



---

# **Security Assertions Markup Language**

*Straw-man Core Assertion Architecture*

• *Phillip Hallam-Baker*

• *VeriSign*

*Draft Version 0.4: March 29th 2001*

---

# Security Assertions Markup Language

Version 0.4

## Table Of Contents

Table Of Contents	2
Table of Figures	4
<b>Executive Summary</b>	<b>5</b>
<b>1 Introduction</b>	<b>5</b>
1.1 Parties	5
1.1.1 Subject / Principal	Error! Bookmark not defined.
1.1.2 Issue Point	Error! Bookmark not defined.
1.1.3 Policy Enforcement Point	6
1.1.4 Policy Decision Point	6
1.2 Data Objects	6
1.2.1 SAML Assertion	6
1.2.2 Ticket	7
<b>2 Example Messages</b>	<b>8</b>
2.1 Web Browser Password Access	8
2.1.1 ❶ Login	9
2.1.2 ❷ Response	9
2.1.3 ❸ Access	10
2.1.4 ❹ Pull Assertion	10
2.1.5 ❺ Assertion	12
2.1.6 ❻ Resource	12
2.2 SSL Certificate Based Client Authentication	13
2.2.1 ❶ Request	14
2.2.2 ❷ Pull Assertion	14
2.2.3 ❸ Assertion	15
2.2.4 ❹ Resource	15
2.3 Server Authorization Delegation	15
2.3.1 ❶ Request	16
2.3.2 ❷ Request Access Decision	16
2.3.3 ❸ Request Access Policy	17
2.3.4 ❹ Access Policy	17
2.3.5 ❺ Request Authorization Assertion	18
2.3.6 ❻ Authorization Assertion	18
2.3.7 ❼ Access Decision	18
2.3.8 ❸ Response	18
2.4 SAML Aware Client	18
2.4.1 ❶ Login	19
2.4.2 ❷ Response	19

2.4.3	③ Access	20
2.4.4	④ Response	20
2.4.5	Using Public Key	20
<b>3</b>	<b>Architecture</b>	<b>20</b>
3.1	Basic Information	22
3.1.1	Assertion Identifier	Error! Bookmark not defined.
3.1.2	Issuer	Error! Bookmark not defined.
3.1.3	Subject	Error! Bookmark not defined.
3.1.4	Issue Date	24
3.1.5	Validity Interval	25
3.1.6	Assertion Status	Error! Bookmark not defined.
3.2	Conditions	25
3.2.1	Online Verification	26
3.2.2	Audience Restriction	26
3.3	Claim: Authority	27
3.4	Advice	27
<b>4</b>	<b>Assertion Syntax</b>	<b>29</b>
4.1	Top Level Elements	Error! Bookmark not defined.
4.1.1	Element <AssertionQuery>	Error! Bookmark not defined.
4.1.2	Element <Assertion>	Error! Bookmark not defined.
4.2	Framework Elements	Error! Bookmark not defined.
4.2.1	Element <AssertionID>	Error! Bookmark not defined.
4.2.2	Element <RequestID>	Error! Bookmark not defined.
4.2.3	Element <Issuer>	Error! Bookmark not defined.
4.2.4	Element <DateTime>	25
4.2.5	Element <ValidityInterval>	25
4.2.6	Element <Conditions>	Error! Bookmark not defined.
4.2.7	Element <Claim>	Error! Bookmark not defined.
4.2.8	Element <Advice>	29
4.2.9	Element <Respond>	23
4.3	Claim Elements	Error! Bookmark not defined.
4.3.1	Element <Subject>	27
4.3.2	Element <Object>	28
4.3.3	Element <Restrictions>	29
4.3.4	Element <Authority>	27
<b>5</b>	<b>References</b>	<b>29</b>
<b>6</b>	<b>Acknowledgements</b>	<b>30</b>
<b>Appendix A</b>	<b>Ticket Encoding Syntax</b>	<b>30</b>
A.1	Self-terminating Integer Encoding	30
A.2	Envelope Format	31
A.3	Body Data	32

**Table of Figures**

Figure 1: Parties to the protocol	5
Figure 2: Multiple Assertion Relationships	6
Figure 3: Web Server Log In	8
Figure 4: Certificate Based Client Auth	14
Figure 5: Delegated Decision Point	16
Figure 6: SAML Aware Client	19

## 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).

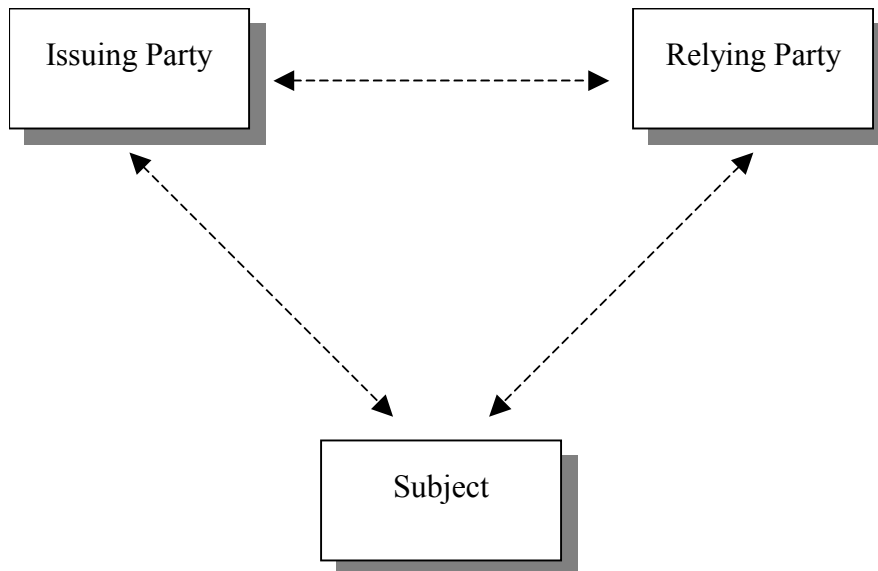


Figure 1: Parties to the protocol

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 access to a particular resource. The Policy Enforcement Point relies upon the latter assertion to grant or deny access to the resource in question.

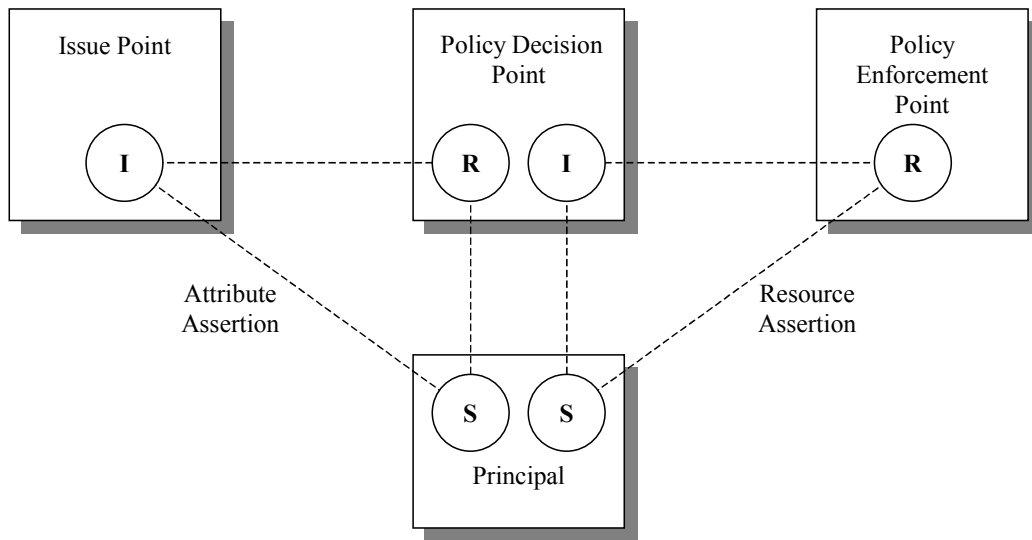


Figure 2: Multiple Assertion Relationships

### 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

A *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

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

## 1.2 Data Objects

### 1.2.1 SAML Assertion

An XML data structure that makes a security assertion. Typical assertions include:

- The party with account ID Alice has the *Plumber* right.

Printed on Wednesday, April 04, 2001

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

Assertions may encode authorization data 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.

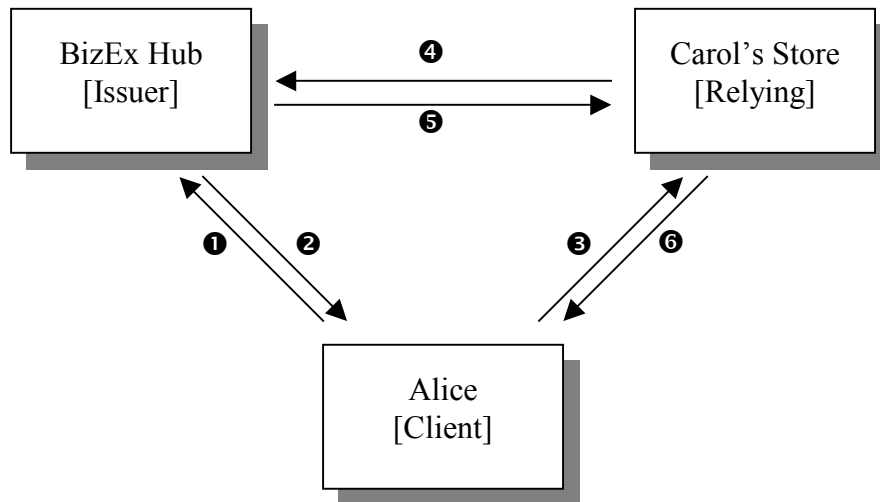


Figure 3: Web Server Log In

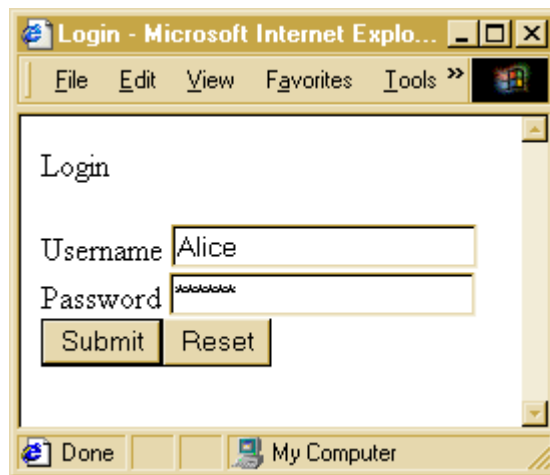
Message	Format	Data
① Login	HTTP/SSL Request	Username, Password
② Response	HTTP/SSL Response, Ticket (as HTML URL)	Ticket = Account, Validity, Assertion_ID Authenticator

Message	Format	Data
③ Access	HTTP/SSL Request	Ticket
④ Pull Assertion	XP Request	Assertion_ID
⑤ Assertion	XP Response	Assertion (see below)
⑥ Resource	HTTP/SSL Response	Resource Data

### 2.1.1 ① 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 ② 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

---

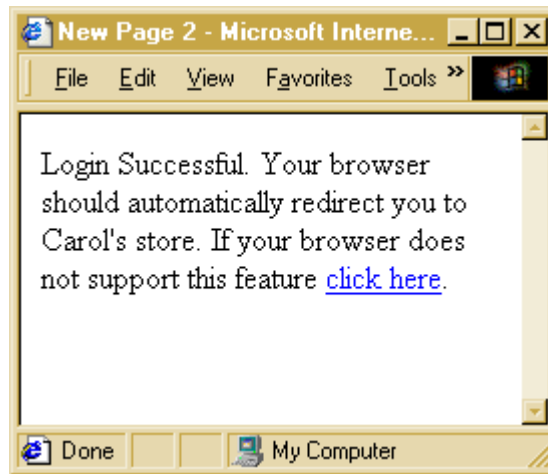
Using base64 encoding this results in a 60 byte string which is passed to Carol encoded as a URL:

```
<% Response.Redirect
"https://store.carol.test/finance/bizex.asp?ticket=jubafOqNEpcwR3RdFsT7b
CqnXPBe5ELh5u4VEy19MzxxXRgrMvavzyBpVR==" %>
<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=jubafOqNEpcwR3Rd
FsT7bCqnXPBe5ELh5u4VEy19MzxxXRgrMvavzyBpVR==">click
here</a>.</p>

</body>
</html>
```



### 2.1.3 ③ 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=jubafOqNEpcwR3RdFsT7bC
qnXPBe5ELh5u4VEy19MzxxXRgrMvavzyBpVR==
```

### 2.1.4 ④ Pull Assertion

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

Printed on Wednesday, April 04, 2001

- 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).

```
<AssertionQuery>  
  <Claim>  
    <Authority>  
      <Resources>  
        <string>http://store.carol.test/finance
```

```
<Subject>
  <Account>Alice
```

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 ⑤ Assertion

The assertion specifies the authorizations attached to the Alice account:

```
<SAML>
  <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
  <Claim>
    <Authority>
      <Subject>
        <Account>Alice
      <Resources>
        <string>http://store.carol.test/finance
        <string>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 ⑥ 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 2.4 .

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 Pull Model**

2.2.1 ❶ Principal to Authority (request)

2.2.2 ❷ Authority to Principal (response)

2.2.3 ❸ Principal to PEP (request)

2.2.4 ❹ PEP to Principal

2.2.5 ❺ PEP to PDP (request)

2.2.6 ❻ PDP to PEP (response)

2.2.7 ❼ PDP to Authority (request)

2.2.8 ❽ Authority to PDP (response)

## **2.3 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.

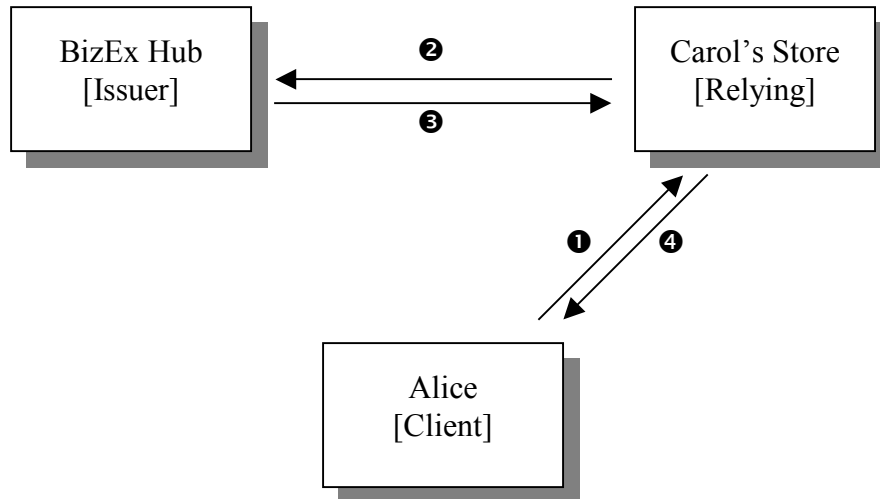


Figure 4: Certificate Based Client Auth

Message	Format	Data
➊ Request	HTTP/SSL Request (with certificate based client authentication)	Certificate, Resource
➋ Pull Assertion	XP Request	Certificate
➌ Assertion	XP Response	Assertion (see below)
➍ Resource	HTTP/SSL Response	Resource Data

### 2.3.1 ➊ 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).

### 2.3.2 ➋ Pull Assertion

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

```

<AssertionQuery>
  <Respond>
    <String>Assertion
  <Claim>
    <Authority>
      <Resources>
        <string>http://store.carol.test/finance
      <Subject>
        <ds:KeyInfo>
          <ds:X509Data>...
    
```

### 2.3.3 ③ Assertion

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

```
<SAML>
  <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
  <Claim>
    <Authority>
      <Subject>
        <ds:KeyInfo>
          <ds:X509Data>...
      <Resources>
        <string>http://store.carol.test/finance
        <string>URN:dns-date:www.bizexchange.test:2001-01-
04:right:finance
```

### 2.3.4 ④ Resource

The resource is returned to Alice.

## 2.4 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.

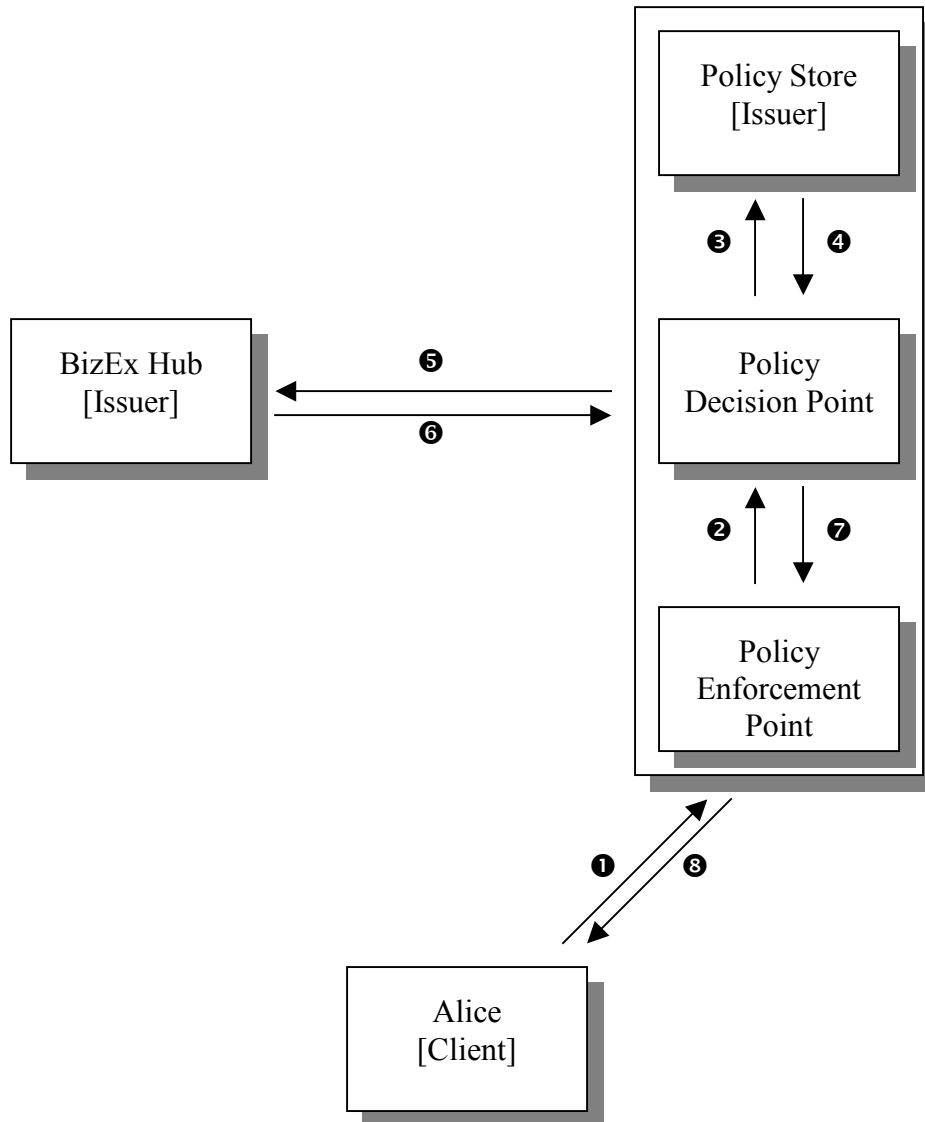


Figure 5: Delegated Decision Point

### 2.4.1 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=jubafOqNEpcwR3RdFsT7bCqnXPBe5ELh5u4VEy19MzxkXRgrMvavzyBpVR==>

### 2.4.2 ❷ Request Access Decision

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

```
<AssertionQuery>  
  <Claim>  
    <Authority>  
    <Resources>
```

```
<string>http://store.carol.test/finance
<Subject>
  <Ticket>jubafOqNEpcwR3RdFsT7bCqnXPBe5ELh5u4VEy19MzxxXRgrMva
  vzyBpVR==
<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.

### 2.4.3 ③ 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:

```
<AssertionQuery>
  <Claim>
    <Policy>
      <Resources>
        <string>http://store.carol.test/finance
```

### 2.4.4 ④ Access Policy

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

```
<TASS-ACL>
  <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>
            <Right>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>
          ...
```

#### 2.4.5 ⑤ 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:

```
<AssertionQuery>
  <Claim>
    <Authority>
      <Resources>
        <string>URN:dns-date:www.bizexchange.test:2001-01-
04:right:finance
        <string>URN:dns-date:www.bizexchange.test:2001-01-
04:right:ops
      <Subject>
        <Account>Alice
```

#### 2.4.6 ⑥ 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:

```
<SAML>
  <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
  <Claim>
    <Authority>
      <Subject>
        <ds:KeyInfo>
          <ds:X509Data>...
        <Resources>
          <string>URN:dns-date:www.bizexchange.test:2001-01-
04:right:finance
```

#### 2.4.7 ⑦ Access Decision

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

```
<SAML>
  <Status>Valid
```

#### 2.4.8 ⑧ Response

The data is returned to the client.

### 2.5 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.

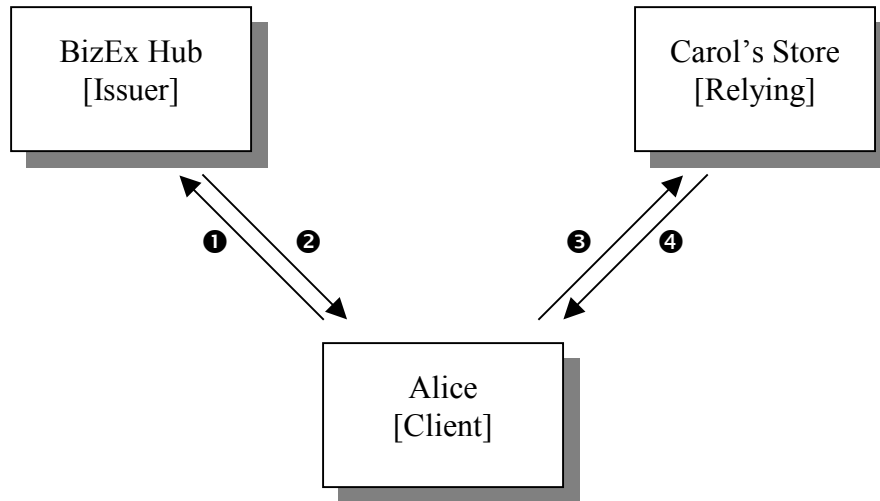


Figure 6: SAML Aware Client

Message	Format	Data
❶ Login	HTTP/SSL Request	Username, Password
❷ Response	HTTP/SSL Response	Assertion
❸ Access	HTTP/SSL Request	Resource_ID, Assertion
❹ Response	HTTP/SSL Response	Data

### 2.5.1 ❶ Login

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

### 2.5.2 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:

```

<SAML>
  <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
  <Claim>
    <Authority>
      <Subject>
        <Account>Alice
  
```

```
<Resources>
  <string>http://store.carol.test/finance
  <string>URN:dns-date:www.bizexchange.test:2001-01-
04:right:finance
  <ds:Signature>...
```

### 2.5.3 Access

Alice presents the assertion to Carol's store:

```
<SAML>
  <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
  <Claim>
    <Authority>
      <Subject>
        <Account>Alice
      <Resources>
        <string>http://store.carol.test/finance
        <string>URN:dns-date:www.bizexchange.test:2001-01-
04:right:finance
    <ds:Signature>...
```

### 2.5.4 Response

The requested data is returned to Alice.

### 2.5.5 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 XML Assertion and Request Syntax

Each SAML protocol exchange consists of a request and response. The embedding of these requests and responses is described in detail in the section on Bindings<sup>[PHB1]</sup>.

The syntax of requests and responses are closely related and so both are described here.



### 3.1 Namespaces

For clarity, some examples of XML are not complete documents and namespace declarations may be omitted from XML fragments. In this document, certain namespace prefixes represent certain namespaces.

All SAML protocol elements are defined using XML schema **[XML-Schema1][XML-Schema2]**. For clarity unqualified elements in schema definitions are in the XML schema namespace:

```
xmlns="http://www.w3.org/2000/10/XMLSchema." [PHB2]
```

References to XML Key Management Specification schema defined herein use the prefix “s0” and are in the namespace:

```
xmlns:s0="http://www.oasis.org/tbs/1066-12-25/" [PHB3]
```

This namespace is also used for unqualified elements in message protocol examples.

The SAML schema specification uses some elements already defined in the XML Signature namespace. The “XML Signature namespace” is represented by the prefix `ds` and is declared as:

```
xmlns:ds="http://www.w3.org/2000/09/xmldsig#" [PHB4]
```

The XML Signature schema” is defined in **[XML-SIG-XSD]** and the `<ds:KeyInfo>` element (and all of its contents) are defined in **[XML-SIG]**§4.4.

### 3.2 SAML Assertion

SAML specifies several different types of assertion for different purposes, these are:

Authentication Assertion

Attribute Assertion

Decision Assertion

The different types of SAML assertion are encoded in a common XML package which at a minimum consists of:

#### Basic Information.

Each assertion **MUST** specify a unique identifier that serves as a name for the assertion. In addition an assertion **MAY** specify the date and time of issue and the time interval for which the assertion is valid.

#### Claims.

The claims made by the assertion. This document describes the use of assertions to make claims for Authorization and Key Delegation applications.

In addition an assertion MAY contain the following additional elements. An SAML client is not required to support processing of any element contained in an additional element **with the sole exception that an SAML client MUST reject any assertion containing a Conditions element that is not supported.**

#### **Conditions.**

The assertion status MAY be subject to conditions. The status of the assertion might be dependent on additional information from a validation service. The assertion may be dependent on other assertions being valid. The assertion may only be valid if the relying party is a member of a particular audience.

#### **Advice.**

Assertions MAY contain additional information as advice. The advice element MAY be used to specify the assertions that were used to make a policy decision.

The SAML assertion package is designed to facilitate reuse in other specifications. For this reason XML elements specific to the management of authentication and authorization data are expressed as claims. Possible additional applications of the assertion package format include management of embedded trust roots [XTASS] and authorization policy [XACML].

The <Assertion> element is specified by the following schema:

```
<element name="Assertion">
  <complexType>
    <sequence>
      <!-- Basic Information -->
      <element name="AssertionID" type="s0:AssertionID"/>
      <element name="RequestID" type="s0:AssertionID"/>
      <element name="Issuer" type="string"/>
      <element name="IssueInstant" type="DateTime"/>
      <element name="ValidityInterval" type="s0:ValidityInterval"/>

      <!-- Data -->
      <element name="Claims" type="s0:Claims"/>
      <element name="Conditions" type="s0:Conditions"/>
      <element name="Advice" type="s0:Advice"/>
    </sequence>
  </complexType>
</element>
```

### **3.3 SAML Request**

#### **3.3.1 Request by Identifier or Location**

An assertion MAY be requested by means of the assertion identifier or a location in which the assertion is stored.

#### **3.3.2 Request by Query**

An assertion MAY be requested by means of a query.

The query specifies the principal and the resources for which access is requested by use of the claim element syntax. The information requested in the response is specified by means of the Respond element described in section 3.3.3.

[Discuss]

The <SAMLQuery> element is defined by the following schema:[PHB5]

```
<element name="SAMLQuery">
  <complexType>
    <sequence>
      <!-- Basic Information -->
      <element name="RequestID" type="s0:AssertionID"/>
      <element name="ValidityInterval" type="s0:ValidityInterval"/>

      <!-- Data -->
      <element name="Query" type="s0:Claims"/>
      <element name="Conditions" type="s0:Conditions"/>
      <element name="Advice" type="s0:Advice"/>

      <element name="Respond" type="s0:Respond"/>
    </sequence>
  </complexType>
</element>
```

### 3.3.3 Element <Respond>

The <Respond> element in the request specifies one or more strings included in the request that specify data elements to be provided in the response.

The Service SHOULD return a requested data element if it is available. The Service MAY return additional data elements that were not requested. In particular, the service MAY return data elements specified in the request with the response.

Defined identifiers include:

Identifier	Description
Decision	Return the result of the Query (True/False).
Static	Specifies that the response may return any data element and omit the RequestID specified in the request. Thus allowing the responder to return a static pre-signed assertion.
ValidityInterval	Return the ValidityInterval element
Conditions	Return the assertion conditions
Claims	Return the assertion claims
Advice	Return additional advice elements

The <Respond> element is defined by the following schema:

```
<element name="Respond" >
  <complexType>
    <sequence>
      <element name="string" type="string"
        minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  </complexType>
</element>
```

### 3.4 SAML Response

[An alternative packaging that might be desirable would be to specify a query response. In this case the natural location for the Request ID would be here]

The <SAMLQueryResponse> element is defined by the following schema:

```
<element name="SAMLQueryResponse">
  <complexType>
    <sequence>
      <!-- Basic Information -->
      <element name="RequestID" type="s0:AssertionID"/>
      <element name="Decision" type="s0:Decision"/>
      <element name="Assertion" type="s0:Assertion"/>
    </sequence>
  </complexType>
</element>
```

### 3.5 Basic Information

Four basic information elements are defined; a unique identifier, the issuer, the time instant of issue, the validity interval and the assertion status.

#### 3.5.1 Element <AssertionID>

Each assertion MUST specify exactly one unique assertion identifier. All identifiers are encoded as a Uniform Resource Identifier (URI) and are specified in full (use of relative identifiers is not permitted).

The URI is used as a *name* for the assertion and not as a *locator*. It is only necessary to ensure that no two assertions share the same identifier. Provision of a service to resolve an identifier into an assertion is not a requirement.

The <AssertionID> element is defined by the following schema:

```
<element name="AssertionID" type="string"/>
```

#### 3.5.2 Element <RequestID>

The RequestID element defines a unique identifier for the assertion request. If an assertion query specifies a RequestID value the same value MUST be returned in the response unless a Respond element of Static is specified.

The <RequestID> element is defined by the following schema:

```
<element name="RequestID" type="string"/>
```

### 3.5.3 Element <Issuer>

The `Issuer` element specifies the issuer of the assertion by means of a URI. It is defined by the following XML schema:

The <Issuer> element is defined by the following schema:

```
<element name="Issue" type="string"/>
```

### 3.5.4 Element <IssueInstant>

The time instant of issue.

The <IssueInstant> element is defined by the following schema:

```
<element name="IssueInstant" type="timeInstant"/>
```

### 3.5.5 Element <ValidityInterval>

The <ValidityInterval> structure specifies limits on the validity of the assertion. It contains the following elements:

Member	Type	Description
<code>NotBefore</code>	<code>DateTime</code>	Time instant at which the validity interval begins
<code>NotAfter</code>	<code>DateTime</code>	Time instant at which the validity interval has ended

The `DateTime` instant **MUST** fully specify the date.

The `NotBefore` and `NotAfter` elements are optional. If the value is either omitted or equal to the start of the epoch it is unspecified. If the `NotBefore` element is unspecified the assertion is valid from the start of the epoch until the `NotAfter` element. If the `NotAfter` element is unspecified the assertion is valid from the `NotBefore` element with no expiry. If neither element is specified the assertion is valid at any time.

In accordance with the XML Schemas Specification, all time instances are interpreted in Universal Coordinated Time unless they explicitly indicate a time zone.

Implementations **MUST NOT** generate time instances that specify leap seconds.

For purposes of comparison, the time interval `NotBefore` to `NotAfter` begins at the earliest time instant compatible with the specification of `NotBefore` and *has ended* at the earliest time instant compatible with the specification of `NotAfter`

For example if the time interval specified is *dayT12:03:02* to *dayT12:05:12* the times 12:03:02.00 and 12:05:11.9999 are within the time interval. The time 12:05:12.0000 is outside the time interval.

The <ValidityInterval> element is defined by the following schema:

```
<complexType name="ValidityInterval">
  <sequence>
    <element name="NotBefore" type="timeInstant"/>
    <element name="NotAfter" type="timeInstant"/>
  </sequence>
</complexType>
```

### 3.6 Conditions

Assertion Conditions are contained in the <Conditions> element. Applications using the framework MAY define additional elements. If an application encounters an element contained within a <Conditions> element that is not understood the status of the Condition MUST be considered Indeterminate.

The following conditions are defined:

Identifier	Type	Description
Audiences	URI []	Specifies the set of audiences to which the assertion is addressed.

The <Conditions> element is defined by the following XML schema:

```
<element name="Conditions">
  <complexType>
    <sequence>
      <element name="Audiences" >
        <complexType >
          <sequence>
            <element name="string" type="string"
              minOccurs="0" maxOccurs="unbounded"/>
          </sequence>
        </complexType>
      </element>
    </sequence>
  </complexType>
</element>
```

#### 3.6.1 Element <Audiences>

Assertions MAY be addressed to a specific audience. Although a party that is outside the audience specified is capable of drawing conclusions from an assertion, the issuer explicitly makes no representation as to accuracy or trustworthiness to such a party.

- Require users of an assertion to agree to specific terms (rule book, liability caps, relying party agreement)
- Prevent clients inadvertently relying on data that does not provide a sufficient warranty for a particular purpose

Printed on Wednesday, April 04, 2001

- Enable sale of per-transaction insurance services.

An audience is identified by a URI that identifies to a document that describes the terms and conditions of audience membership.

Each client is configured with a set of URIs that identify the audiences that the client is a member of, for example:

```
http://cp.verisign.test/cps-2000
Client accepts the VeriSign Certification Practices Statement
```

```
http://rule.bizexchange.test/bizexchange_ruebook
Client accepts the provisions of the bizexchange rule book.
```

An assertion MAY specify a set of audiences to which the assertion is addressed. If the set of audiences is the empty set there is no restriction and all audiences are addressed. Otherwise the client is not entitled to rely on the assertion unless it is addressed to one or more of the audiences that the client is a member of. For example:

```
http://cp.verisign.test/cps-2000/part1
Assertion is addressed to clients that accept the provisions of a specific part of the
VeriSign CPS.
```

In this case the client accepts a superset of the audiences to which the assertion is addressed and may rely on the assertion.

The <Audiences> element is defined by the following XML schema:

```
<element name="Audiences" >
  <complexType >
    <sequence>
      <element name="string" type="string"
        minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  </complexType>
</element>
```

### 3.7 Claims

[Discuss]

The <Claims> element is defined by the following XML schema:

```
<element name="Claims">
  <complexType>
    <sequence>
      <element name="Authority" type="so:Authority"
        minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  </complexType>
</element>
```

#### 3.7.1 Element <Authority>

Access control requires means to both authenticate a party and determine the actions that the party is authorized to perform. In an enterprise environment it is convenient to centralize administration of authorization data. An individual who requires access to one finance application is likely to require access to others. If authorization to access one resource is terminated it is likely that access to other resources should be revised (or at least reviewed) as well.

Although authentication data and authorization data are typically managed separately, it is convenient to combine delivery of authorization data with delivery of authentication data.

Authorization data is bound to a cryptographic key by means of a URI that corresponds to a set of resources for which access is granted.

The correspondence between a URI and a set of resources are established by agreement between the Assertion Service and the client. In the case that the resource is accessed by a URL it is not necessary that there be a direct or systematic relationship between the URL corresponding to the resource and the URI corresponding to the authorization to access the resource.

The <Authority> element is defined by the following XML schema:

```
<element name="Authority">
  <complexType>
    <sequence>
      <!-- Basic Information -->
      <element name="Subject" type="s0:Subject"/>
      <element name="Object" type="s0:Object"/>
      <element name="Restrictions" type="s0:Restrictions"/>
    </sequence>
  </complexType>
</element>
```

### 3.7.2 Element <Subject>

The <Subject> element is defined by the following XML schema:

```
<element name="Subject">
  <complexType>
    <sequence>
      <element name="Account" type="string"/>
      <element name="Role" type="string"/>
      <element name="KeyInfo" type="ds:KeyInfo"/>
    </sequence>
  </complexType>
</element>
```

### 3.7.3 Element <Object>

The <Object> element is defined by the following XML schema:

```
<element name="Object">
  <complexType>
    <sequence>
      <element name="Resource" type="string"
        minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  </complexType>
</element>
```

```
</complexType>  
</element>
```

### 3.7.4 Element <Restrictions>

[There have been proposals to allow expression of authorizations such as ‘Alice can access this resource on Monday through Friday when there is no Q in the month’. This is a bit on the policy side for my taste, however if this was to be specified this is where the information would logically fit.]

The <Restrictions> element is defined by the following XML schema:

```
<element name="Restrictions">  
  ... Define if we decide we need to ...  
</element>
```

## 3.8 Advice

The Advice element is a general container for any additional information that does not affect the semantics or validity of the assertion itself.

### 3.8.1 Element <Advice>

The <Advice> element permits evidence supporting the assertion claims to be cited, either directly (through incorporating the claims) or indirectly (by reference to the supporting assertions).

The <Advice> element is defined by the following XML schema:

```
<element name="Advice">  
  <complexType>  
    <sequence>  
      <element name="Assertion" type="Assertion"  
        minOccurs="0" maxOccurs="unbounded"/>  
    </sequence>  
  </complexType>  
</element>
```

[An alternative use for the Advice element that is exploited in XTASS 1.0a is for specifying reissue information. This is not employed in SAML but is the reason for the change of name since the last version in case people were wondering.]

## 4 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>

[WSSSL] 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/xmlsig-core/>

[XML-SIG-XSD] XML Signature Schema available from  
<http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-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/>

## 5 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 <sub>H</sub>
1	81			= 01 <sub>H</sub>
2	82			= 01 <sub>H</sub>
127	FF			= 7F <sub>H</sub>
128	00	81		= 00 <sub>H</sub> + 80 <sub>H</sub> · 81 <sub>H</sub>
16383	7F	FF		= 7F <sub>H</sub> + 80 <sub>H</sub> · 7F <sub>H</sub>
2097151	7F	7F	FF	= 7F <sub>H</sub> + 80 <sub>H</sub> · 7F <sub>H</sub> + 4000 <sub>H</sub> · 7F <sub>H</sub>

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	0.4	0.4	0 = AES encryption with HMAC-SHA1

Suite			
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

Printed on Wednesday, April 04, 2001

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.

Page: 20

[PHB1] Which may possibly get moved to a different document in which case adjust text accordingly.

Page: 21

[PHB2] I took this URL from XKMS but the schema group may have progressed since, they sometimes do.

Page: 21

[PHB3] We have to align with the OASIS convention here.

Page: 21

[PHB4] Before everyone points it out, yes the final hash mark is required.

Page: 23

[PHB5] Here we may specify additional query types as is considered desirable by the group. My contention is that the schema will not be significantly different however.