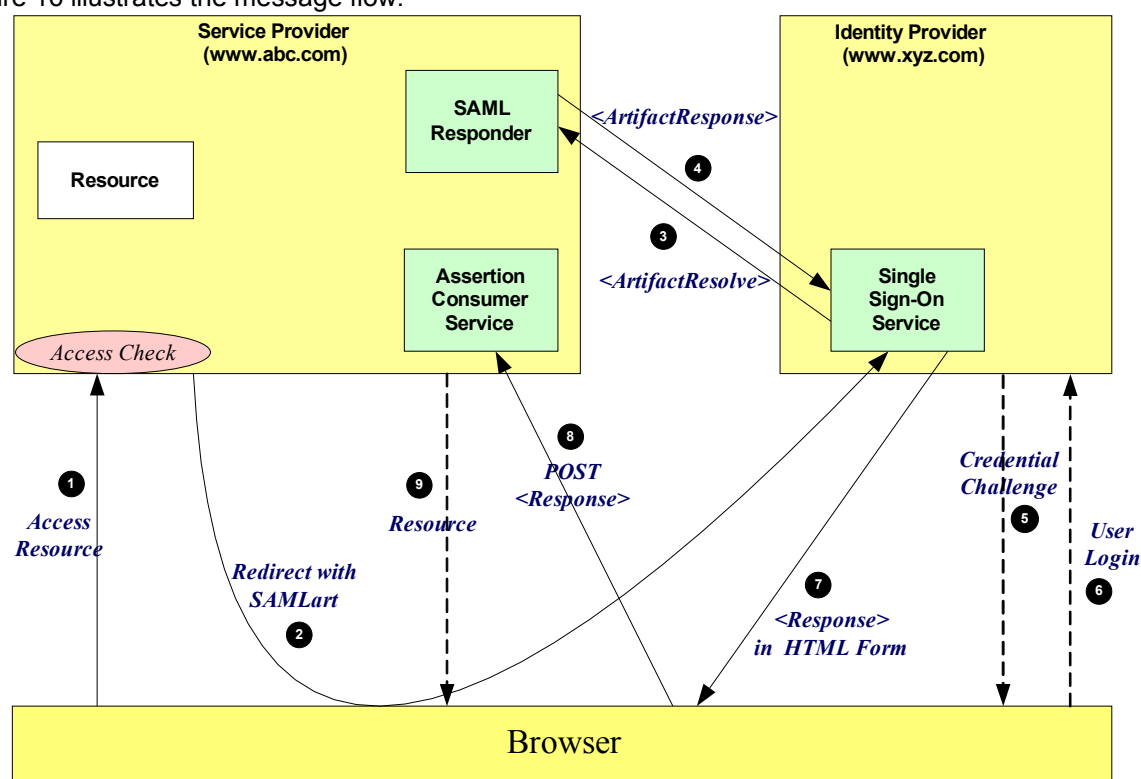


Figure 16: SP initiated: Artifact->POST binding

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 AssertionHandle). 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 Inter-site Transfer 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 a HTTP POST.
3. On receiving the HTTP message, the Single Sign-On Service, extracts the source-ID from the SAML artifact. A mapping between source IDs and remote Responders will already have been established administratively. The Assertion Consumer will therefore know that it has to contact the www.abc.com

625 SAML responder at the prescribed URL. It sends the SAML <ArtifactResolve> message to the
626 Service Provider's SAML responder containing the artifact supplied by its Inter-site Transfer Service.

627 4. The SAML responder supplies back a SAML <ArtifactResponse> message containing the <Authn
628 Request> previously generated.

629 5. The Sign-on Service determines whether the user, for which the <AuthnRequest> pertains, has any
630 current security context on the Identity Provider, or that the policy defines that authentication is
631 required. If the user requires to be authenticated he will be challenged to provide valid credentials.

632 6. The user provides valid credentials and a security context is created for the user.

633 7. The Single Sign-On Service sends a HTML form back to the browser. The HTML FORM contains a
634 SAML response, within which is a SAML assertion. The SAML specifications mandate that the
635 response must be digitally signed. Typically the HTML FORM will contain an input or submit action that
636 will result in a HTTP POST.

637 8. The browser, either due to a user action or via an "auto-submit", issues a HTTP POST containing the
638 SAML response to be sent to the Service Provider's Assertion Consumer service.

639 9. The Service Provider's Assertion Consumer validates the digital signature on the SAML Response. If
640 this validates correctly, it sends a HTTP redirect to the browser causing it to access the TARGET
641 resource, with a cookie that identifies the local session. An access check is then made to establish
642 whether the user has the correct authorization to access the www.abc.com web site and the TARGET
643 resource. The TARGET resource is then returned to the browser.

644 **4.1.5 SP initiated: POST->Artifact binding**

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

652 Figure 17 illustrates the message flow:

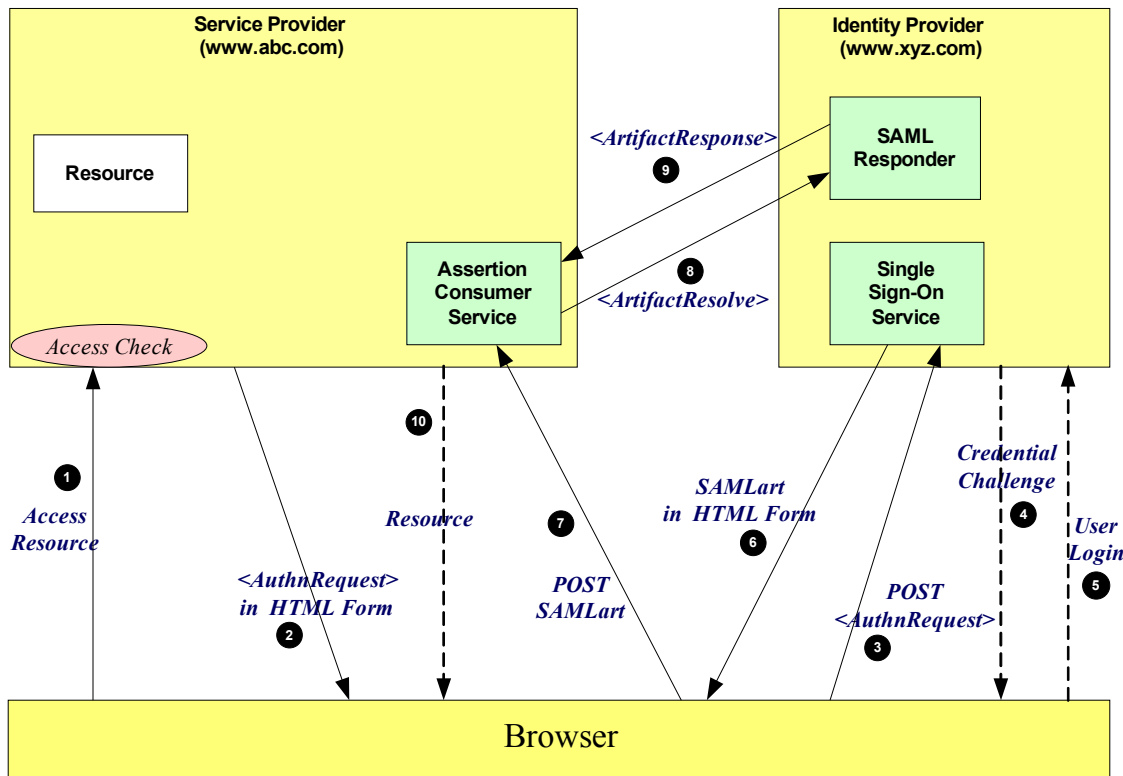


Figure 17: SP initiated: POST->Artifact binding

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 a HTTP POST.
3. The browser, either due to a user action or via an "auto-submit", issues a HTTP POST containing the SAML `<AuthnRequest>` to the Identity Provider's Single Sign-On service.
4. If the user does not have any current 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 AssertionHandle). 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 a HTTP POST.
7. On receiving the HTTP message, the Assertion Consumer Service, extracts the source-ID from the SAML artifact. A mapping between source IDs and remote Responders will already have been established administratively. The Assertion Consumer 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 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 then sends a redirection message containing a cookie back to the browser. The cookie identifies the session. The browser then processes the redirect message and issues a HTTP GET to the TARGET resource on www.abc.com. The GET message contains the cookie supplied back by the Assertion Consumer. 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: 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 <AuthnRequest> to the Identity Provider. The response is in the form of a SAML Artifact. In this example the SAML Artifact is provided back within a 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:

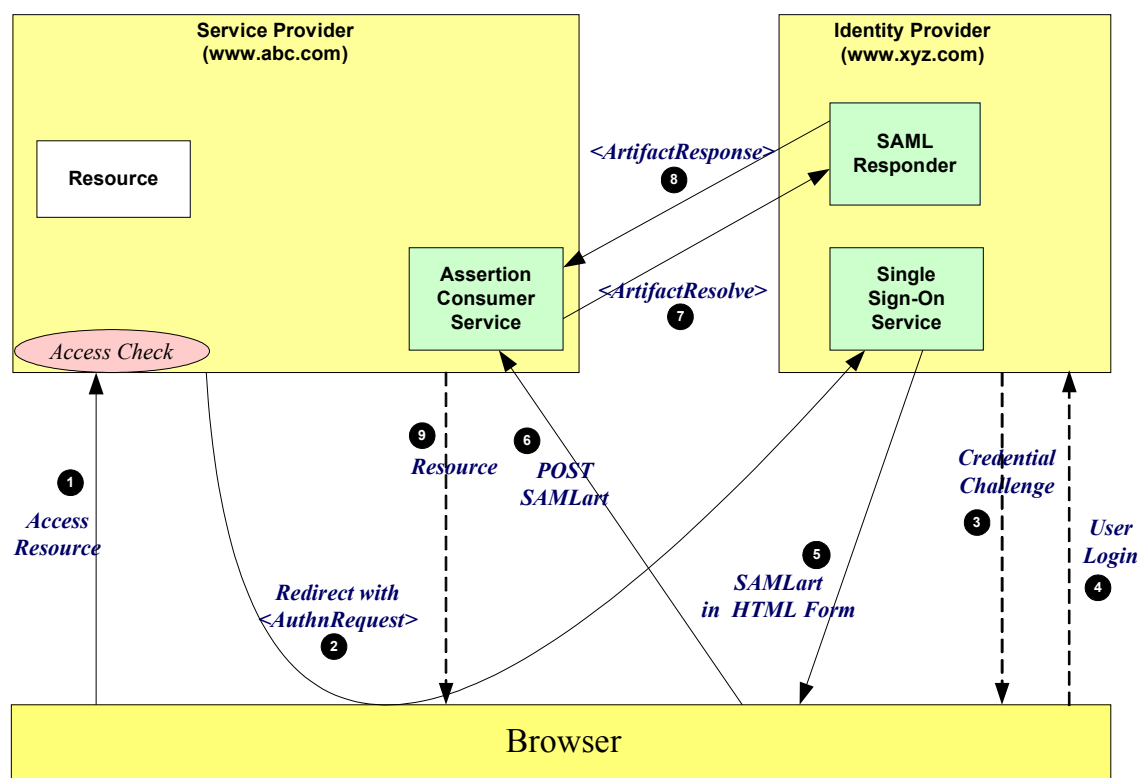


Figure 18: SP initiated: Redirect->Artifact binding

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 any current 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 AssertionHandle). 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 a HTTP POST.
6. On receiving the HTTP message, the Assertion Consumer Service, extracts the source-ID from the SAML artifact. A mapping between source IDs and remote Responders will already have been established administratively. The Assertion Consumer 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 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 then sends a redirection message containing a cookie back to the browser. The cookie identifies the session. The browser then processes the redirect message and issues a HTTP GET to the TARGET resource on www.abc.com. The GET message contains the cookie supplied back by the Assertion Consumer. 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 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 a 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 a 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:

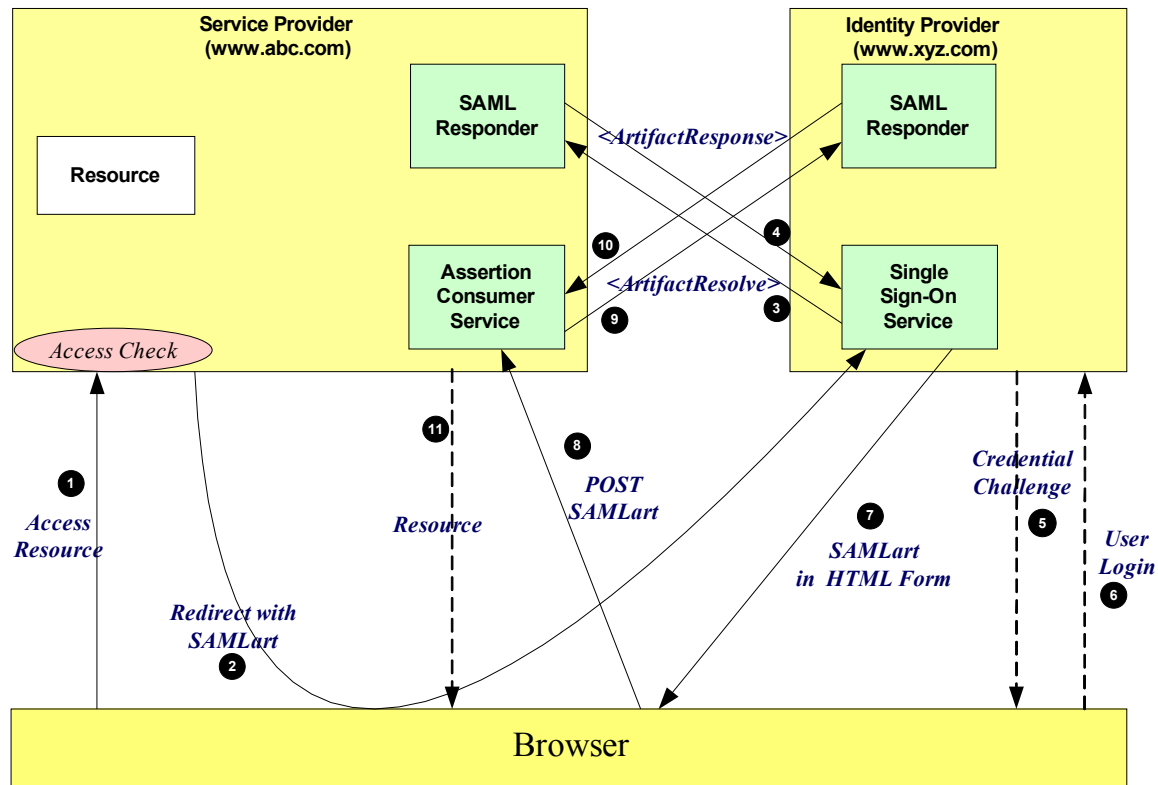


Figure 19: SP initiated: Artifact->Artifact binding

The processing is as follows:

1. The user attempt to access a resource on www.abc.com. The user does not have any current login 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 *AssertionHandle*). 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 Inter-site Transfer 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 a HTTP POST.
3. On receiving the HTTP message, the Single Sign-On Service, extracts the source-ID from the SAML artifact. A mapping between source IDs and remote Responders will already have been established administratively. The Assertion Consumer 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 Inter-site Transfer Service.
4. The SAML responder supplies back a SAML *<ArtifactResponse>* message containing the *<AuthnRequest>* previously generated..
5. The Sign-on Service determines whether the user, for which the *<AuthnRequest>* pertains, has any current 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 *AssertionHandle*). 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 a HTTP POST.

8. On receiving the HTTP message, the Assertion Consumer Service, extracts the source-ID from the SAML artifact. A mapping between source IDs and remote Responders will already have been established administratively. The Assertion Consumer 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 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 then sends a redirection message containing a cookie back to the browser. The cookie identifies the session. The browser then processes the redirect message and issues a HTTP GET to the TARGET resource on www.abc.com. The GET message contains the cookie supplied back by the Assertion Consumer. 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.

4.1.8 IdP initiated: POST binding

In this use case the user has a security context on the Identity Provider and wishes to access a resource on a remote server (www.abc.com). The SAML assertion is transported to the Service Provider using the POST binding.

Figure 20 shows the process flow:

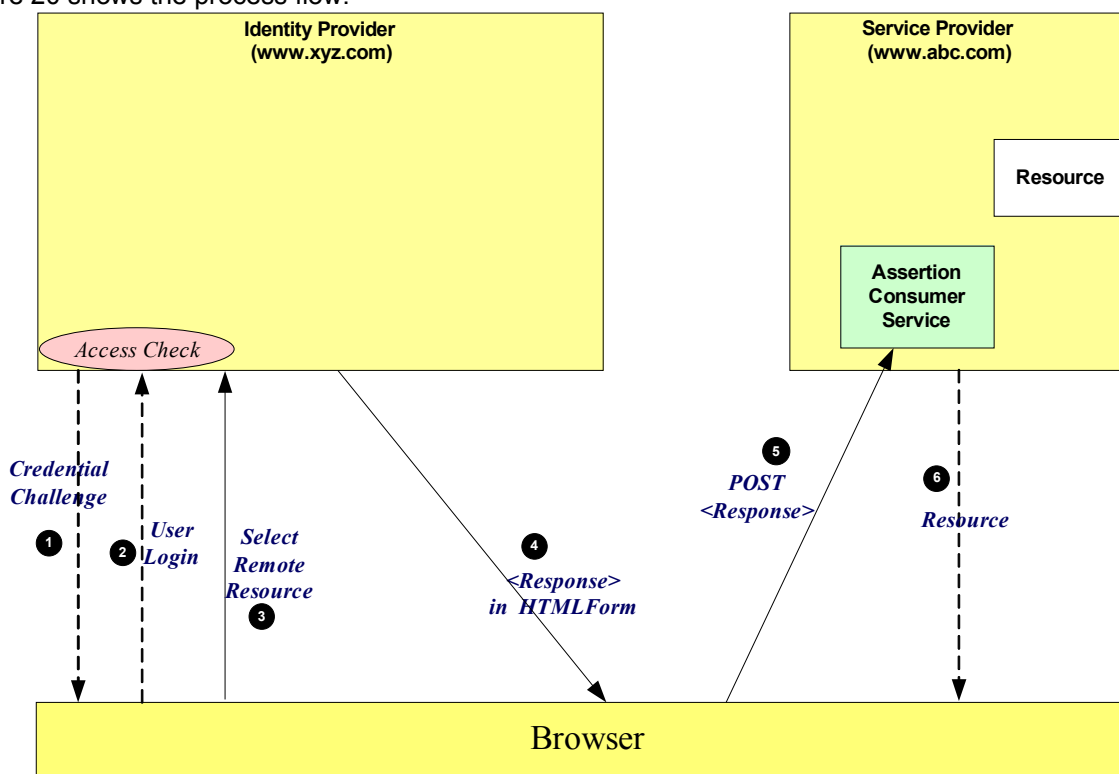


Figure 20: IdP initiated: POST binding

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 specifications mandate that the response must be digitally signed. Typically the HTML FORM will contain an input or submit action that will result in a HTTP POST.
5. The browser, either due to a user action or via an “auto-submit”, issues a HTTP POST containing the SAML response to be sent to the Service provider' Assertion Consumer service.
6. The Service Provider's Assertion Consumer validates the digital signature on the SAML Response. If this validates correctly, it sends a HTTP redirect to the browser causing it to access the TARGET resource, withing with a cookie that identifies the local session. 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. The TARGET resource is then returned to the browser.

4.1.9 IdP initiated: Artifact binding

In this use case the user has a security context on the Identity Provider and wishes to access a resource on a remote server (www.abc.com). An artifact is provided to the Service Provider, which its can use (that is “de-reference”) to obtain the associated SAML response from the Identity Provider.

Figure 21 shows the process flow:

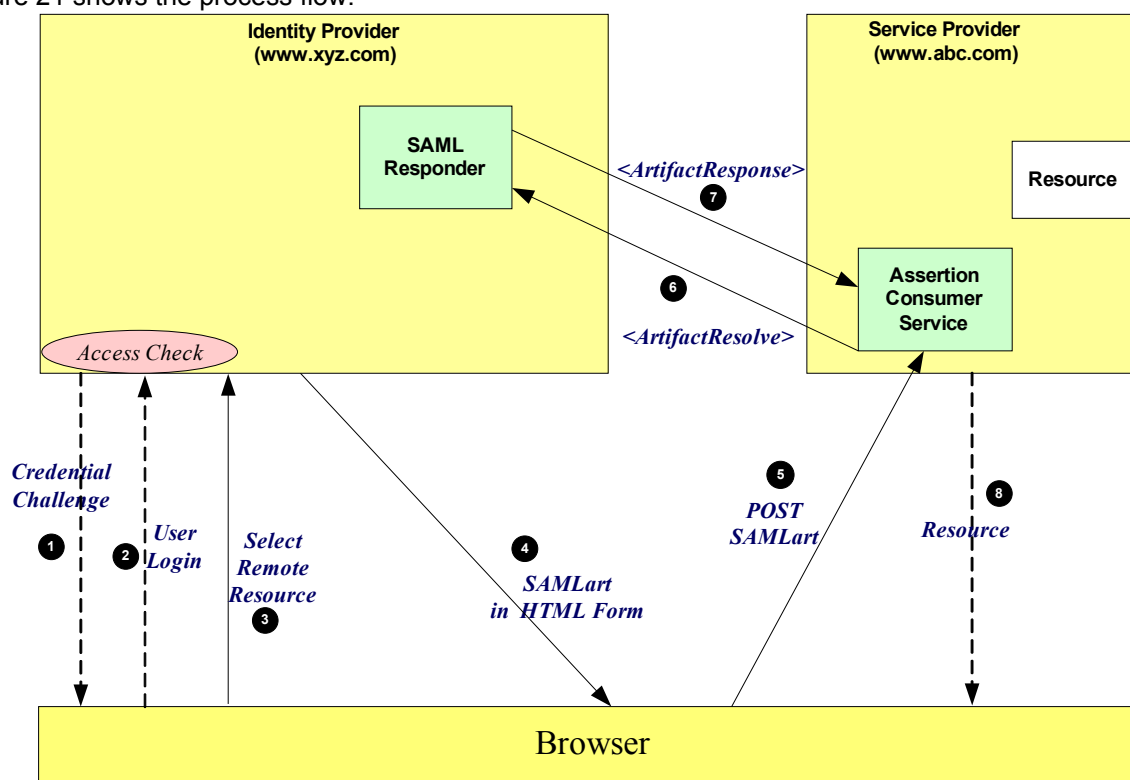


Figure 21: IdP initiated: Artifact binding

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 AssertionHandle). 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 Inter-site Transfer 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 a HTTP POST.

5. On receiving the HTTP message, the Assertion Consumer Service, extracts the source-ID from the SAML artifact. A mapping between source IDs and remote Responders will already have been established administratively. The Assertion Consumer 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 will send a SAML `<ArtifactResolve>` message to the Identity Provider's SAML responder containing the artifact supplied by its Inter-site Transfer Service.
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 then sends a redirection message containing a cookie back to the browser. The cookie identifies the session. The browser then processes the redirect message and issues a HTTP GET to the TARGET resource on www.abc.com. The GET message contains the cookie supplied back by the Assertion Consumer. 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 can not be used)

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 22 shows the message flows between the ECP, Service Provider and Identity Provider. The ECP is shown as a single logical entity.

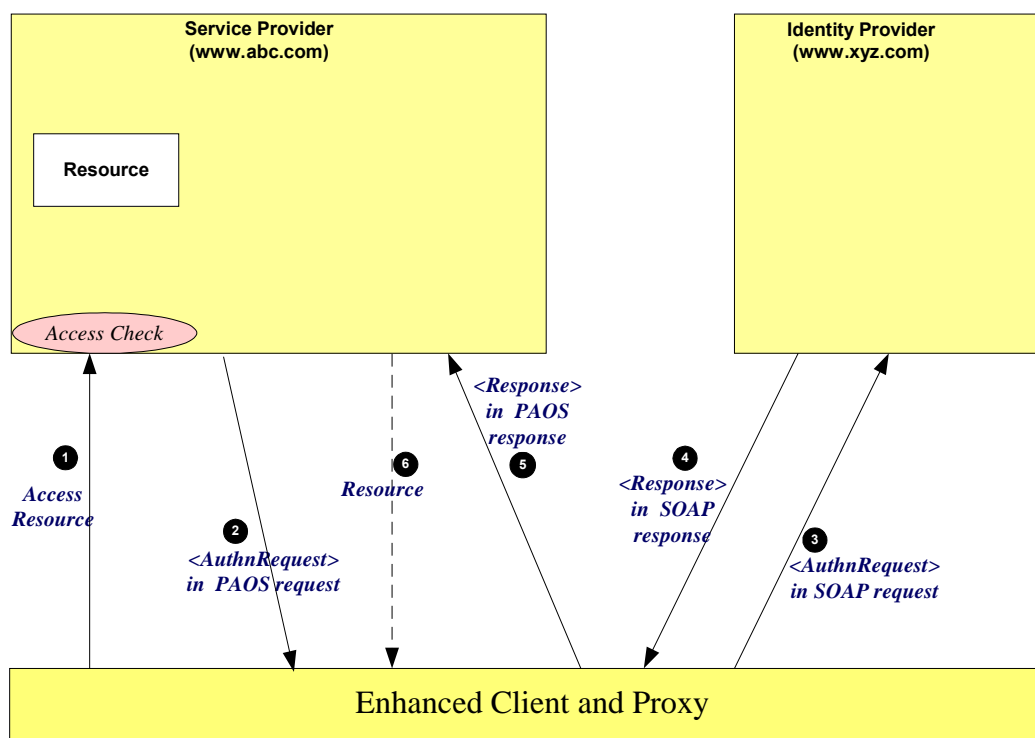


Figure 22: ECP with PAOS

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 a 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 a HTTP response containing the resource originally requested.

4.3 Federation

TBD

4.3.1 Introduction

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

- **Federation during `<AuthnRequest>`:** an Identity Provider federates the Identity Provider's Principal with the Principal's identity at the Service Provider.
- **Federation Termination:** termination of a Federation
- **Accounting Linking:** mapping between two existing accounts on service Providers via an Identity Provider.

874 **4.3.2** *Federation during <AuthnRequest>*

875 TBD

876 **4.3.3** *Federation Termination*

877 TBD

878 **4.3.4** *Accounting Linking*

879 TBD

5 Documentation roadmap

- **Security Assertion Markup Language (SAML) 2.0 Executive Overview.** (sstc-saml-exec-overview-2.0) Provides a brief overview of SAML and describes its primary benefits.
- **Security Assertion Markup Language (SAML) 2.0 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 2.0 and SAML 1.1

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 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 is 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 protocol (“core”) specification is now referred to as Assertions and Protocols, 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.

6.2 Versioning Differences

- The SAML assertions namespace (known by its convention prefix `saml:`) and protocols namespace (known by its conventional prefix `samlp:`) namespaces now contain the string “2.0” in recognition of this new major version of SAML.
- The `MajorVersion` and `MinorVersion` attributes that appear on various elements now need to contain the string values “2” and “0”, respectively in recognition of this new major version of SAML.
- 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 inner statements. (The `<Subject>` element is optional for

extensibility reasons, but is required for all SAML-specified statement types.)

- The `<ConfirmationMethod>` element is now non-repeatable (it is still required for one to appear inside its parent).
- The `<ds:KeyInfo>` element is now allowed only inside `<SubjectConfirmationData>`.

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.

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.
- The `Issuer` field is now an element and is based on the same datatype that underlies name identifiers, for more unified treatment.
- 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.

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

- A `SessionIndex` attribute has been added to the `<Statement>` and `<SubjectQuery>` elements. Thus, this index is available on all statements, not just `<AuthenticationStatement>`.
- 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

- 988 can be specified.
- 989 • In such an assertion, it is now possible to specify many more details about the authentication that
 - 990 was performed using the new authentication context schemas; the old `AuthenticationMethod`
 - 991 field has been removed.
 - 992 • There is a new federated name management (registration and deregistration) protocol.
 - 993 • There is a new name identifier mapping protocol.

994 **6.10 Differences in Bindings and Profiles**

- 995 • A lot of profile detail has been refactored out to become new, more generic bindings; the profiles are
- 996 much thinner. For example, there's now an HTTP redirect/POST binding.
- 997 • There is a new HTTP-based binding added for retrieval of assertions by means of URIs.
- 998 • A PAOS (reverse SOAP) binding has been added.
- 999 • An enhanced client profile has been added.
- 1000 • The two original browser profiles (browser/artifact and browser/POST) have become a single web
- 1001 SSO profile.
- 1002 • A set of mechanisms for relaying state have been added to most of the bindings.

1003 **6.11 Other Differences**

- 1004 • XSD element substitution has been blocked.
- 1005 • The `<ds:Signature>` that allows for the digital signing of assertions and messages has been
- 1006 positioned earlier in the respective content models.
- 1007 • The usage of `<ds:KeyInfo>` has been clarified to more clearly allow for impersonation.
- 1008 • The authorization decision feature (statement and query) has been frozen; if more functionality is
- 1009 desired, it is suggested that XACML [XACML] be used.
- 1010 • A `<ProxyRestriction>` element **and other conditions** has been added.

1011 **TBS: validity period semantics and syntax extended, element and attribute name changes, terminology,**

1012 **wildcarding changes, 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-cd-04. 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. 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. <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. 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. 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. 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. See <http://www.oasis-open.org/committees/security/>.
- [SAMLWeb]** OASIS Security Services Technical Committee website, <http://www.oasis-open.org/committees/security/>.
- [WSS]** A. Nadalin, ed., *OASIS Web Services Security: SOAP Message Security 1.0*. Available on the OASIS WSS 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

1050

1051 The editors would like to acknowledge the contributions of the OASIS Security Services Technical
1052 Committee, whose voting members at the time of publication were:

1053 • TBD

1054

B. Revision History

Rev	Date	By Whom	What
00	6 Nov 2003	John Hughes	Storyboard version
01	22 Jul 2004	John Hughes	First draft
02	27 Sept 2004	John Hughes	Second Draft .General updates, limited distribution
03	20 Feb 2005	John Hughes	DCE/Kerberos use section removed. Use of SAML in other frameworks added. SAML 2.0 XML examples included. Updated Web SSO examples to remove use of ITS

1055

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.

Copyright © OASIS Open 2004. All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself does not be modified in any way, such as by removing the copyright notice or references to OASIS, except as needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.