Security Assertions Markup Language

Core Assertion Architecture – Examples and Explanations

Phillip Hallam-Baker *VeriSign*

Draft Version 0.7: May 14 2001

Printed on Monday, May 14, 2001

Security Assertions Markup Language

Version 0.7

Table Of Contents

Table of Figures

Executive Summary

A straw-man architecture is proposed to elucidate the architectural implications of the requirements implicit in the SAML use cases document.

1 Introduction

1.1 Parties

In its simplest form an assertion concerns three distinct parties, the *Issuing Party* who originates the assertion, the *Relying Party* that reads the assertion and the *Subject* who is the party that the assertion is a statement about (Figure 1).

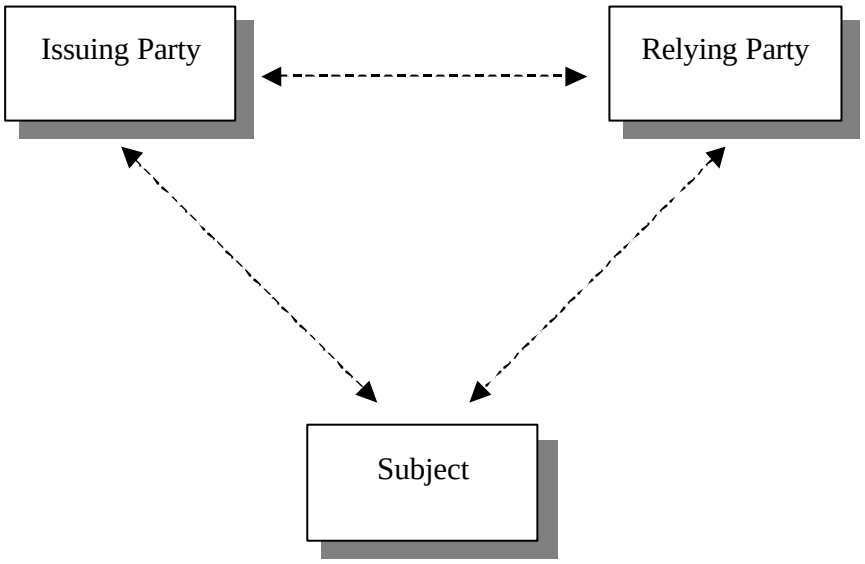


Figure 1: Parties to the protocol

The relationships between the three parties may or may not be expressed in the protocol dataflow. An application might reasonably apply the message specifications defining exchanges between the issuing and relying parties to an application in which the subject was not a protocol participant, for example the exchange of credit rating data.

In addition a particular protocol exchange may be divided into multiple 3-corner relationship models. Figure 2 shows an example involving two separate assertions that both refer to the same subject. The Issue Point issues an assertion that states that the principal has a particular attribute. The Policy Decision Point relies on this assertion to issue a second assertion that states that the Principal is allowed assess to a particular resource. The Policy Enforcement Point relies upon the latter assertion to grant or deny access to the resource in question.

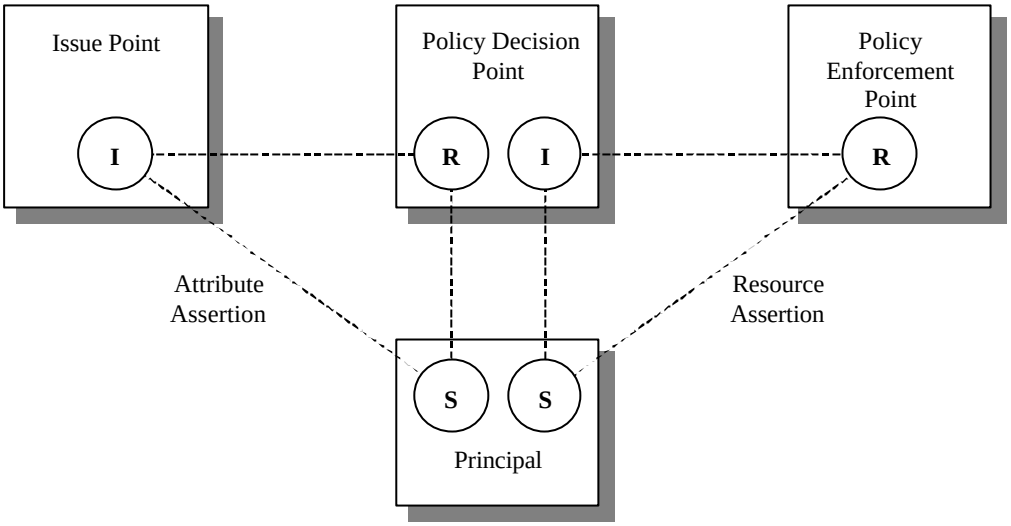


Figure 2: Multiple Assertion Relationships

1.1.1 Principal

A *Principal* is in each case the *Subject* of a SAML assertion.

1.1.2 Authentication Authority

An *Authentication Authority* is the Issue Point of a SAML authentication assertion.

1.1.3 Attribute Authority

An *Attribute Authority* is the Issue Point of a SAML attribute assertion.

1.1.4 Policy Decision Point

A *Policy Decision Point* (PDP) is the relying party of SAML assertions issued by authentication authorities, attribute authorities and other Policy Decision Points. A SAML Policy Decision Point is the issue point of a SAML decision assertion.

1.1.5 Policy Enforcement Point

The Policy Enforcement Point (PEP) is by definition the relying party of an SAML decision assertion.

1.2 Data Objects

1.2.1 SAML Assertion

A SAML Assertion is an XML data structure that makes a security assertion. Typical assertions include:

- The party with account ID Alice has the *Plumber* right.
- The Party with the account ID Alice is permitted to access resource X

Assertions may express authorization data syntactically in one of two ways:

1. As a URI identifying either a resource itself (i.e. a URL of the resource) or a rights identifier associated with the resource (e.g. via a URN). The mapping of rights identifiers to resources themselves may be achieved using SAML or through another mechanism outside the scope of the specification.
2. By incorporating additional elements into the assertion that are defined in a separate schema.

Each assertion shares a common set of XML elements specifying information about the assertion, including:

- A URI that uniquely identifies the assertion
- Status of the assertion
- Validity interval
- Conditions placed on validity
- Additional information relating to the assertion.

1.2.2 Ticket

A *ticket* is an assertion encoded as a compact data structure that identifies a particular assertion. A ticket MAY be authenticated and MAY carry encrypted data.

The principal purpose of tickets is to support the constraints imposed by zero footprint clients. It is not possible to encode all the information encoded in an assertion in the minimal space available in a URL fragment or HTTP cookie.

A second use of tickets is to provide a lightweight means of communicating cryptographic keying material in the manner of Kerberos [Kerberos].

A possible syntax for encoding tickets is provided in Appendix A . Issuing servers and relying servers may use a different ticket format by private agreement however.

For architectural purposes it is desirable that tickets have the following properties:

- Be compact, allowing the minimum data set to be encoded in 64 bytes or less.
- Support authentication by means of a shared key
  [Could add option to do a DSA signature]

- Support encryption by means of a shared key
- Specify the account identifier of the party to whom the ticket was issued and whether the identifier was authenticated.
- Allow encoding of authentication data (e.g. a shared key established between the client and issuing server)
- Be extensible to allow applications to encode data from arbitrary XML assertion elements.

2 Example Messages

[This section is included to provide an illustration of the data flows, it is not normative however and should be moved to the use case / overview document]

2.1 Web Browser Password Access

Alice is a customer of the business exchange; she needs to access a resource at Carol's store that is restricted to members of the exchange.

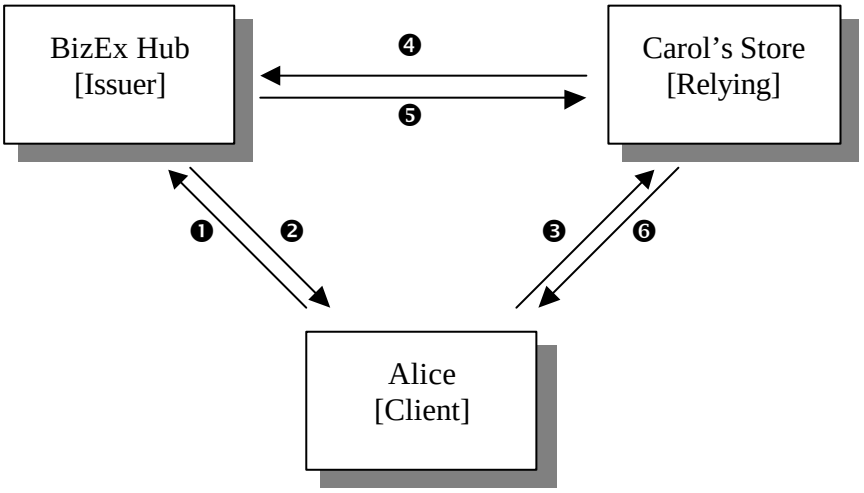


Figure 3: Web Server Log In

Message	Format	Data
❶ Login	HTTP/SSL Request	Username, Password
❷ Response	HTTP/SSL Response, Ticket (as HTML URL)	Ticket = Account, Validity, Assertion_ID Authenticator

Message	Format	Data
❸ Access	HTTP/SSL Request	Ticket
❹ Pull Assertion	XP Request	Assertion_ID
❺ Assertion	XP Response	Assertion (see below)
❻ Resource	HTTP/SSL Response	Resource Data

2.1.1 ❶ Login

The login data is posted in response to the following HTML form:

```
<form method="POST" action="https://login.bizex.test/login.asp">
  <p>Username <input type="text" name="username" size="20"><br>
  Password <input type="password" name="Password" size="20"><br>
  <input type="submit" value="Submit" name="B1"><input type="reset"
value="Reset" name="B2"></p>
</form>
```

Alice enters “Alice” as her username and “secret” as her password. This data is encoded as follows:

```
username=Alice&password=secret
```

2.1.2 ❷ Response

The business exchange service authenticates the username and password [resented by Alice and issues the ticket. The ticket contains the following data:

Item	Size	Data
Assertion_ID	7+2	`[10.20.1.123] AE 02 21`
Validity	4+2	`10-Mar-2001 12:00 for 24 hours`

Account	5+2	"Alice"
Authentication	20+2	HMAC-SHA1 (Assertion_ID, Validity, Account)
	44	

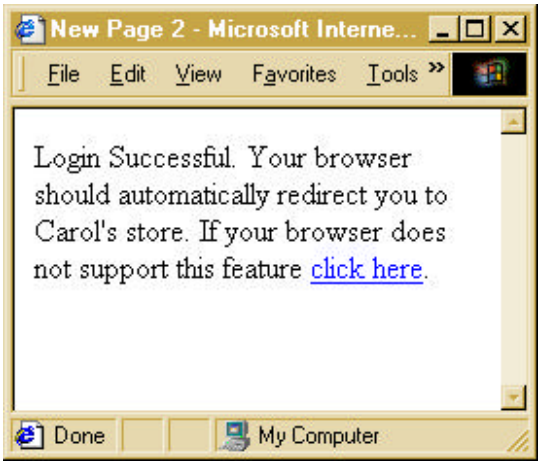
Using base64 encoding this results in a 60 byte string which is passed to Carol encoded as a URL:

```
<% Response.Redirect
"https://store.carol.test/finance/bizex.asp?ticket=jubafOqNEpcwR3RdFsT7
bCqnXPBe5ELh5u4VEy19MzxkXRgrMvavzyBpVR==" %>
<html>
<head><title>Carol's Store</title</head>

<body>

<p>Login Successful. Your browser should automatically redirect you to
Carol's store. If your browser does not support this feature <a
href="https://store.carol.test/finance/bizex.asp?ticket=jubafOqNEpcwR3R
dFsT7bCqnXPBe5ELh5u4VEy19MzxkXRgrMvavzyBpVR==">click
here</a>.</p>

</body>
</html>
```



2.1.3 ❸ Access

Alice's Web browser is redirected to Carol's Web site. The access ticket is encoded in the URL:

```
https://store.carol.test/finance/bizex.asp?ticket=jubafOqNEpcwR3RdFsT7b
CqnXPBe5ELh5u4VEy19MzxkXRgrMvavzyBpVR==
```

2.1.4 ❹ Pull Assertion

Carol's store receives the URL and decodes the ticket. This tells the server that:

- The issuer of the ticket belongs to the domain `10.20.1.123`. The security policy of Carol's store recognizes this domain identifier as 'Bob's Business Exchange"
- The HMAC value of the ticket agrees with the value calculated from a shared symmetric key exchanged out of band with Bob's Business Exchange'.
- The ticket was issued to a party that the issuing server authenticated as "Alice".
- The current time is within the validity interval of the ticket.
- More information may be obtained from the specified assertion.

In this case the assertion reference can be resolved directly since it encodes the IPv4 address of the assertion server and a unique assertion reference.

Alternate means of identifying the assertion

- Compact Locator: IPv4 Address + serial number
- Compact Locator: IPv6 Address + serial number
- Compressed URI [Use 6bit->8bit compression on ASCII URL]
- Compact Name: Large pseudo-unique number
- Serial number alone [does not work across domains]

Carol's store has a policy of accepting a ticket from Bob's Business Exchange as proof that a person is a member of the Business Exchange. Certain pages on Carol's site MAY be accessible using locally managed authorization data and the authorization ticket.

The ticket is an assertion in its own right, typically the ticket encodes a subset of the data encoded in the full assertion.

Access to the resource requested by Alice in this instance requires specific authorization. Carol's store therefore requests that the issuer supply the full assertion.

```
http://10.20.1.123/?assertion=AE0221
```

Alternatively the ticket might not be bound to a specific assertion and specify only the authenticated account (possibly pseudonymous).

```
<SAMLQuery>
   <RequestID>urn:random:32q4schaw983y5982q35yh98q324==
   <Query>
      <Binding>
```

```
         <Subject>
            <NameID>mailto:Alice@bizex.test
         <Object>
            <Authority>
               <Permission>Read
               <Resource>http://store.carol.test/finance
   <Respond>
      <string>Assertion
```

This mode of interaction is useful when the number of resources to which access is controlled is large and the Policy Enforcement Point (in this case Carol's store) does not support de-referencing of higher-level abstractions such as rights.

2.1.5 ❺ Assertion

The assertion specifies the authorizations attached to the Alice account:

```
<SAMLQueryResponse>
   <RequestID>urn:random:32q4schaw983y5982q35yh98q324==
   <Assertion>
      <AssertionID>http://www.bizexchange.test/assertion/AE0221
      <Issuer>URN:dns-date:www.bizexchange.test:2001-01-03:19283
      <ValidityInterval>
         <NotBefore>
         <NotOnOrAfter>
      <Conditions>
         <Audience>http://www.bizexchange.test/rule_book.html
      <Claims>
         <Subject>
             <NameID>mailto:Alice@bizex.test
         <Object>
            <Authority>
               <Permission>Read
               <Resource>http://store.carol.test/finance
            <Role>URN:dns-date:www.bizexchange.test:2001-01-
04:right:finance
```

This assertion specifies that Alice is authorized to access two resources:

- The web pages in the tree `http://store.carol.test/finance`
- Resources mapped to the domain specific rights identifier "finance"

The assertion also specifies that it is addressed to a specific audience – informally members of the business exchange, more specifically it is the parties that agree to be bound by the exchange rule book.

2.1.6 ❻ Resource

Carol's store receives back the assertion and authenticates it. The assertion may be authenticated by means of a secure transport layer, by and XML Signature Digital Signature or MAC, or other means.

The mapping from the resources specified in the assertion is under control of the resource owner. In this case the resource owner simply performs a direct mapping from the first resource identified in the assertion to the site. For a more comprehensive authorization decision see Section 2.5 .

To simplify further accesses Carol's store issues two cookies to Alice's browser marked as 'ephemeral', i.e. not to be saved to disk.

1. The ticket issued by Bob's Business Exchange
2. An additional authenticated cookie issued by Carol that specifies authorizations extracted or derived from the assertion.

2.2 Alternative Version

If the PEP lacks the ability to cache or process assertion data the interaction protocol may be simplified by requesting only the authorization decision.

2.2.1 ❹ Pull Assertion

Instead of requesting the assertion describing the set of objects that Alice might attempt to access the PEP asks for a specific decision on the specific resource in question:

```
<SAMLQuery>
   <RequestID>urn:random:zslkiut098q2374haw4987zset08t==
   <Query>
      <Binding>
         <Subject>
            <NameID>mailto:Alice@bizex.test
         <Object>
            <Authority>
               <Permission>Read
               <Resource>http://store.carol.test/finance
   <Respond>
      <string>Decision
```

2.2.2 ❺ Decision

The response specified that the result of the access request:

```
<SAMLQueryResponse>
   <RequestID>urn:random:zslkiut098q2374haw4987zset08t==
   <Decision>Permit
```

This assertion specifies that Alice is authorized to access two resources:

- The web pages in the tree `http://store.carol.test/finance`
- Resources mapped to the domain specific rights identifier "finance"

The assertion also specifies that it is addressed to a specific audience – informally members of the business exchange, more specifically it is the parties that agree to be bound by the exchange rule book.

2.3 Pull Model

2.3.1 ❶ Principal to Authority (request)

2.3.2 ❷ Authority to Principal (response)

2.3.3 ❸ Principal to PEP (request)

2.3.4 ❹ PEP to Principal

2.3.5 ❺ PEP to PDP (request)

2.3.6 ❻ PDP to PEP (response)

2.3.7 ❼ PDP to Authority (request)

2.3.8 ❽ Authority to PDP (response)

2.4 SSL Certificate Based Client Authentication

In this scenario Alice authenticates herself by means of a public key mechanism, this avoids the need to perform an initial authentication exchange with the business exchange prior to visiting Carol's store.

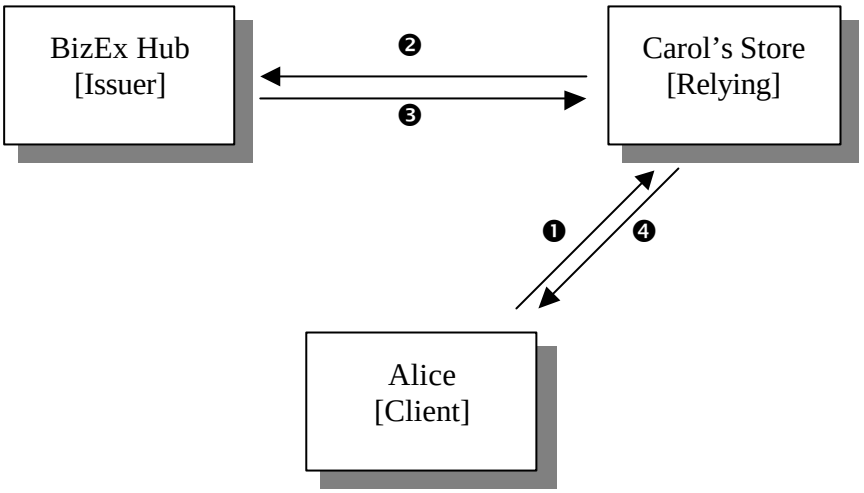


Figure 4: Certificate Based Client Auth

Message	Format	Data
❶ Request	HTTP/SSL Request (with certificate based client authentication)	Certificate, Resource
❷ Pull Assertion	XP Request	Certificate
❸ Assertion	XP Response	Assertion (see below)
❹ Resource	HTTP/SSL Response	Resource Data

2.4.1 ❶ Request

The client authenticates itself to Carol's store using a public key based challenge response scheme, in this case SSL certificate based client authentication. The details of this protocol are not visible to the SAML layer which receives only the result of the authentication, the resource request itself and the credential under which it was authenticated (in this case the certificate).

2.4.2 ❷ Pull Assertion

Carol's store requests authorization information from Bob's Business Exchange:

```
<SAMLQuery>
   <RequestID>urn:random:aw3s5098swe45w30462j09245==
   <Query>
      <Binding>
         <Subject>
            <NameID>mailto:Alice@bizex.test
            <Authenticator>
               <ds:KeyInfo>
                  <ds:X509Data>...
```

```
         <Object>
            <Authority>
               <Permission>Read
               <Resource>http://store.carol.test/finance
   <Respond>
      <string>Assertion
```

2.4.3 ❸ Assertion

The business exchange responds that any party authenticating itself with the specified credentials is authorized to access the specified resources:

```
<SAMLQueryResponse>
   <RequestID>urn:random:aw3s5098swe45w30462j09245==
   <Assertion>
      <AssertionID>http://www.bizexchange.test/assertion/AE0221
      <Issuer>URN:dns-date:www.bizexchange.test:2001-01-03:19283
      <ValidityInterval>
         <NotBefore>
         <NotOnOrAfter>
      <Conditions>
         <Audience>http://www.bizexchange.test/rule_book.html
      <Claims>
         <Subject>
            <NameID>mailto:Alice@bizex.test
            <Authenticator>
               <ds:KeyInfo>
                  <ds:X509Data>...
         <Object>
            <Authority>
               <Permission>Read
               <Resource>http://store.carol.test/finance
            <Role>URN:dns-date:www.bizexchange.test:2001-01-
04:right:finance
```

2.4.4 ❹ Resource

The resource is returned to Alice.

2.5 Server Authorization Delegation

In this example Carol's store uses SAML for internal exchange of authorization data. Authorization decisions are controlled by a central Policy Decision Point (PDP) which is consulted by the store server that receives the access request from Alice, the Policy Enforcement Point.

In the interests of completeness the Policy Decision Point consults a separate Policy Store to obtain the access policy for the resource in question. This is however, an extreme example. Few applications would require this degree of granularity. In a more typical example the functions of the Policy Decision Point would be combined with those of either the Policy Store or the Policy Enforcement Point.

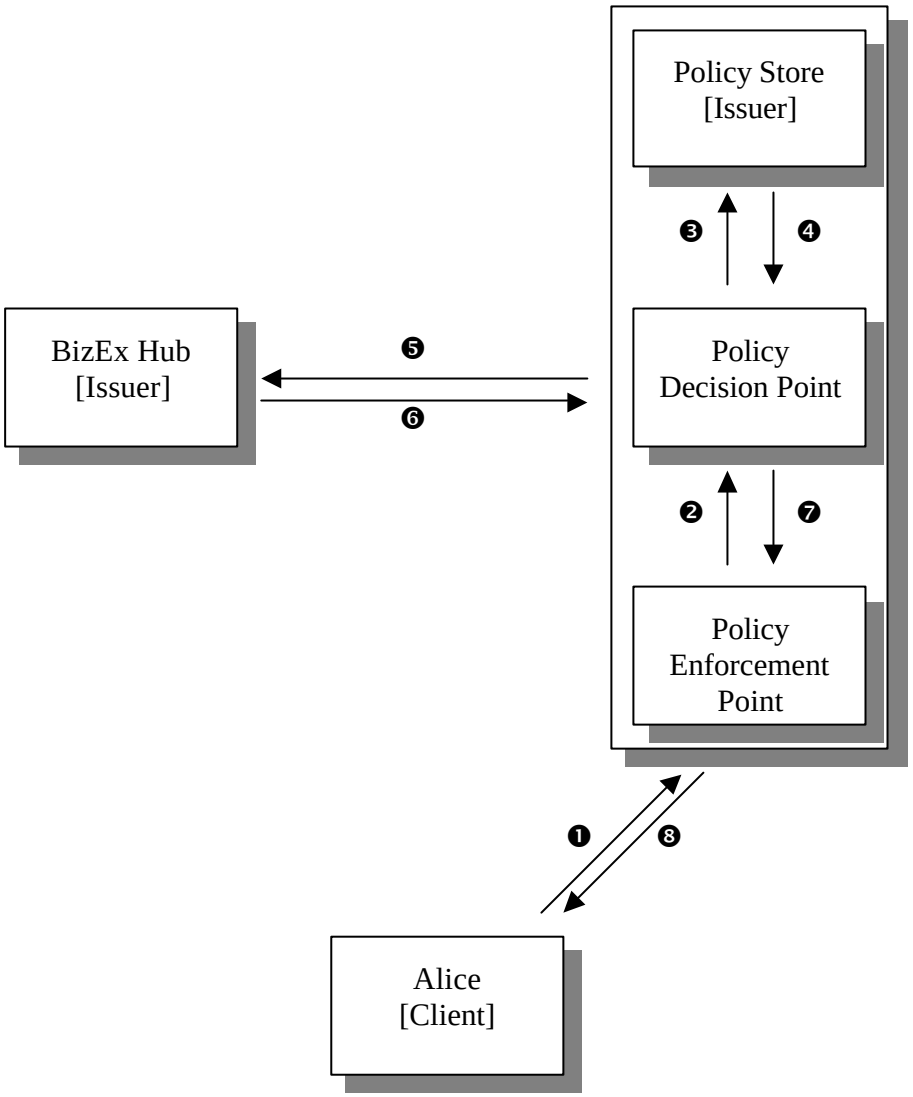


Figure 5: Delegated Decision Point

2.5.1 ❶ Request

Alice requests a resource from Carol's store. Alice authenticates herself, either by means of public key or as in this case by a ticket issued by Bob's Business Exchange.

```
https://store.carol.test/finance/bizex.asp?ticket=jubafOqNEpcwR3RdFsT7b
CqnXPBe5ELh5u4VEy19MzxkXRgrMvavzyBpVR==
```

2.5.2 ❷ Request Access Decision

The server receiving the access request delegates authorization decision processing to a Policy Decision Point.

```
<SAMLQuery>
   <RequestID>urn:random:aks4ewht98a2745hwa498720o9i9==
   <Query>
```

```
      <Binding>
         <Subject>
            <NameID>mailto:Alice@bizex.test
            <Authenticator>
               <Protocol>http://oasis/schema-sstc/bindings-ticket-01

<Authdata>jubafOqNEpcwR3RdFsT7bCqnXPBe5ELh5u4VEy19MzxkXRg
               rMvavzyBpVR==
         <Object>
            <Authority>
               <Permission>Read
               <Resource>http://store.carol.test/finance
   <Respond>
      <string>Decision
```

Note that responsibility for authenticating the authentication ticket MAY be placed on either the Policy Enforcement Point or the Policy Decision Point or both under local configuration control.

Depending on circumstances the Policy Enforcement point may require the PDP to return an assertion or just the result of the decision. In this instance only the result is required.

2.5.3 ❸ Request Access Policy

The Policy Decision Point makes a request for an access control policy for the specified resource from the Policy Issuing Server. The format in which the access policy is requested is outside the scope of SAML. A typical policy request might be:

```
<TBS-POLICY-Query>
   <RequestID>urn:random:zwos43i55098w4tawo3i5j09q==
   <Query>
      <Binding>
         <Object>
            <Authority>
               <Resource>http://store.carol.test/finance
   <Respond>
      <string>Decision
```

2.5.4 ❹ Access Policy

The format in which the access policy is specified is outside the scope of SAML. A typical policy request might be:

```
<TBS-POLICY-QueryResponse>
   <RequestID>urn:random:zwos43i55098w4tawo3i5j09q==
   <Assertion>
      <AssertionID>http://policy.carol.test/assertion/
      <Issuer>URN:dns-date:policy.carol.test:2001-03-03:1204
      <ValidityInterval>
         <NotBefore>
         <NotOnOrAfter>
      <Claim>
         <Policy>
            <Resources>
               <string>http://store.carol.test/finance
```

```
            <ACL>
               <ACE>
                  <Subject>
                     <Role>URN:dns-date:www.bizexchange.test:2001-01-
04:right:finance
                  <Permit>RWED
               <ACE>
                  <Deny>ED
                  <Subject>
                     <Right>URN:dns-date:www.bizexchange.test:2001-01-
04:right:ops
                  <Permit>R
               <ACE>
                  ...
```

2.5.5 ❺ Request Authorization Assertion

The Policy Decision Point does not wish to disclose the specific resource request to the business exchange. Instead the resource rights identifiers specified in the ACL are specified:

```
<SAMLQuery>
   <RequestID>urn:random:zslkrejt03a856q32n60978q3204967==
   <Query>
      <Binding>
         <Subject>
            <NameID>mailto:Alice@bizex.test
         <Object>
            <Role>URN:dns-date:www.bizexchange.test:2001-01-
04:right:finance
            <Role>URN:dns-date:www.bizexchange.test:2001-01-
04:right:ops
            <Attribute>URN:dns-date:www.bizexchange.test:2001-01-
04:attribute:certified_public_accountant
   <Respond>
      <string>Assertion
```

2.5.6 ❻ Authorization Assertion

The SAML authorization assertion is similar to that in the example of section 2.1 .In this case however the Business Exchange only returns the specific information requested:

```
<SAMLQueryResponse>
   <RequestID>urn:random:zslkrejt03a856q32n60978q3204967==
   <Assertion>
      <AssertionID>http://www.bizexchange.test/assertion/AE0221
      <Issuer>URN:dns-date:www.bizexchange.test:2001-01-03:19283
      <ValidityInterval>
         <NotBefore>
         <NotOnOrAfter>
      <Conditions>
         <Audience>http://www.bizexchange.test/rule_book.html
      <Claims>
         <Subject>
            <NameID>mailto:Alice@bizex.test
            <Authenticator>
               <ds:KeyInfo>
```

```
                    <ds:X509Data>...
          <Object>
             <Authority>
                <Permission>Read
                <Resource>http://store.carol.test/finance
             <Role>URN:dns-date:www.bizexchange.test:2001-01-
04:right:finance
```

2.5.7 ❼ Access Decision

The access decision alone is returned to the client. No validity interval, conditions or resource data was requested.

```
<SAMLQueryResponse>
   <RequestID>urn:random:aks4ewht98a2745hwa498720o9i9==
   <Decision>Permit
```

2.5.8 ❽ Response

The data is returned to the client.

2.6 SAML Aware Client

An SAML aware client can optimize requests by using the information in an assertion to present the correct data in a request. In addition the need to exchange data between the Issuer and Relying servers directly is avoided.

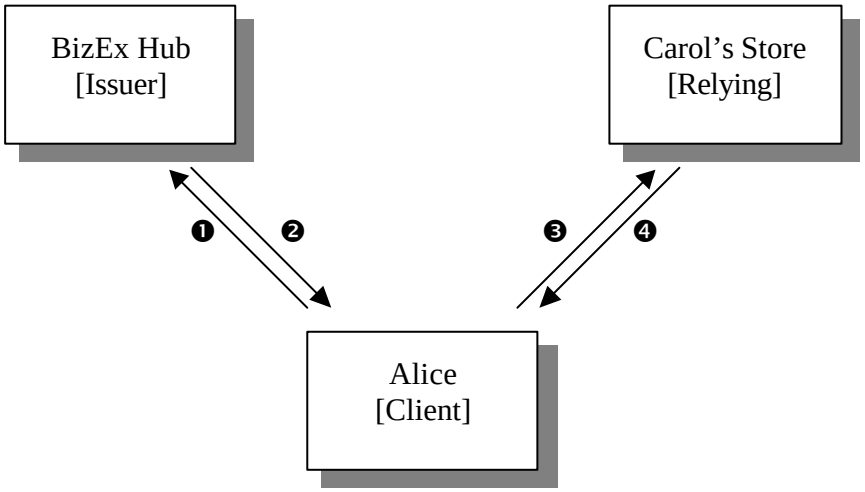


Figure 6: SAML Aware Client

Message	Format	Data
❶ Login	HTTP/SSL Request	Username, Password
❷ Response	HTTP/SSL Response	Assertion

Message	Format	Data
❸ Access	HTTP/SSL Request	Resource_ID, Assertion
❹ Response	HTTP/SSL Response	Data

2.6.1 ❶ Login

Alice authenticates herself to the server using either a password or public key based authentication.

2.6.2 ❷ Response

Bob's Business Exchange returns the assertion to Alice. In this particular configuration the assertion itself is an authentication instrument and presentation of the assertion alone will grant authorization to the "Alice" account:

```
<Assertion>
   <AssertionID>http://www.bizexchange.test/assertion/AE0221
   <Issuer>URN:dns-date:www.bizexchange.test:2001-01-03:19283
   <ValidityInterval>
      <NotBefore>
      <NotOnOrAfter>
   <Conditions>
      <Audience>http://www.bizexchange.test/rule_book.html
   <Claims>
      <Subject>
         <NameID>mailto:Alice@bizex.test
      <Object>
         <Authority>
            <Permission>Read
            <Resource>http://store.carol.test/finance
            <Resource>URN:dns-date:www.bizexchange.test:2001-01-
04:right:finance
```

2.6.3 ❸ Access

Alice presents the assertion to Carol's store:

```
<Assertion>
   <AssertionID>http://www.bizexchange.test/assertion/AE0221
   <Issuer>URN:dns-date:www.bizexchange.test:2001-01-03:19283
   <ValidityInterval>
      <NotBefore>
      <NotOnOrAfter>
   <Conditions>
      <Audience>http://www.bizexchange.test/rule_book.html
   <Claims>
      <Subject>
         <NameID>mailto:Alice@bizex.test
      <Object>
         <Authority>
            <Permission>Read
            <Resource>http://store.carol.test/finance
```

```
            <Resource>URN:dns-date:www.bizexchange.test:2001-01-
04:right:finance
```

2.6.4 ❹ Response

The requested data is returned to Alice.

2.6.5 Using Public Key

One disadvantage of the SAML aware client approach is that the client may not have enough information to determine the applicability of a specific rights identifier. If the assertion itself is the authentication token the consequences of sending the assertion to the wrong location are a severe security failure.

A more robust approach is to use an assertion bound to a public key. The corresponding private key may be a long-term private key held by the assertion subject or may be generated ephemerally and established with the assertion issuer through a password based key exchange scheme.

3 References

[Kerberos] *TBS*

[SAML-USE] *TBS*

[PKCS1] Kaliski, B., *PKCS #1: RSA Encryption Version 2.*0, RSA Laboratories, also IETF RFC 2437, October 1998.

[RFC-2104] Krawczyk, H., Bellare, M. and R. Canetti, *HMAC: Keyed Hashing for Message Authentication*, IETF RFC 2104, February 1997.

[SOAP] D. Box, D Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. Frystyk Nielsen, S Thatte, D. Winer. *Simple Object Access Protocol (SOAP) 1.1*, W3C Note 08 May 2000, http://www.w3.org/TR/SOAP

[WSSL] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, *Web Services Description Language (WSDL) 1.0* September 25, 2000, http://msdn.microsoft.com/xml/general/wsdl.asp

[XACML] *TBS*

[XTASS] P. Hallam-Baker, *XML Trust Assertion Service Specification*, To Be Published, January 2001

[XML-SIG] D. Eastlake, J. R., D. Solo, M. Bartel, J. Boyer , B. Fox , E. Simon. *XML-Signature Syntax and Processing*, World Wide Web Consortium. http://www.w3.org/TR/xmldsig-core/

[XML-SIG-XSD] XML Signature Schema available from http://www.w3.org/TR/2000/CR-xmldsig-core-20001031/xmldsig-core-schema.xsd

[XML-Enc] *XML Encryption Specification*, In development.

[XML-Schema1] H. S. Thompson, D. Beech, M. Maloney, N. Mendelsohn. *XML Schema Part 1: Structures*, W3C Working Draft 22 September 2000, http://www.w3.org/TR/2000/WD-xmlschema-1-20000922/, latest draft at http://www.w3.org/TR/xmlschema-1/

[XML-Schema2] P. V. Biron, A. Malhotra, *XML Schema Part 2: Datatypes*; W3C Working Draft 22 September 2000, http://www.w3.org/TR/2000/WD-xmlschema-2-20000922/, latest draft at http://www.w3.org/TR/xmlschema-2/

4 Acknowledgements

Appendix A Ticket Encoding Syntax

It may be argued that the ticket encoding syntax can be left to private agreement between servers.

- Allow encoding of any SAML assertion data element
- Place no arbitrary restrictions on the lengths of data objects
- Require minimal overhead, allowing a ticket to be encoded in 64 bytes of data
- Identify the version number of the ticket encoding

A.1 Self-terminating Integer Encoding

Fields representing data length and tag values are encoded using a simple self-terminating length encoding. In this encoding values are encoded as a sequence of octets. The most significant bit of the last octet in sequence is set, the most significant bit of each of the leading octets is clear. The value of the integer is encoded in the lower 7 bits of the octet sequence, with the first octet being the least significant.

Examples:

Integer	Data (hexadecimal)			Value
0	80			$= 00_H$
1	81			$= 01_H$
2	82			$= 01_H$
127	FF			$= 7F_H$
128	00	81		$= 00_H + 80_H \cdot 81_H$
16383	7F	FF		$= 7F_H + 80_H \cdot 7F_H$

2097151	7F	7F	FF	$= 7F_H + 80_H \cdot 7F_H + 4000_H \cdot 7F_H$

Data may be encoded using the following procedure:

```
Encode (integer v, out octet s[], out integer i)
  while (v > 127)
    s [i] = octet (v & 127)
    v = v / 128
    i = i + 1
  s [i] = (128 + v)
  i = i + 1
```

Data may be decoded using the following procedure:

```
Decode (octet s[], out integer v, out integer i)
  integer base = 1
  v = 0
  while (s[i] < 128)
    v = v + s[i] * base
    i = i + 1
    base = base * 128
  v = v + (s[i] – 128) * base
  i = i + 1
```

Additional code may be required to perform range checking if the language does not support integers of indefinite size.

A.2 Envelope Format

Field	Length (bytes.bits)	Max	Description
Version	0.4	0.4	Equals 0 for this version
Encryption Suite	0.4	0.4	0 = AES encryption with HMAC-SHA1
Key ID length	1	1	
Key ID	1	20	
Body length	1	2	
Body	22	256+	
Checksum length	1	1	
Checksum	12	20	

Total	39	301	

A.3 Body Data

Body data is encoded as a sequence of Tag, Length Data triplets where the tag values are specified as follows:

Tag Value	Description
0	SHA-1 hash of the assertion
1	Locator for assertion
2	Authenticated account identifier
3	Unauthenticated account identifier
4	Expiry date and time (format TBD)
5	Symmetric keying material
…	To be specified

Both tag and length are encoded using the self-terminating integer encoding

Examples:

```
8094 16E4 C8F6 681D C786 560B 9012 712C 602E 348F 39EE
```

Tag is 0 (SHA-1 hash of the assertion), Length is 20 bytes, and data is C8F6 39EE.

```
8285 "Alice"
```

Tag is 2 (Authenticated account identifier), Length is 5 bytes, and data is `Alice`.