Security Assertion Markup Language (SAML) 2.0 Technical Overview

Working Draft 01, 22 July 2004

Document identifier:
sstc-saml-tech-overview-2.0-draft-01

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

Editors:
John Hughes, Entegrity Solutions
Eve Maler, Sun Microsystems

Contributors:
TBD

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

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 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, its 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

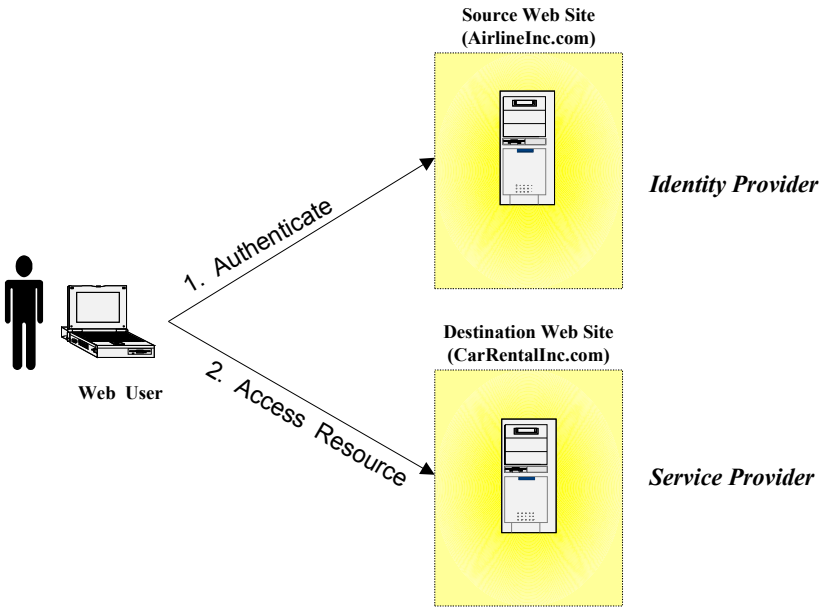
This is the original use case as supported in SAML 1.0 and 1.1. It illustrates the support for Cross Domain Single Sign-On. A user has a logon session (that is a *security context)* on a website (AirlineInc.com) and is accessing resources on that site. At some either explicitly or transparently he is 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 jdoe and on HotelBookings.com as johnd. Account Linking enables a pseudonym to be established that links the two accounts.

MORE DETAILS TO ADD – JohnH.

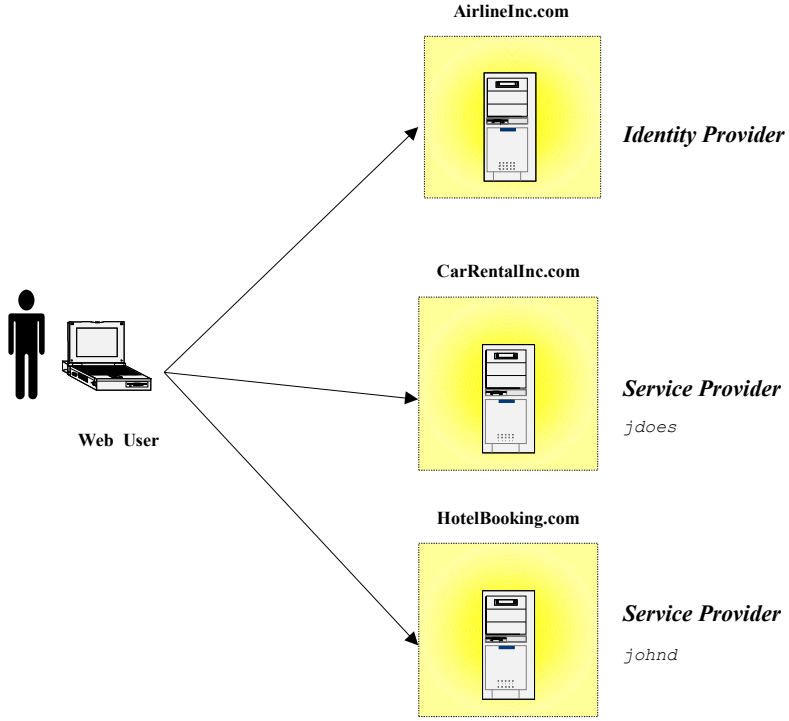


Figure 2: Federation Use Case

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 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 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 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 ([SAML-Meta] [SAML-AuthnCxt] 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:

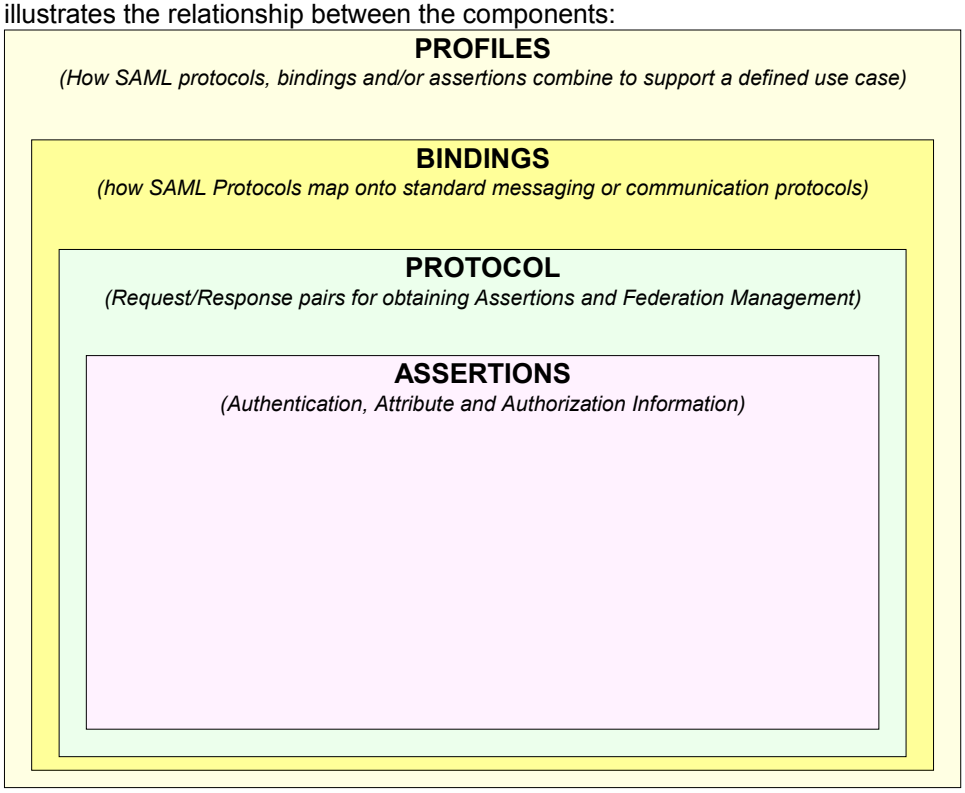


Figure 3: SAML Components

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 withing a SOAP Body. Note that a SAML Response could contain multiple assertions, although its more typical to have a single assertion within a response.

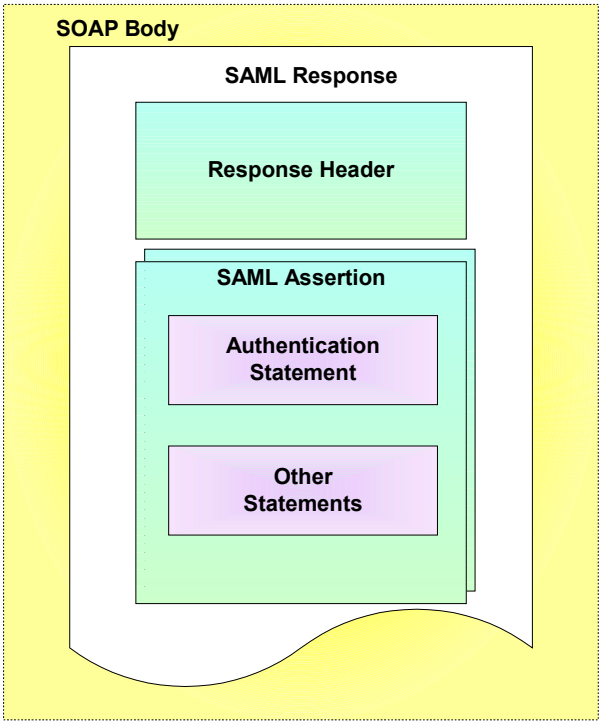


Figure 4: SAML Assertion Structure

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 “joe”. The format of the subject has been defined. In this case its a custom format; however, a number of predefined formats have been provided in the SAML specification, including email addresses and X.509 subject names.
- Joe was originally authenticated using a password mechanism at “2002-06-19T17:05:17.706Z".
- TO DO – SAML 2.0 VERSION

```
<saml:Assertion
  xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
  MajorVersion="1"
  MinorVersion="1"
  AssertionID="buGxcG4gILg5NlocyLccDz6iXrUa"
  Issuer="www.acompany.com"
  IssueInstant="2002-06-19T17:05:37.795Z">
       <saml:Conditions NotBefore="2002-06-19T17:00:37.795Z"
         NotOnOrAfter="2002-06-19T17:10:37.795Z"/>
       <saml:AuthenticationStatement
         AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password"
         AuthenticationInstant="2002-06-19T17:05:17.706Z">
              <saml:Subject>
                     <saml:NameIdentifier
                       NameQualifier=http://www.acompany.com
                       Format="http://www.customformat.com/">
                            uid=joe
                     </saml:NameIdentifier>
                     <saml:SubjectConfirmation>
                            <saml:ConfirmationMethod>
                            urn:oasis:names:tc:SAML:1.0:cm:artifact-01
                            </saml:ConfirmationMethod>
                     </saml:SubjectConfirmation>
              </saml:Subject>
       </saml:AuthenticationStatement>
</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.

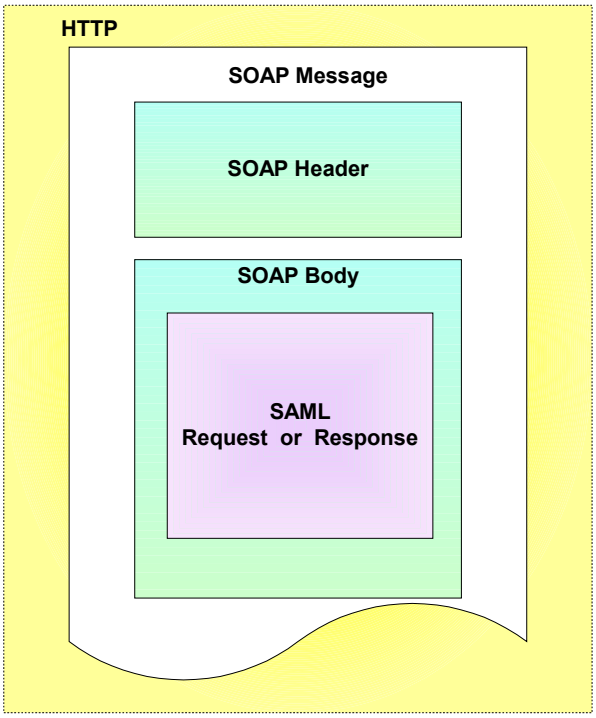


Figure 6: SOAP over HTTP binding

Figure 7 shows an example of a SAML request being transported within a SOAP message. In this example, a SAML assertion is being requested pertaining to a supplied artifact. The use of the artifact is explained later in the Use Case and Profiles section. The SAML request has been highlighted. (TO DO: PROVIDE SAML 2.0 EXAMPLE)

```
<env:Envelope
       xmlns:env="http://www.w3.org/2003/05/soap/envelope/">
       <env:Body>
              <samlp:Request
xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
                     xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
                      MajorVersion="1"
                     MinorVersion="0"
                     RequestID="192.168.16.51.1024506224022"
                     IssueInstant="2002-06-19T17:03:44.022Z">
                     <samlp:AssertionArtifact>
                            AAGZE1RNQJEFzYNCGAGPjWvtDIRSZ4
                            lWDqBphqAEYkgG/RBdHoeMsu1f
                     </samlp:AssertionArtifact>
              </samlp:Request>
       </env:Body>
</env:Envelope>
```

Figure 7: SAML Artifact Request

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. NEED SAML 2.0 VERSION

```
<env:Envelope
xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
       <env:Body>
              <samlp:Response
                     xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
                     ResponseID="P1YaA+Q/wSM/t/8E3R8rNhcpPTM="
                     InResponseTo="192.168.16.51.1024506224022"
                     MajorVersion="1"
                     MinorVersion="0"
                     IssueInstant="2002-06-19T17:05:37.795Z">
                     <samlp:Status>
                            <samlp:StatusCode Value="samlp:Success" />
                     </samlp:Status>

                     …… SAML ASSERTION AND STATEMENTS

              </samlp:Response>
       </env:Body>
</env:Envelope>
```

Figure 8: SAML Response with 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 enables the following to be provided. Where appropriate the relevant SAML mechanisms or schema element(s) are highlighted.

- ***Identity Single Sign-On:*** 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 `<NameID>` element which is in the `<Subject>` element of the `<Assertion>` header.
- ***Attribute Single Sign-On***: Similar to Pure Single Sign-On, 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 Single Sign-On***: It is also possible to provide *Anonymity*. To support this only attribute statements are provided to the Service Provider, the Principal `<NameID>` element not being sent.
- ***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* where linked.
- ***Affiliation:*** Permits grouping of Service Providers to form a set. Allows organizations to form alliances. Affiliations are indicated by the `SPNameQualifier` attribute in the `<NameID>` and `<NameIDPolicy>` elements.
- ***Pseudonyms:*** When accounts are linking a persistent pseudonym can be used to identity the account linkage.

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

3.5 Use of SAML in other Frameworks

DESCRIBE REALTIONSHIP WITH SHIBBOLETH, LIBERTY AND WSS and XACML

3.6 Security in SAML

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). Whilst 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 are 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) Profiles
- Using Kerberos
- 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 9 compares the pull and push approaches.

Figure 9: 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 10 compares the two approaches.

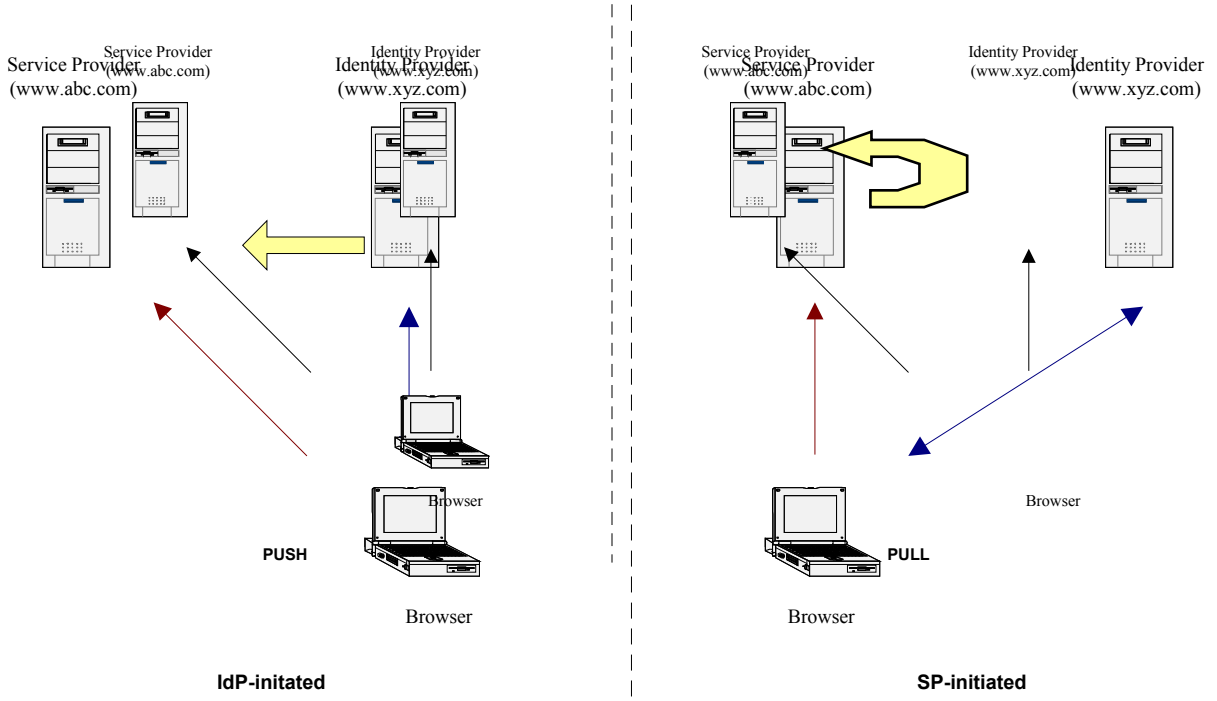


Figure 10: IdP and SP initiated approaches

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 11 illustrates the message flow:

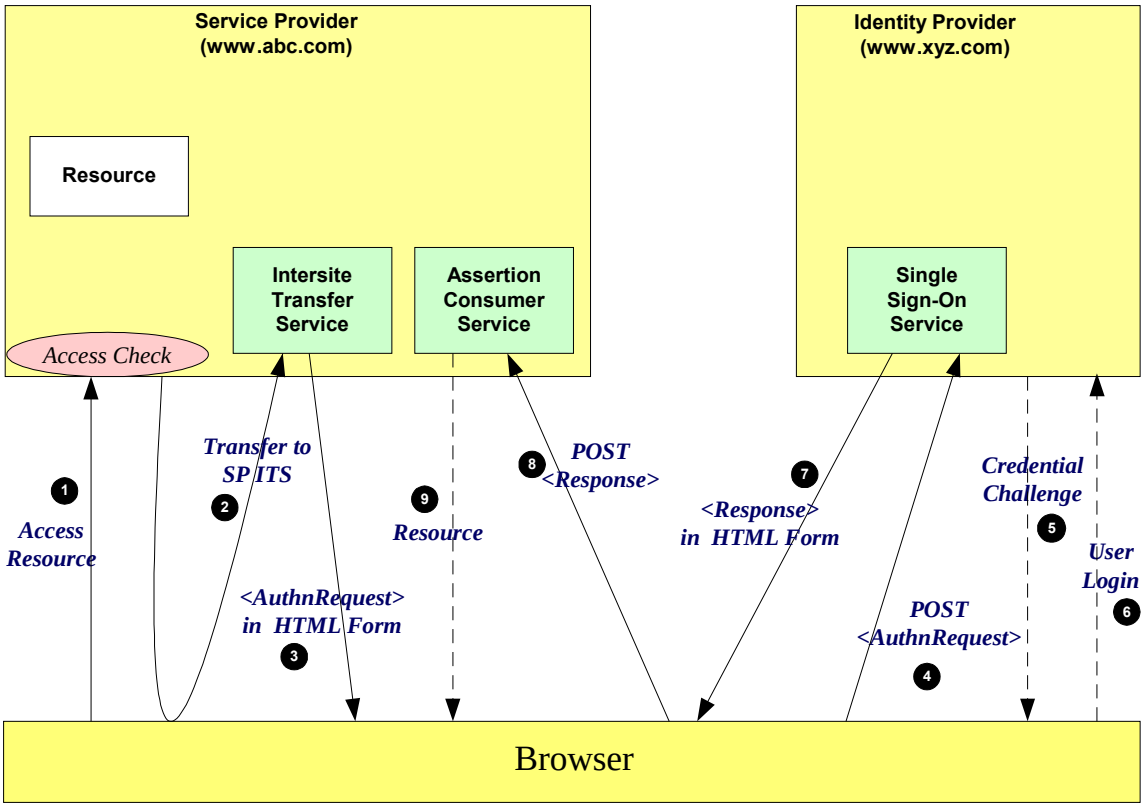


Figure 11: SP initiated: POST->POST binding

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 application then directs the request to the local Inter-site Transfer Service. The request contains the URL of the resource on the destination site (the TARGET URL). The URL would look something like the following (without the URL encoding):

   https://www.abc.com:8002/InterSiteTransfer?TARGET=http://www.xyz.com/index.asp
3. The Inter-site Transfer Service 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.
4. 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.
5. 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.
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 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.
8. 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.
9. 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 12 illustrates the message flow:

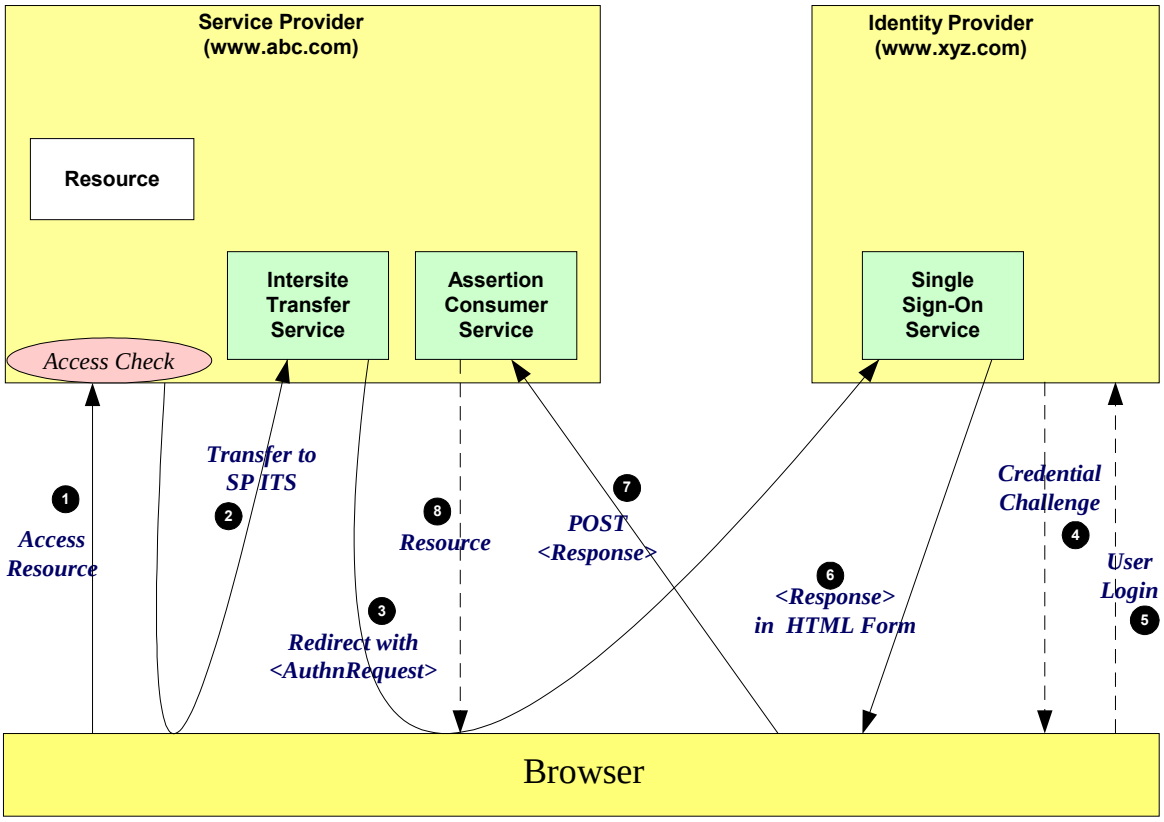


Figure 12: SP initiated: Redirect->POST binding

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 application then directs the request to the local Inter-site Transfer Service. The request contains the URL of the resource on the destination site (the TARGET URL). The URL would look something like the following (without the URL encoding):

   https://www.abc.com:8002/InterSiteTransfer?TARGET=http://www.xyz.com/index.asp

3. The Inter-site Transfer Service 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.

4. 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.
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.

4.1.4 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 13 illustrates the message flow:

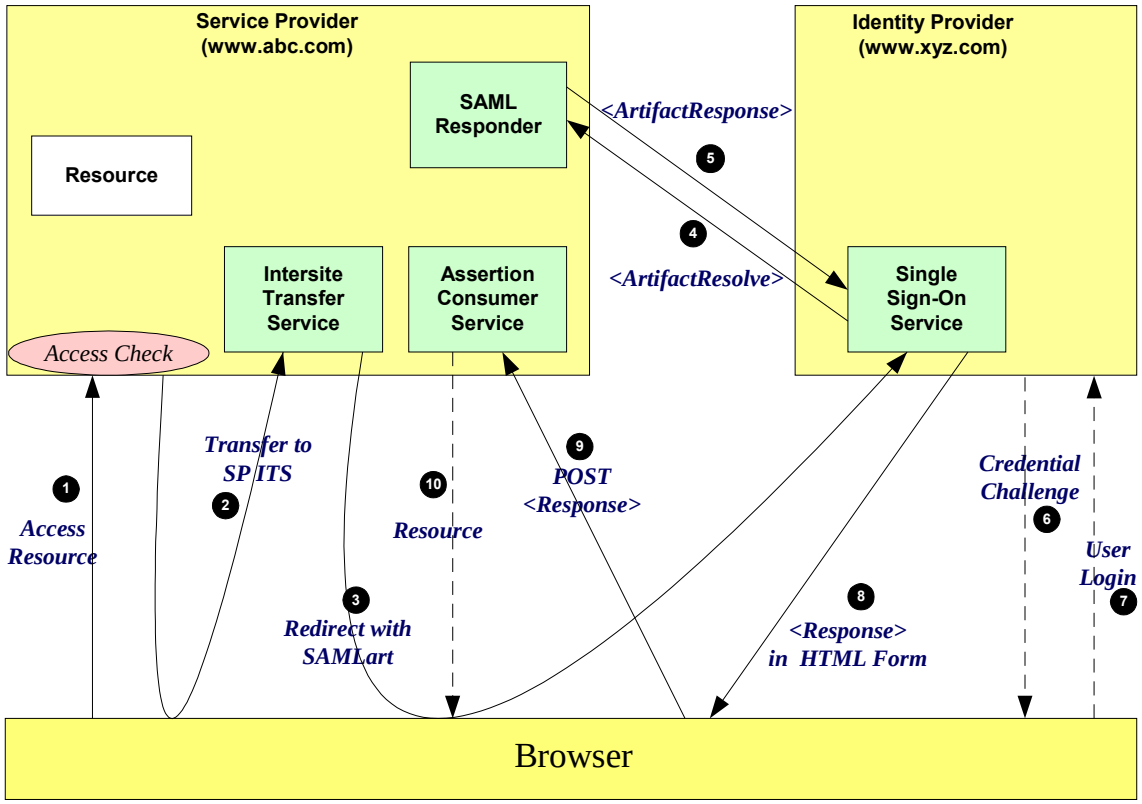


Figure 13: SP initiated: Artifact->POST binding

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 application then directs the request to the local Inter-site Transfer Service. The request contains the URL of the resource on the destination site (the TARGET URL). The URL would look something like the following (without the URL encoding):

   https://www.abc.com:8002/InterSiteTransfer?TARGET=http://www.xyz.com/index.asp

3. The Inter-site Transfer Service 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.
4. 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.
5. The SAML responder supplies back a SAML `<ArtifactResponse>` message containing the `<Authn Request>` previously generated.
6. 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.
7. The user provides valid credentials and a security context is created for the user.
8. 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.
9. 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.
10. 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.

4.1.5 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 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 14 illustrates the message flow:

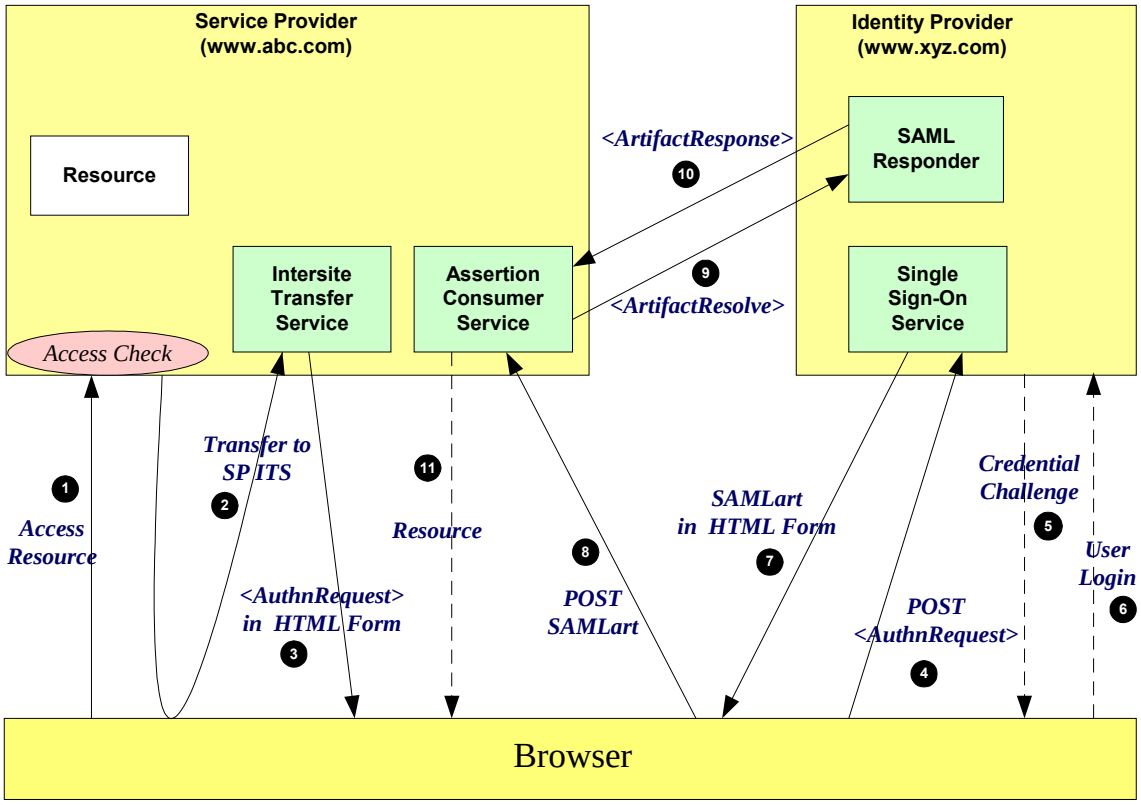


Figure 14: SP initiated: POST->Artifact binding

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 application then directs the request to the local Inter-site Transfer Service. The request contains the URL of the resource on the destination site (the TARGET URL). The URL would look something like the following (without the URL encoding):

   https://www.abc.com:8002/InterSiteTransfer?TARGET=http://www.xyz.com/index.asp
3. The Inter-site Transfer Service 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.
4. 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.
5. 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.
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.

4.1.6 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 <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 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 15 illustrates the message flow:

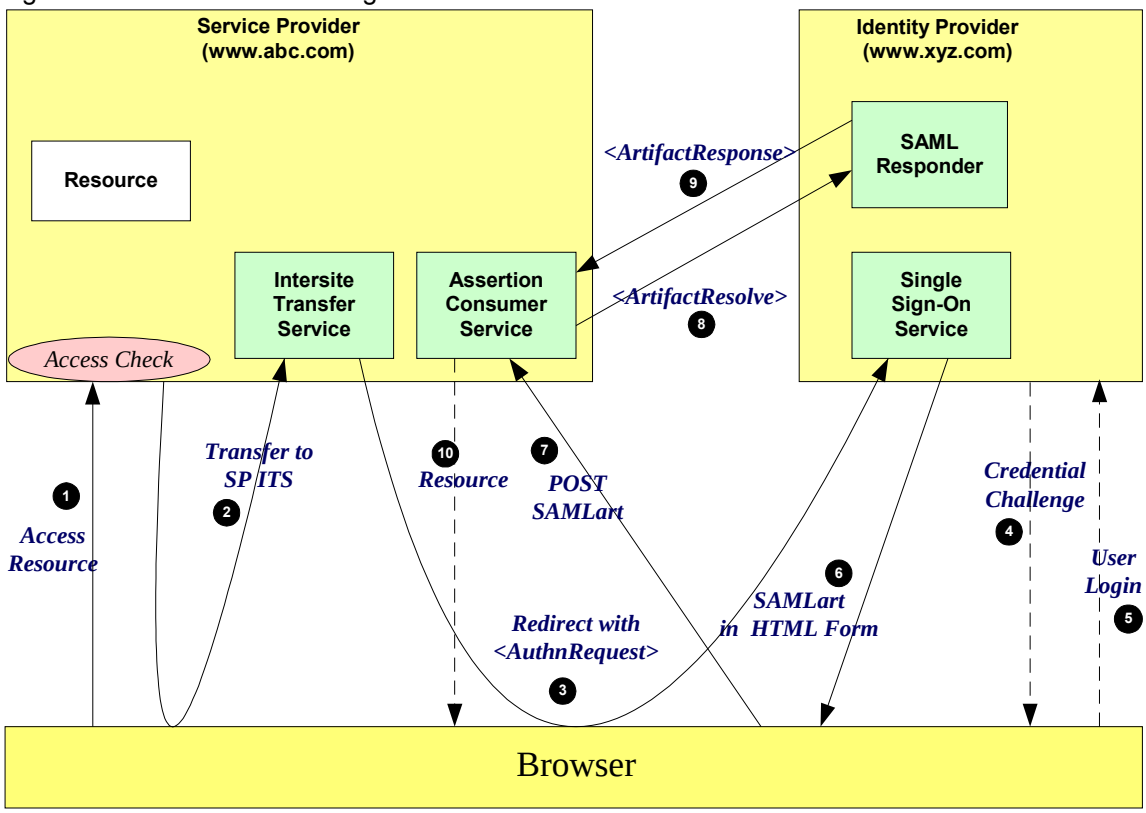


Figure 15: SP initiated: Redirect->Artifact binding

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 application then directs the request to the local Inter-site Transfer Service. The request contains the URL of the resource on the destination site (the TARGET URL). The URL would look something like the following (without the URL encoding):

   https://www.abc.com:8002/InterSiteTransfer?TARGET=http://www.xyz.com/index.asp

3. The Inter-site Transfer Service 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.
4. 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.
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.

4.1.7 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 16 illustrates the message flow:

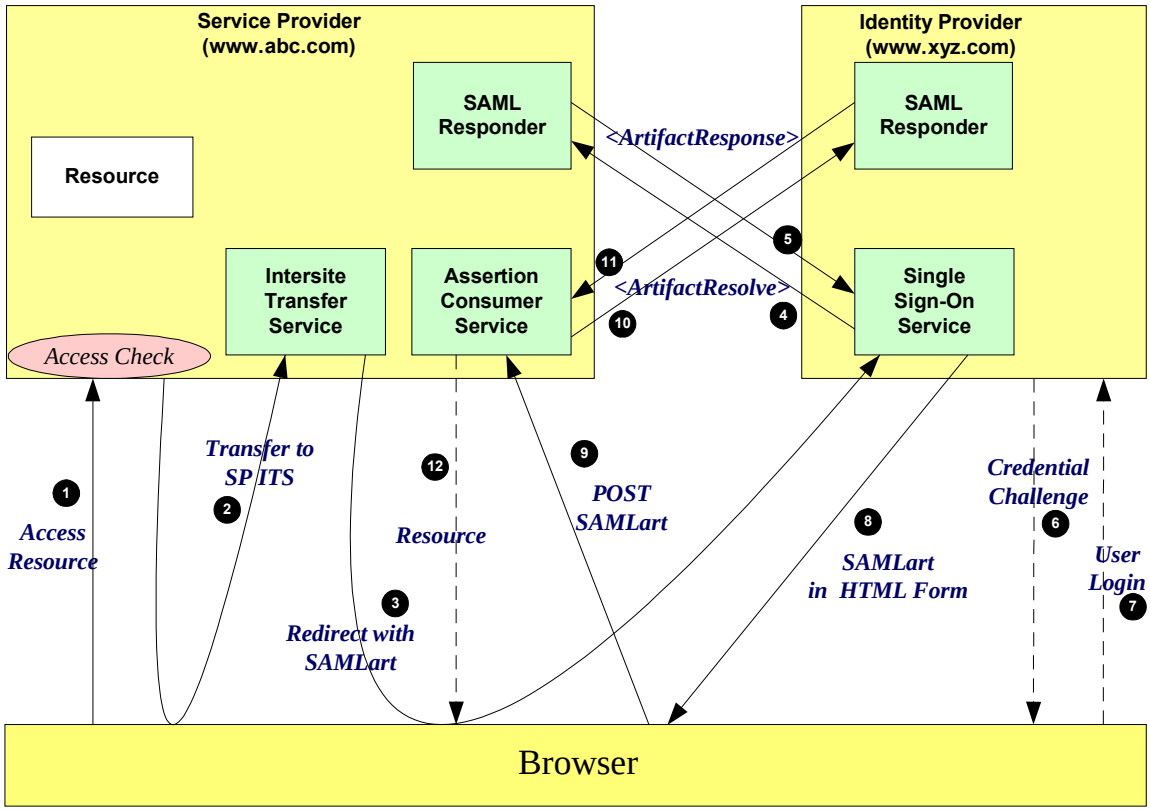


Figure 16: SP initiated: Artifact->Artifact binding

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 application then directs the request to the local Inter-site Transfer Service. The request contains the URL of the resource on the destination site (the TARGET URL). The URL would look something like the following (without the URL encoding):

   https://www.abc.com:8002/InterSiteTransfer?TARGET=http://www.xyz.com/index.asp

3. The Inter-site Transfer Service 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.
4. 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.
5. The SAML responder supplies back a SAML `<ArtifactResponse>` message containing the `<AuthnRequest>` previously generated..
6. 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.
7. The user provides valid credentials and a security context is created for the user.

8. 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.
9. 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.
10. 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.
11. 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.
12. 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 17 shows the process flow:

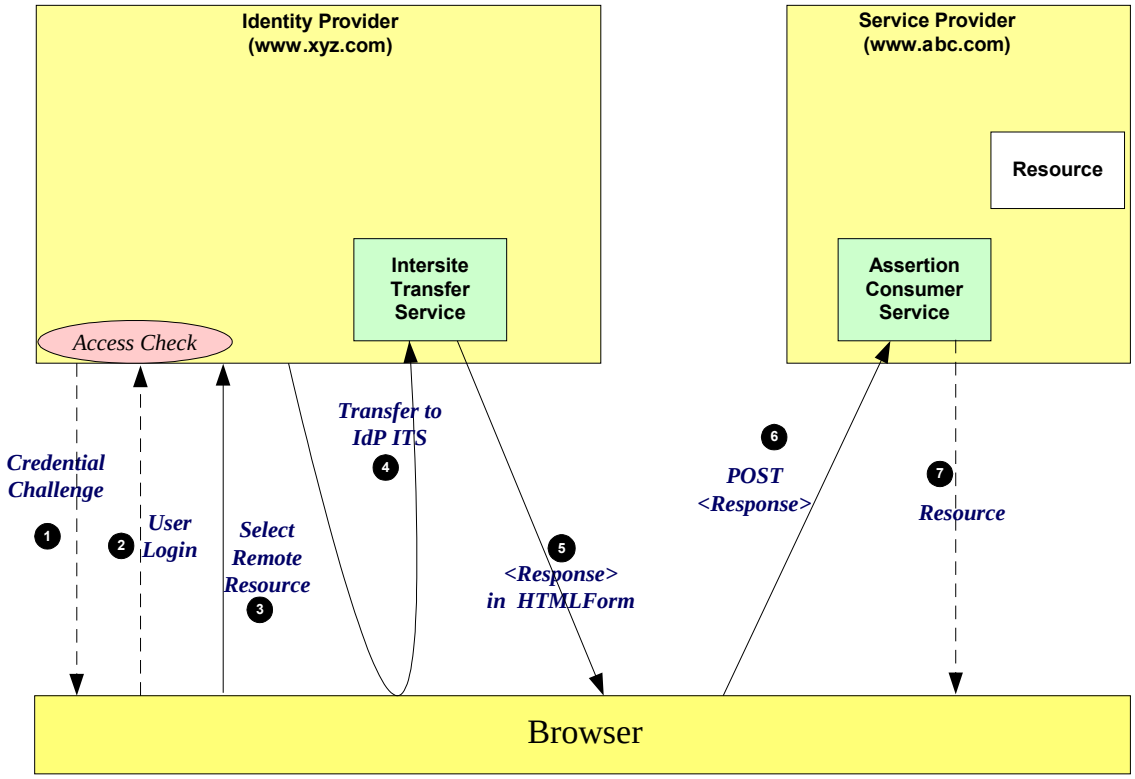


Figure 17: IdP initiated: POST binding

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 application then directs the request to the local Inter-site Transfer Service (in this example, hosted on the same web site). The request contains the URL of the resource on the destination site (the TARGET URL). The URL would look something like the following (without the URL encoding):

   https://www.xyz.com:8002/InterSiteTransfer?TARGET=http://www.abc.com/index.asp

5. The Inter-site Transfer 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' 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, 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 18 shows the process flow:

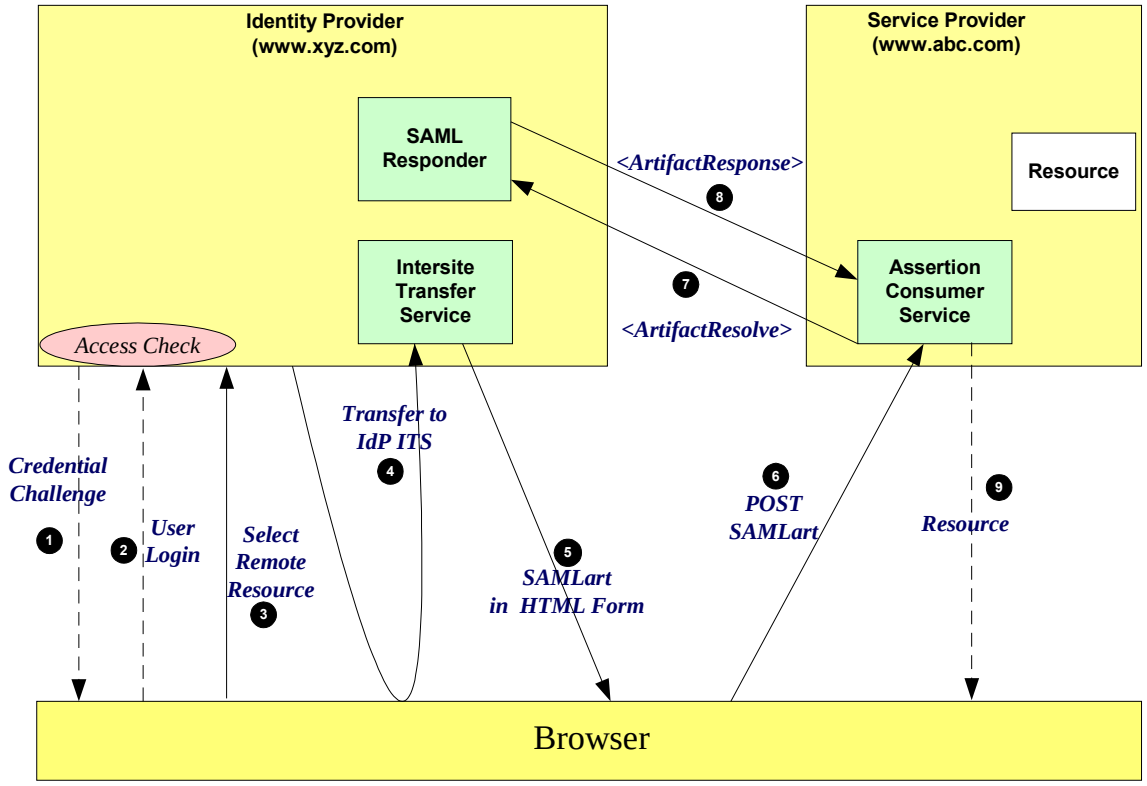


Figure 18: IdP initiated: Artifact binding

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 application causes a HTTP request to be sent to the Identity Provider's Inter-site Transfer Service. The request contains the URL of the resource on the destination site. This is known as the TARGET URL. The URL would look something like the following (without the URL encoding):

   https://www.xyz.com:8002/InterSiteTransfer?TARGET=http://www.abc.com/index.asp

5. The Inter-site Transfer 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 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.
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 its Inter-site Transfer Service.
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.

4.2 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 19 shows the message flows between the ECP, Service Provider and Identity Provider. The ECP is shown as a single logical entity.

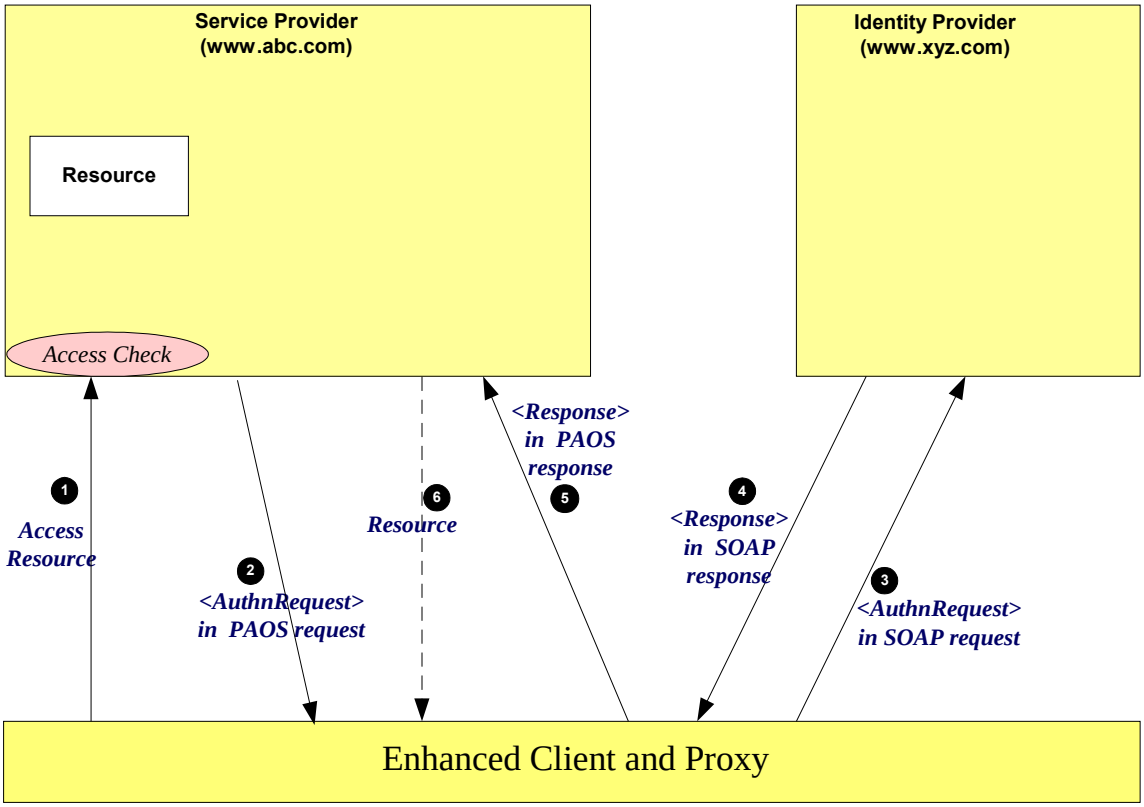


Figure 19: ECP with PAOS

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 Kerberos

TBD - TIM ALSOP

4.4 Federation

PROVIDE DETAILS OF A FEW FEDERATION SCENARIOS/PROFILES – JOHN HUGHES

4.4.1 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.

4.4.2 Federation during <AuthnRequest>

TBD – John Hughes

4.4.3 Federation Termination

TBD – John Hughes

4.4.4 Accounting Linking

TBD – John Hughes

5 Documentation roadmap

- **Security Assertion Markup Language (SAML) 2.0 Executive Overview**. (sstc-saml-exec-overview-2.0). Provides a senior executive
- **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
- sstc-saml-impl-guidelines-2.0
- **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.
- sstc-saml-schema-assertion-2.0
- sstc-saml-schema-protocol-2.0

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 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.

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 recognization 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 XML Encryption schema has been imported into the assertions [also protocols?] schema.
- 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> and <AttributeDesignator> elements 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.
- There is a new <AssertionURIReference> element to go with a new HTTP-based retrieval binding.

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 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/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.

6.11 Other Differences

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

TBS: validity period semantics and syntax extended, element and attribute name changes, terminology, wildcarding changes, removal of QNames in content, etc.

7 References

[SAML-ASSERT] Assertions and Protocols for the OASIS Security Assertions Markup Language (SAML) V2.0 (sstc-saml-core-2.0).

[SAML-BIND] Binding for the OASIS Security Assertions Markup Language (SAML) V2.0 (sstc-saml-bindings-2.0).

[SAML-PROF] Profiles for the OASIS Security Assertions Markup Language (SAML) V2.0 (sstc-saml-profiles-2.0).

[SAML-SEC] Security and Privacy Considerations for the OASIS Security Assertions Markup Language (SAML) V2.0 (sstc-saml-sec-consider-2.0)

[SAML-Meta] Metadata for the OASIS Security Assertions Markup Language (SAML) V2.0 (sstc-saml-metadata-2.0).

[SAML-AuthnCxt] Authentication Context for the OASIS Security Assertions Markup Language (SAML) V2.0 (sstc-saml-authn-context-2.0)

[XACML] TBD

Etc etc

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

992 # B. Revision History

Rev	Date	By Whom	What
00	6 Nov 2003	John Hughes	Storyboard version
01	22 Jul 2004	John Hughes	First draft

993

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.