Security Assertion Markup Language
(SAML) 2.0 Technical Overview

Working Draft 067, 1013 MayJuly 2005

Document identifier:
sstc-saml-tech-overview-2.0-draft-067

Location:

Editors:
John Hughes, PA Consulting Group
Eve Maler, Sun Microsystems

Contributors:
Hal Lockhart, BEA
Thomas Wisniewki, Entrust
Prateek Mishra, Principal Identity
Nick Ragouzis, Individual Associate

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

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-
one.org/committees/security/).
## Table of Contents

1 Introduction......................................................................................................................... 4  
2 SAML Use Cases.................................................................................................................... 5  
   2.1 Attribute Federation Use Case.......................................................................................... 5  
   2.2 Account Linking Use Case.............................................................................................. 6  
3 SAML Architecture.................................................................................................................. 8  
   3.1 Basic Concepts.................................................................................................................. 8  
   3.2 Summary of SAML Components..................................................................................... 8  
   3.3 SAML Structure and Examples....................................................................................... 10  
      3.3.1 Assertions.................................................................................................................. 10  
      3.3.2 SOAP over HTTP Binding......................................................................................... 12  
   3.4 Federated Identity Principles............................................................................................ 13  
   3.5 Use of SAML in other Frameworks................................................................................ 14  
      3.5.1 Web Services Security (WSS).................................................................................. 14  
      3.5.2 eXtensible Access Control Markup Language (XACML)....................................... 16  
3.6 Security in SAML................................................................................................................ 18  
4 Profiles.................................................................................................................................... 19  
   4.1 Web Browser SSO Profile............................................................................................... 19  
      4.1.1 Concept.................................................................................................................... 19  
      4.1.2 SP initiated: POST->POST binding....................................................................... 21  
      4.1.3 SP initiated: Redirect->POST binding................................................................... 22  
      4.1.4 SP initiated: Artifact->POST binding.................................................................... 24  
      4.1.5 SP initiated: POST->Artifact binding................................................................. 25  
      4.1.6 SP initiated: Redirect->Artifact binding.............................................................. 27  
      4.1.7 SP initiated: Artifact->Artifact binding............................................................... 28  
      4.1.8 IdP initiated: POST binding.................................................................................. 30  
      4.1.9 IdP initiated: Artifact binding............................................................................... 31  
   4.2 ECP Profile....................................................................................................................... 32  
      4.2.1 Introduction............................................................................................................. 32  
      4.2.2 ECP Profile using PAOS binding.......................................................................... 33  
4.3 Identity Federation Protocols............................................................................................ 34  
   4.3.1 Introduction............................................................................................................... 34  
   4.3.2 Single Sign-On with Out-of-band Account Linking.................................................. 34  
   4.3.3 Attribute Federation................................................................................................... 35  
   4.3.4 Persistent Federation.................................................................................................. 37  
   4.3.5 Transient Federation................................................................................................. 39  
   4.3.6 Federation Termination.............................................................................................. 41  
4.4 Single Logout...................................................................................................................... 42  
5 Documentation roadmap ....................................................................................................... 45  
6 Comparison Between SAML V2.0 and SAML V1.1............................................................. 46  
   6.1 Differences in the Organization of the Specifications.................................................... 46  
   6.2 Versioning Differences.................................................................................................... 46  
   6.3 Subject and Subject Confirmation Differences............................................................. 47  
   6.4 Encryption-Related Differences..................................................................................... 47
6.5 Attribute-Related Differences

6.6 Differences in the Request-Response Mechanism

6.7 Differences in the Protocols for Retrieving Assertions

6.8 Session-Related Differences

6.9 Federation-Related Differences

6.10 Differences in Bindings and Profiles

6.11 Other Differences

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

What are the entities involved in a SAML interaction? At the heart of most SAML assertions is a subject (a principal – an entity that can be authenticated – within the context of a particular security domain) about which something is being asserted. The subject could be a human but could also be some other kind of entity, such as a company or a computer.

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

Typically there are a number of service providers (SPs) that can make use of assertions about a subject in order to control access and provide customized services, and accordingly they become the relying parties of an asserting party called an identity provider (IdP). For example, a typical assertion from an identity provider might convey that “This user is John Doe, he has an email address of john.doe@acompany.com, and he was authenticated into this system using a password mechanism.” A service provider could choose to use this information, depending on its access policies, to grant access to local resources.
2 SAML Use Cases

Why is SAML required for exchanging security information? There are three drivers behind the creation of the SAML standard:

• **SSO interoperability:** How different products implement SSO and Cross-Domain SSO (CDSSO) are completely proprietary. Most pre-SAML Single Sign-On products use browser cookies to maintain state so that re-authentication is not required. Browser cookies are not transferred between DNS domains. So, if you obtain a cookie from www.abc.com, then that cookie will not be sent in any HTTP messages to www.xyz.com. This could even apply within a single organization that has separate DNS domains. SAML solves the CDSSO problem by providing a standard vendor-independent protocol for transferring information about a (browser-equipped) user from one web server to another without relying on cookies.

• **Web services:** Security within Web Services is still being defined. Most of the focus has been on how to provide confidentiality, authentication, and integrity services on an end-to-end basis. The SAML standard provides a means by which security assertions about messages can be exchanged between communicating service endpoints.

• **Federated identity:** Federated identity deals with the sharing of information about user identities across organizational boundaries while maintaining privacy protection. From an administrative perspective, this type of sharing can help reduce identity management costs as multiple organizations do not need to independently collect and maintain identity-related data (e.g., passwords). From a user-centric viewpoint, as explained under SSO interoperability, this also results in an enhanced user experience with fewer sign-ons. In addition, administrators do not have to maintain the mappings; rather the onus can be placed on the user.

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 Attribute Federation Use Case

This is the original use case as supported in SAML V1.0 and V1.1. 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 website (in a different DNS domain). The Identity Provider site (AirlineInc.com) asserts to the Service Provider (AirlineInc.com) that the user is known to it and provides the user's name and session attributes (e.g., "Gold member"). In this case, the user's identity is federated between AirlineInc.com and CarRentalInc.com by business agreement between the partners with certain attributes (user name, membership level) used to describe the user. 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.
2.2 Account Linking Use Case

There are a number of 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. In addition to the AirlineInc.com, users are registered on both service provider sites, but using different names. At AirlineInc.com, user joe may be registered as johndoe, on CarRentalInc.com as jdoe and on HotelBookings.com as johnd. SAML 2.0 supports a model for federated identity based upon pseudonyms. A pseudonym is a privacy preserving identifier shared between a few entities. In this use case, AirlineInc.com describes the user to CarRentalInc.com and HotelBooking.com using (distinct) pseudonyms. Each of CarRentalInc.com and HotelBooking.com can link the pseudonym to the existing user account, once user consent has been obtained. In subsequent access, the user will only need to login once to to AirlineInc.com and conduct business at CarRentalInc.com and HotelBookings.com using account information available at these sites.
Figure 2: Account Linking 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, attribute and authorization information to be exchanged between autonomous organizations. The “core” SAML specification defines the structure and content of Assertions – which carry statements about a Principal as asserted by an Asserting Party. These are defined by an XML Schema. Assertions are either requested or just “pushed” out to the Service Provider. How and which assertions are requested is defined by the SAML Protocols, which have their own XML Schema. The lower-level communication or messaging protocols (such as HTTP or SOAP) 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 to interpret attribute information carried within an Assertion using common attribute/directory technologies.

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

- Metadata: Metadata defines how to express and share configuration information between two communicating entities. 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 an XML schema. SAML defines three kinds of statements that can be carried within an assertion:
  - Authentication statements: These are issued by the party that successfully authenticated the user. They define who issued the assertion, the authenticated subject, validity period, plus other authentication related information.
  - Attribute statements: These contain specific details about the user (for example, that they have “Gold” status).
  - Authorization decision statements: These identify 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, which are encoded in an XML schema as a set of request-response pairs. The protocols defined are:
  - Assertion Query and Request Protocol: Defines a set of queries by which existing SAML
assertions may be obtained. The query can be on the basis of a reference, subject, or the
statement type.

• **Authentication Request Protocol:** Defines a protocol by which a Service Provider or Principal can request assertions from an Identity Provider tailored to the requirements of a particular SAML profile, for example the Web Browser SSO Profile.

• **Artifact Resolution Protocol:** Provides a mechanism by which protocol messages may be passed by reference using a small, fixed-length value called an artifact. The artifact receiver uses the Artifact Protocol to dereference the actual protocol message.

• **Name Identifier Management Protocol:** Provides mechanisms to change the value 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, due to a session timeout or because a user access rights have been revoked. Logout can be initiated by a provider site.

• **Name Identifier Mapping Protocol:** Provides a mechanism to programmatically map one SAML name identifier into another, subject to appropriate policy controls.

• **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 an HTTP client to be a SOAP responder. Used in the Enhanced Client and Proxy Profile and particularly designed to support WAP gateways.

  • **HTTP Redirect Binding:** Defines how SAML protocol messages can be transported using HTTP redirect messages (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:** Defines a means for retrieving a SAML assertion by resolving a URI (uniform resource identifier).

• **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 for interoperability in particular usage scenarios. Some of these are described in detail later on in the document. In summary they are:

  • **Web Browser SSO Profile:** Defines a mechanism for single sign-on by unmodified web browsers to multiple Service Providers using the Authentication Request protocol in combination with the HTTP Redirect, POST HTTP POST, and Artifact bindings.

  • **Enhanced Client and Proxy (ECP) Profile:** Defines a profile of the Authentication Request protocol in conjunction with the Reverse-SOAP and SOAP bindings suited to clients or...
gateway devices with knowledge of one or more Identity Providers.

- **Identity Provider Discovery Profile**: Defines one possible mechanism for a set of cooperating Identity and Service Providers to obtain the Identity Providers used by a Principal.

- **Single Logout Profile**: A profile of the SAML Single Logout protocol is defined. Defines how SOAP, HTTP Redirect, HTTP POST and HTTP Artifact bindings may be used.

- **Name Identifier Management Profile**: Defines how the Name Identifier Management protocol may be used with SOAP, HTTP Redirect, HTTP POST and HTTP Artifact bindings.

- **Artifact Resolution Profile**: Defines how the Artifact Resolution protocol uses a synchronous binding, for example the SOAP binding.

- **Assertion Query/Request Profile**: Defines how the SAML query protocols (used for obtaining SAML assertions) use a synchronous binding such as the SOAP binding.

- **Name Identifier Mapping Profile**: Defines how the Name Identifier Mapping protocol uses a synchronous binding such as the SOAP binding.

Figure 3 illustrates the relationship between the components:

![Figure 3: SAML Components](image)

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

### 3.3 SAML 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 it is 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, it’s 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
<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>
                sstc-saml-tech-overview-2.0-draft-0
            </saml:AuthnContextClassRef>
        </saml:AuthnContext>
    </saml:AuthnStatement>
</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 SAML request or SAML 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.doe). The SAML <AuthnRequest> has been highlighted.
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 (response ID), InResponseTo, vVersion 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.

Figure 8: SAML Response within SOAP message

### 3.4 Federated Identity Principles

SAML 2.0 supports single-sign: the transmission of information about users between providers (autonomous domains) in an interoperable and vendor-independent format. Once it is possible to share such information between providers, further questions arise concerning the properties of the transmitted information. These include such issues as the format and values transmitted, relevant processing rules at identity and service providers as well as assumptions about contents of identity stores at the providers. These tasks fall into the general category of identity management and are more specifically described by the term identity federation.

A user's identity is said to be federated between a set of providers when there is agreement between the providers on a set of identifiers and/or attributes to use when referring to the user. There are many different techniques that may be used to implement the data flows required for such agreements between providers. In some cases, some of the required exchanges of identity-related information may take place outside of the SAML 2.0 protocol using other infrastructure. For example, providers may choose to share information about newly registered or de-registered users via batch “identity feeds” that are driven by identity sources (e.g., HR databases) at the identity provider and propagated to service providers. Subsequently, the user name may be placed in a SAML assertion and propagated between providers to implement single sign-on.

SAML 2.0 includes support for certain identity federation protocols, and this functionality is detailed
further in Section 4.3, but as explained above, this should not be taken to mean that SAML-2.0 requires the use of these protocols. In some cases, as in Section 2.1, identity federation may be achieved purely by a business agreement which states that an identity provider will refer to a user based on certain attribute names and values, with no additional flows required for maintaining and updating user information between providers.

SAML-2.0 assertions allow a range of identity-related information to be conveyed between providers. The following elements are typically used to convey identity-related data:

➔ **NameIdentifier**: SAML-2.0 defines a number of different formats and corresponding values may be used to describe the user via this element. They include:

1. Email Address
2. X.509 Subject Name
3. Windows Domain Qualified Name
4. Kerberos Principal Name
5. Entity Identifier
6. Persistent Identifier
7. Transient Identifier

Of these, persistent and transient identifiers require further discussion. Transient identifiers support "anonymity" at the service provider as they correspond to a "one-time use" identifier created at the identity provider.

-Persistent identifiers support pseudonymity at service providers; they are privacy-preserving and their use is restricted to a identity and service provider pair. An extension of this concept, affiliation, permits a group of service providers to consume a single shared persistent identifier used to describe a user. Affiliations are indicated by the SPNameQualifier attribute in the `<NameID>` and `<NameIDPolicy>` elements.

• **Attribute Statements**: Identity-related information may also be conveyed via attribute statements. An example of this is using a Role attribute, for example "Gold Member".

These may modify or extend the information carried by the NameIdentifier element. For example, a combination of a persistent identifier and attributes may be used to support identity federation of the type found in Shibboleth.

• **SAML-2.0** supports several other features that are desirable when working with federated identity. Confidentiality is supported by permitting name identifiers and attribute statements to be encrypted. Providers can capture information about user consent and transmitted within SAML messages.
3.5 Use of SAML in other Frameworks

3.5.1 Web Services Security (WSS)

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

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

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

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

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

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

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

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

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

Figure 9: 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.
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. Ultimately, it was decided to allow them to proceed independently but to align them. Compatibility with SAML was written in to the Charter of the XACML TC.

As a result, SAML and XACML can each be used independently of the other, or both can be used together. Using SAML and XACML in combination would typically involve the following steps.

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

Figure 11 illustrates the typical use of SAML with XACML.
The SAML and XACML specification sets contain some features specifically designed to facilitate their combined use. The XACML Attribute Profile, which can be found in the SAML Profiles specification, defines how SAML attributes may be mapped to XACML Attributes. A schema is provided by SAML to facilitate this. The XACML specification, SAML V2.0 profile of XACML provides additional information on mapping SAML Attributes to XACML Attributes.

The SAML V2.0 profile of XACML also defines a new type of Authorization decision query specifically designed for use in an XACML environment. It extends the SAML protocol schema and provides a request and response that contains exactly the inputs and outputs defined by XACML. The same document also contains two additional features that extend the SAML schemas. While they are strictly speaking not intended primarily to facilitate combining SAML and XACML, they are worth noting. The first is the XACML Policy Query. This extension to the SAML protocol schema allows the SAML protocol to be used to retrieve XACML policy which may be applicable to a given access decision. The second feature extends the SAML schema by allowing the SAML Assertion envelope to be used to wrap an XACML policy. This makes available to XACML features such as Issuer, Validity interval and signature, without requiring the definition of a redundant or inconsistent scheme. This promotes code and knowledge reuse between SAML and XACML.

### 3.6 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 is recommended.

- When an assertion or request “pushed” to a relying party (for example using the HTTP POST binding), then it is mandated that the response message be digitally signed using the XML digital signature standard.
4 Profiles

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

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

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 is artifact carried (either as a query variable or as a control in a POST HTTP POST body). The artifact is a base-64 encoded string. It consists of a unique identity of the Identity Provider (called the Source ID) and a unique reference to the assertion (called the Message Assertion Handle). 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 artifact resolve request, containing the artifact, to the Identity Provider asking it to dereference the supplied artifact what it can assert about the user. The assertions are transferred back embedded in a SAML artifact 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.
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 potentially authorization attributes. The Service Provider directs an authentication request to their Identity Provider so that it may provider obtain back SAML assertion(s) in order to validate whether they have access rights to the resource.

Figure 13 compares the two 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 an existing session context|logon session on this site (e.g. logged on) 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:
The processing is as follows:

1. The user attempts to access a resource on www.abc.com. The user does not have any current logon session (i.e. security context) on this site, and is unknown to it.
2. The SP sends a HTML form back to the browser. The HTML FORM contains a SAML <AuthnRequest> defining the user for which authentication and authorization information is required. Typically the HTML FORM will contain an input or submit action that will result in an HTTP POST.
3. The browser, either due to a user action or via an "auto-submit", issues an HTTP POST containing the SAML <AuthnRequest> to the Identity Provider's Single Sign-On service.
4. If the user does not have an existing security context on the Identity Provider, or the policy defines that authentication is required, the 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 an HTTP POST.
7. The browser, either due to a user action or via an "auto-submit", issues an HTTP POST containing the SAML response to be sent to the Service Provider's Assertion Consumer service.
8. The Service Provider's Assertion Consumer service validates the digital signature on the SAML Response. If this validates correctly, it sends an HTTP redirect to the browser causing it to access the TARGET resource, with a cookie that identifies the local session (use of a cookie is implementation specific, other techniques to maintain the security context at the SP can be used). An access check is then made to establish whether the user has the correct authorization to access the www.abc.com web site and the TARGET resource. If the access check passes, the TARGET resource is then returned to the browser.
4.1.3 SP initiated: Redirect->POST binding

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

Figure 15 illustrates the message flow:

The processing is as follows:

1. The user attempt to access a resource on www.abc.com. The user does not have any current 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 an 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 service 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 (use of a cookie is implementation specific, other techniques to maintain the security context at the SP can be used). An access check is then made to establish whether the user has the correct authorization to access the www.abc.com web site and the TARGET resource. If the access check passes, the TARGET resource is then returned to the browser.

4.1.4 SP initiated: 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 <AuthnRequest> from the Service Provider’s SAML Responder. When the Identity Provider obtains the SAML <AuthnRequest> it provides back to the Service Provider the SAML response using the POST binding mechanism.

Figure 16 illustrates the message flow:

![Figure 16: SP initiated: Artifact->POST binding](image-url)

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
AssertionHandleMessageHandle). The HTTP Artifact binding allows the choice of either HTTP
redirection or a HTML form as the delivery mechanism to the Service Provider service provider. The
figure shows the use of the HTML form mechanism. The Inter-site Transfer Service service provider,
sends a HTML form back to the browser. The HTML FORM contains the SAML artifact, the control
name being SAMLart. Typically the HTML FORM will contain an input or submit action that will
result in a HTTP POST.

3. On receiving the HTTP message, the Single Sign-On Service, extracts the source-IDSourceID from
the SAML artifact. A mapping between source IDs and remote Responders will already have been
established administratively. The Assertion Consumer Assertion Consumer service will therefore know
that it has to contact the www.abc.com SAML responder at the prescribed URL. It sends the SAML
<ArtifactResolve> message to the Service Provider service provider’s SAML responder
containing the artifact supplied by its Inter-site Transfer Service service provider.

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

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

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

7. The Single Sign-On Service sends a HTML form back to the browser. The HTML FORM contains a
SAML response, within which is a SAML assertion. The SAML specifications mandates 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 service provider’s Assertion
Consumer service.

9. The Service Provider service provider’s Assertion Consumer Assertion Consumer service validates the
digital signature on the SAML Response. If this validates correctly!If this, and the Assertion validate
correctly, it sends a HTTP redirect to the browser causing it to access the TARGET resource,
with a cookie that identifies the local session (use of a cookie is implementation specific, other
 techniques to maintain the security context at the SP can be used). An access check is then made to
establish whether the user has the correct authorization to access the www.abc.com web site and the
TARGET resource. The TARGET resource is then returned to the browser. If the access check
passes, 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 identity provider so that the Identity Provider identity
provider can provide back a SAML assertion concerning the user. A HTTP POST message is used to
deliver the SAML <AuthnRequest> to the Identity Provider 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 service provider uses the SAML artifact to obtain the SAML response
(containing the SAML assertion) from the Identity Provider identity provider’s SAML Responder.

Figure 17 illustrates the message flow:
The processing is as follows:

1. The user attempts 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, the 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 ArtifactResolutionHandle). The HTTP Artifact binding allows the choice of either HTTP redirection or a HTML form as the delivery mechanism to the Service Provider 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 Service will therefore know that it has to contact the www.xyz.com SAML responder at the prescribed URL.
8. The **www.abc.com Assertion Consumer** service will send a SAML `<ArtifactResolve>` message to the Identity Provider's SAML responder containing the artifact supplied by the Identity Provider.

9. The SAML responder supplies back a SAML `<ArtifactResponse>` message containing the assertion previously generated. In most implementations, if a valid assertion is received back, then a session on **www.abc.com** is established for the user (the relying party) at this point.

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

### 4.1.6 SP initiated: 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 18 illustrates the message flow:

![Figure 18: SP initiated: Redirect->Artifact binding](image-url)
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-ProviderIdentity 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 ProviderIdentity 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 AssertionHandleMessageHandle). The HTTP Artifact binding allows the choice of either HTTP redirection or a HTML form as the delivery mechanism to the Service Provider 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-IdSourceID from the SAML artifact. A mapping between source IDs and remote Responders will already have been established administratively. The Assertion Consumer Assertion Consumer service will therefore know that it has to contact the www.xyz.com SAML responder at the prescribed URL.

7. The www.abc.com Assertion Consumer Assertion Consumer service will send a SAML <ArtifactResolve> message to the Identity ProviderIdentity provider’s SAML responder containing the artifact supplied by the Identity ProviderIdentity 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 Assertion Consumer service then sends a redirection message containing a cookie back to the browser. The cookie identifies the session (use of a cookie is implementation specific, other techniques to maintain the security context at the SP can be used).

   The browser then processes the redirect message and issues a HTTP POST to the TARGET resource on www.abc.com. The GET message contains the cookie supplied back by the Assertion-Consumer Assertion Consumer service. An access check is then back to established whether the user has the correct authorization to access the www.abc.com web site and the index.asp resource.

### 4.1.7 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 ProviderIdentity provider (using a HTTP redirect), which it uses to obtain a SAML <AuthnRequest> from the Service Provider service provider’s SAML Responder. When the Identity-ProviderIdentity provider obtains the SAML <AuthnRequest> it provides back to the Service Provider service provider another SAML Artifact. In this example the SAML Artifact is provided back within a HTTP POST message. The Service Provider service provider uses the SAML artifact to obtain the SAML response (containing the SAML assertion) from the Identity ProviderIdentity provider’s SAML Responder.

Figure 19 illustrates the message flow:
The processing is as follows:

1. The user attempts 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 AssertionHandleMessageHandle). The HTTP Artifact binding allows the choice of either HTTP redirection or a HTML form as the delivery mechanism to the Service Provider service provider. The figure shows the use of the HTML form mechanism. The Inter-site Transfer Service service provider sends a HTML form back to the browser. The HTML FORM contains the SAML artifact, the control name being SAMLart. Typically the HTML FORM will contain an input or submit action that will result in 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 service will therefore know that it has to contact the www.abc.com SAML responder at the prescribed URL. It sends the SAML <ArtifactResolve> message to the Service Provider’s SAML responder containing the artifact supplied by its Inter-site Transfer Service provider.

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 AssertionHandleMessageHandle). 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-IDSourceID, from the SAML artifact. A mapping between source IDs and remote Responders will already have been established administratively. The Assertion Consumer service will therefore know that it has to contact the www.xyz.com SAML responder at the prescribed URL.

9. The www.abc.com Assertion Consumer service will send a SAML <ArtifactResolve> message to the Identity Provider's SAML responder containing the artifact supplied by the Identity Provider.

10. The SAML responder supplies back a SAML <ArtifactResponse> message containing the assertion previously generated. In most implementations, if a valid assertion is received back, then a session on www.abc.com is established for the user (the relying party) at this point.

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

4.1.8 IdP initiated: POST binding

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

Figure 20 shows the process flow:
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’s Assertion Consumer service.
6. The Service Provider’s Assertion Consumer service 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 (use of a cookie is implementation specific, other techniques to maintain the security context at the SP can be used). An access check is then made to establish whether the user has the correct authorization to access the www.abc.com web site and the TARGET resource. The TARGET resource is then returned to the browser. If the access check passes, 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 (www.xyz.com) and wishes to access a resource on a remote server at a service provider (www.abc.com). An artifact is provided to the Service Provider, which its can use (e.g. that is “de-reference”) to
obtain the associated SAML response from the Identity Provider.

Figure 21 shows the process flow:

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 and MessageHandle). The HTTP Artifact binding allows the choice of either HTTP redirection or a HTML form as the delivery mechanism to the Service Provider. The figure shows the use of the HTML form mechanism. The 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 an HTTP POST.
5. On receiving the HTTP message, the Assertion Consumer Service, extracts the source IDSourceID from the SAML artifact. A mapping between source IDs and remote Responders will already have been established administratively. The Assertion Consumer Assertion Consumer service will therefore know that it has to contact the www.xyz.com SAML responder at the prescribed URL.
6. The www.abc.com Assertion Consumer Assertion Consumer service 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 Assertion Consumer service` then sends a redirection message containing a cookie back to the browser. The cookie identifies the session (use of a cookie is implementation specific, other techniques to maintain the security context at the SP can be used). The browser then processes the redirect message and issues a HTTP GET to the TARGET resource on `www.abc.com`. The GET message contains the cookie supplied back by the `Assertion Consumer Assertion Consumer service`. An access check is then back to established whether the user has the correct authorization to access the `www.abc.com` web site and the index.asp resource.

4.2 **ECP Profile**

4.2.1 **Introduction**

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

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

Figure 22 illustrates two use cases for using the ECP Profile.

The ECP profile defines a single binding – PAOS (Reserve SOAP). The Profile uses SOAP headers and SOAP bodies to transport SAML `<AuthnRequest>` and SAML `<Response>` messages between the Service Provider and the Identity Provider.
4.2.2 ECP Profile using PAOS binding

Figure 23 shows the message flows between the ECP, Service Provider and Identity Provider. The ECP is shown as a single logical entity.

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 Identity Federation Protocols

4.4

• **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 preserve user privacy.

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

4.4.1 Introduction

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

• **Single Sign-on with Out-of-Band Account Linking:** Not a true example of federation but a worth while example of what is required to be established if only the Single Sign-On features of SAML are used.

• **Attribute Federation:** Attributes of the principal, as defined by the identity provider, are used to link to the account used at the service provider.

• **Persistent Federation during <AuthnRequest>:** an Identity Provider federates the Identity Provider's Principal with the Principal's identity at the Service Provider using a persistent ID.

• **Transient Federation:** a transient ID is used to federate between the IdP and the SP.

• **Federation Termination:** termination of a Federation

• **Accounting Linking:** mapping between two existing accounts on service Providers via an Identity Provider.

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

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

4.4.2 **Single Sign-On with Out-of-band Account Linking**

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

1. The user is challenged to supply their credentials to the site **AirlineInc.com**.
2. The user successfully provides their credentials and has a security context with the **AirlineInc.com** identity provider.
3. The user selects a menu option (or function) on the **AirlineInc.com** application that means the user wants to access a resource or application on **CarRentalInc.com**.
4. The **AirlineInc.com** service provider sends a HTML form back to the browser. The HTML FORM contains a SAML response, within which is a SAML assertion about user **joe**.
5. The browser, either due to a user action or via an “auto-submit”, issues an HTTP POST containing the SAML response to be sent to the **CarRentalInc.com** Service provider.
6. The **CarRentalInc.com** service provider’s Assertion Consumer service validates the digital signature on the SAML Response. If this, and the Assertion validate correctly it creates a local session for user **joe**, based on the local joe account. It then sends an HTTP redirect to the browser causing it to access the TARGET resource, with a cookie that identifies the local session. An access check is then made to establish whether the user **joe** has the correct authorization to access the **CarRentalInc.com** web site and the TARGET resource. The TARGET resource is then returned to the browser.

### 4.4.3 Attribute Federation

Attribute Federation is when the identity provider sends an assertion to the service provider where the supplied NameID is not used to map or create a session on the SP, rather an attribute (or possibly several attributes) are used to define the account to be used.
In this example the processing is as follows:

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

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

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

The processing is as follows:

1. The user attempts to access a resource on CarRentalInc.com. The user does not have any current logon session (i.e. security context) on this site and is unknown to it.
2. The service provider sends a HTTP redirect to the identity provider (AirlineInc.com). The HTTP redirect contains a SAML <AuthnRequest> requesting that the identity provider provide an assertion about the requesting user. The request asks that the identity provider sends back a persistent identifier.
3. The user will be challenged to prove valid credentials.
4. The user provides valid credentials and a security context is created for the user. The user identifies themselves as joe. The identity provider looks up user joe in its identity store and determines the persistent name identifier to be used for this federation (61611).
5. The Single Sign-On Service sends a HTML form back to the browser. The HTML FORM contains a
The SAML response, within which is a SAML assertion. The name identifier used in the assertion is a persistent identifier. The attribute "gold member" and a membership number attribute are provided. The name \textit{joe} is not contained anywhere in the assertion.

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

7. The \textit{CarRentalInc.com} service provider’s Assertion Consumer service validates the digital signature on the SAML Response and validates the SAML assertion. The supplied name identifier is then used to look up to establish whether a previous federation has been established. If a previous federation has been established (because the name identifier maps to a local account) then go to step 9. If no federation is in existence then the user will be challenged to provide valid credentials. Optionally the user could be asked whether he would like to federate the two accounts.

8. The user provides valid credentials and identifies themselves as \textit{jdoe}. The persistent name identifier is then stored and registered with the \textit{jdoe} account along with who the identity provider is.

9. A session created is for user \textit{jdoe} and an access check is then made to establish whether the user \textit{jdoe} has the correct authorization to access the \textit{CarRentalInc.com} web site and the TARGET resource.

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

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

![Figure 27: Persistent Federation – IdP-initiated](image)

The processing is as follows:

1. The user attempts to access a resource on \textit{AirlineInc.com} which eventually will pass them over to a
2. If the user does not have a current security context they will be challenged to provide valid credentials.

3. The user provides valid credentials and a security context is created for the user. The user identifies themselves as `joe`. The identity provider looks up user `joe` in its identity store and determines the persistent name identifier to be used for this federation (61611).

4. The Single Sign-On Service sends a HTML form back to the browser. The HTML FORM contains a SAML response, within which is a SAML assertion. The name identifier used in the assertion is a persistent identifier. The attribute “gold member” and a membership number attribute are provided. The name `joe` is not contained anywhere in the assertion.

5. The browser, either due to a user action or via an “auto-submit”, issues an HTTP POST containing the SAML response to be sent to the service provider’s Assertion Consumer service.

6. The CarRentalInc.com service provider’s Assertion Consumer service validates the digital signature on the SAML Response and validates the SAML assertion. The supplied name identifier is then used to look up to establish whether a previous federation has been established. If a previous federation has been established (because the name identifier maps to a local account) then go to step 9. If no federation is in existence then the user will be challenged to provide valid credentials. Optionally the user could be asked whether he would like to federate the two accounts.

7. The user provides valid credentials and identifies themselves as `jdoe`. The persistent name identifier is then stored and registered with the `jdoe` account along with who the identity provider is.

8. A session created is for user `jdoe` and an access check is then made to establish whether the user `jdoe` has the correct authorization to access the CarRentalInc.com web site and the TARGET.

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

4.4.5 TBD Transient Federation

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

- avoid having to manage userids and passwords at the service provider. Therefore all authentication is performed at the identity provider.
- have a scheme whereby the service provider does not have to manage specific user accounts, for instance it could be a site with a “group-like” access policy.
- support a truly anonymous service

As with the Persistent Federation use case one can have SP and IdP-initiated variations. Figure 28 shows the SP-initiated use case.
The processing is as follows:

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

2. The service provider sends a HTTP redirect to the identity provider (AirlineInc.com). The HTTP redirect contains a SAML `<AuthnRequest>` requesting that the identity provider provide an assertion about the requesting user. The request asks that the identity provider sends back a transient identifier.

3. The user will be challenged to provide valid credentials.

4. The user provides valid credentials and a security context is created for the user. The user identifies themselves as joe. The identity provider looks up user joe in its identity store and creates a transient name identifier to be used for this federation (294723).

5. The Single Sign-On Service sends a HTML form back to the browser. The HTML FORM contains a SAML response, within which is a SAML assertion. The name identifier used in the assertion is a transients identifier. The attribute "gold member" and a membership number attribute (1357) are provided. The name joe is not contained anywhere in the assertion.

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

7. The CarRentalInc.com service provider's Assertion Consumer service validates the digital signature on the SAML Response and validates the SAML assertion. The supplied name identifier is then used to dynamically create a session based in the received assertion. In this example it could be the membership number attribute which maps to the jdoe account. A session created is for user jdoe and an access check is then made to establish whether the user jdoe has the correct authorization to access the CarRentalInc.com web site and the TARGET.

8. If the access check passes, the TARGET resource is then returned to the browser.
The IdP-initiated use case is shown in figure 29.

Figure 29: Transient Federation – IdP-initiated

The processing is as follows:

1. The user attempt to access a resource on AirlineInc.com which eventually will pass them over to a resource on CarRentalInc.com.
2. If the user does not have a current security context they will be challenged to provide valid credentials.
3. The user provides valid credentials and a security context is created for the user. The user identifies themselves as joe. The identity provider looks up user joe in its identity store and creates a transient name identifier to be used for this federation (294723).
4. The Single Sign-On Service sends a HTML form back to the browser. The HTML FORM contains a SAML response, within which is a SAML assertion. The name identifier used in the assertion is a transient identifier. The attribute “gold member” and a membership number attribute (1357) are provided. The name joe is not contained anywhere in the assertion.
5. The browser, either due to a user action or via an “auto-submit”, issues an HTTP POST containing the SAML response to be sent to the service provider’s Assertion Consumer service.
6. The CarRentalInc.com service provider’s Assertion Consumer service validates the digital signature on the SAML Response and validates the SAML assertion. The supplied name identifier is then used to dynamically create a session based in the received assertion. In this example it could be the membership number attribute which maps to the jdoe account. A session created is for user jdoe and an access check is then made to establish whether the user jdoe has the correct authorization to access the CarRentalInc.com web site and the TARGET.
7. If the access check passes, the TARGET resource is then returned to the browser.
4.4.6 Federation Termination

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

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

Figure 30: Federation Termination

In this example the processing is as follows:

1. The service provider, CarRentalInc.com, determines that the local account, jdoe, should no longer be federated. An example of this could be that the account has been deleted. The service provider sends to the AirlineInc.com identity provider a <ManageIDNameRequest> defining that the persistent identifier (previously established) must no longer be used. The request is carried in a SOAP message which is transported using HTTP, as defined by the SAML SOAP binding. The request is also digitally signed by the service provider.

2. The identity provider verifies the digital signature ensuring that the <ManageIDNameRequest> originated from a known and trusted service provider. The identity Provider processes the request and returns a <ManageIDNameResponse> containing a suitable status code response. The response is carried within a SOAP over HTTP message and is digitally signed.

4.5 Accounting Linking Single Logout

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

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

The processing is as follows:

1. A user was previously authenticated by the _AirlineInc.com_ identity provider and is interacting with the _CarRentallnc.com_ service provider. The user decides to terminate their session and logout.

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

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

If in step 3 the identity provider determines that other service providers are participants in the session, then the identity provider will send _<LogoutRequest>_ messages to them. Figure 32 illustrates this processing. Notice that different bindings are used between the two different exchanges with the service providers. One using the redirect binding the other using a back channel, illustrating the point that different combinations of bindings can be used.
The two previous examples showed the user instigating the logout. Of course the service provider itself could initiate the logout, and in that case step 1 would not occur. There is one other use case possible, and that is when the identity provider initiates the logout. Figure 33 illustrates this example.
5 Documentation roadmap

- **Security Assertion Markup Language (SAML) V2.0 Executive Overview.** Provides a brief overview of SAML and describes its primary benefits.

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


- **Profiles for the OASIS Security Assertions Markup Language (SAML) V2.0** (sstc-saml-profiles-2.0). Defines how the assertions, protocols and bindings combine to define specific profiles.

- **Conformance Program Specification for the OASIS Security Assertions Markup Language (SAML) V2.0** (sstc-saml-conform-2.0). Describes the program and technical requirements for SAML conformance.

- **Metadata for the OASIS Security Assertions Markup Language (SAML) V2.0** (sstc-saml-metadata-2.0). Describes metadata format to enable configuration data to be shared in a standardized format.

- **Glossary for the OASIS Security Assertions Markup Language (SAML) V2.0** (sstc-saml-glossary-2.0). Defines terms used throughout the OASIS Security Assertion Markup Language (SAML) specifications.

- **Authentication Context for the OASIS Security Assertions Markup Language (SAML) V2.0** (sstc-saml-authn-context-2.0). Defines a syntax for the definition of authentication context declarations.
6 Comparison Between SAML V2.0 and SAML V1.1

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

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

The on-the-wire representations of SAML V2.0 assertions and messages are incompatible with SAML V1.x processors. As is explained in the SAML assertions and protocols specification [SAMLCore], only new major versions of SAML (of which this is one) typically cause this sort of incompatibility. However, most such incompatibility is syntactic in nature; the expressiveness of SAML has increased rather than markedly changed.

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

6.1 Differences in the Organization of the Specifications

- The assertion and 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.
- There is a new non-normative executive overview.
- The conformance specification now serves explicitly as the entry point for the SAML V2.0 OASIS Standard specifications.

6.2 Versioning Differences

- The SAML assertions namespace (known by its convention prefix saml:) and protocols namespace (known by its conventional prefix samlp:) now contain the string “2.0” in recognition of this new major version of SAML.
- The MajorVersion and MinorVersion attributes that appeared on various elements have been changed to a single Version attribute that must have the value “2.0”.
- A series of changes planned during SAML the V1.x design cycles have been made:
  - The deprecated <AuthorityBinding> element has been removed.
  - The deprecated <RespondWith> element has been removed.
  - The deprecated name identifier and artifact URI-based identifiers have been removed.
  - URI references are now required to be absolute.
  - The description of appearance of the <Status> element in SOAP messages has been improved.
6.3 Subject and Subject Confirmation Differences

- The `<SubjectStatement>` element and its type have been removed.
- The `<Subject>` element has been moved up to appear on the `<Assertion>` element, where the subject so specified applies to all enclosed statements. The `<Subject>` element is now optional for extensibility reasons, but is required for all SAML-specified statement types.
- The new `BaseID` complex type is an extension point that permits non-string identification of subjects.
- The `<SubjectConfirmation>` element is now repeatable, with the formerly repeatable `<ConfirmationMethod>` element now an attribute within that element.
- The `<ds:KeyInfo>` element is now allowed only inside `<SubjectConfirmationData>`. Further, the usage of `<ds:KeyInfo>` within `<SubjectConformationData>` has been clarified to more clearly allow for impersonation.
- A set of generic attributes in `<SubjectConfirmationData>` have been defined for use in constraining the bearer method or other confirmation methods. Overall assertion validity is more flexible within profiles that use bearer as a result.

6.4 Encryption-Related Differences

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

6.5 Attribute-Related Differences

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

6.6 Differences in the Request-Response Mechanism

- The request datatype hierarchy has been reorganized; all queries are now kinds of requests, not inside requests, and the plain `<Query>` has been removed.
- `Consent` and `<Extensions>` constructs have been added to all requests and responses.
- The `Issuer` field is now an element and is based on the same datatype that underlies name identifiers, for more unified treatment. Also, in addition to appearing on assertions, it now appears on requests and responses as well.
- The response type hierarchy has been reorganized; most response elements in the various protocols are simply of `StatusResponseType`.
- New status codes have been added to reflect possible statuses when using the new protocols. Status codes are now URIs instead of QNames.
6.7 Differences in the Protocols for Retrieving Assertions

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

6.8 Session-Related Differences

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

6.9 Federation-Related Differences

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

6.10 Differences in Bindings and Profiles

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

6.11 Other Differences

• A number of elements, attributes, and types have been renamed for brevity and consistency. List them.
• The SAML schema extensibility mechanisms have been rationalized and, in some cases, enhanced. XSD element substitution has been blocked in favor of type extension. The `<xs:anyAttribute>` wildcard has been added selectively to structures where it has been deemed valuable to add arbitrary “foreign” attributes without having to create a schema extension; these structures include subject confirmation data and attributes.
• The notion of special “SAML namespaces” (attribute namespaces and action namespaces) has been deemphasized, with attribute namespaces being removed entirely in favor of URIs as attribute format identifiers.

• The `<ds:Signature>` that allows for the digital signing of assertions and messages has been positioned earlier in the respective content models.

• The authorization decision feature (statement and query) has been frozen; if more functionality is desired, it is suggested that XACML [XACML] be used.

• Two new conditions, `<ProxyRestriction>` element and `<OneTimeUse>`, have been added. The relationship of the latter to the `NotBefore` and `NotOnOrAfter` conditions has been delineated.

• The terminology used to describe various SAML system entities has been rationalized and enhanced to incorporate the Liberty Alliance notion of “identity providers” as opposed to “authentication authorities” and similar.

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


A. Acknowledgments

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

• TBD
### B. Revision History

<table>
<thead>
<tr>
<th>Rev</th>
<th>Date</th>
<th>By Whom</th>
<th>What</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Nov 6, 2003</td>
<td>John Hughes</td>
<td>Storyboard version</td>
</tr>
<tr>
<td>01</td>
<td>Jul 22, 2004</td>
<td>John Hughes</td>
<td>First draft</td>
</tr>
<tr>
<td>02</td>
<td>27 Sept 2004</td>
<td>John Hughes</td>
<td>Second Draft. General updates, limited distribution</td>
</tr>
<tr>
<td>03</td>
<td>Feb 20, 2005</td>
<td>John Hughes</td>
<td>DCE/Kerberos use section removed. Use of SAML in other frameworks added. SAML-2.0 V2.0 XML examples included. Updated Web SSO examples to remove use of ITS</td>
</tr>
<tr>
<td>04</td>
<td>10 Apr 2005</td>
<td>Eve Maler</td>
<td>Edits based on comments made by myself and Scott Cantor. Fleshed out the list of 1.1-&gt;2.0 differences, but it's not complete yet. More work to come.</td>
</tr>
<tr>
<td>05</td>
<td>May 10, 2005</td>
<td>Prateek Mishra</td>
<td>Updated Section 2 and 3.4, Section 4.3 remains incomplete</td>
</tr>
<tr>
<td>06</td>
<td>Jun 3, 2005</td>
<td>John Hughes</td>
<td>Added Section 4.3 plus a few minor corrections</td>
</tr>
<tr>
<td>07</td>
<td>Jul 13, 2005</td>
<td>John Hughes</td>
<td>Addressed comments from SSTC, primarily re-vamping section 4.3</td>
</tr>
</tbody>
</table>
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 2005. 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.