Send comments to:
Phillip Hallam-Baker, Senior Author
401 Edgewater Place, Suite 280
Wakefield MA 01880
Tel 781 245 6996 x227
Email: pbaker@verisign.com

X-TASS: XML Trust Assertion Service Specification

Phillip Hallam-Baker *VeriSign*

Draft Version 0.9: January 5th 2001

Printed on Friday, January 05, 2001

X-TASS: XML Trust Assertion Service Specification

Version 0.9

Table Of Contents

Table of Figures

Executive Summary

This specification defines an architecture and retrieval protocol for *Trust Assertions*. A Trust Assertion consists of a statement bound to a unique identifier that is cryptographically authenticated. Trust Assertions may be used to establish and manage long term trust relations between principals.

As examples the distribution of authorization data and the management of trust roots are considered.

1 Introduction

This document describes mechanisms that support management of long-term trust relationships between parties and binding of additional attributes to a public key.

This specification is intended to complement other XML security standards and proposals, in particular XML Signature [XML-SIG], XML Encryption [XML-ENC] and XML Key Management [XKMS].

1.1 Introduction to this Document

The trust assertion architecture is designed to be extensible to support management of any form trust assertion. In particular assertions need not be bound to a public key infrastructure. A Trust Assertion may be bound directly to a document that represents or facilitates a financial transaction, for example bonds, equities and bills of lading.

XTASS provides a generic framework for specifying information relevant to any form of trust assertion:

- The identifier of the Issuer.
- The time instant of issue.
- Reissue location and scheduling.
- Assertions may be addressed to a specific audience.
- Relying parties may be required to verify the status of an assertion before each use.

Another example use extends the `KeyBinding` element defined in XKMS to allow additional attributes to be bound to a public key. These attributes permit:

- Simultaneous distribution of both authentication data and authorization data bound to a cryptographic key.
- Limited delegation to support separate management of online and offline trust roots.

One example use of XTASS is to enable management of Trust Roots that may be embedded in client devices, in particular trust roots that are used to establish trust between a client and a Trust Service.

XTASS provides a general mechanism for reporting assertion status. All assertions carry a unique identifier specified by means of a URI. Meta-Assertions may be issued that make specific claims about the status of a single assertion or a group of assertions.

1.2 Structure of this document

The remainder of this document describes the Trust Assertion Service Specification.

Section 2: Architecture
The Trust Assertion Architecture are described

Section 3: Message Set.
The semantics of the protocol messages is defined.

2 Architecture

A Trust Assertion is an XML element that contains a unique URI identifier and makes a statement concerning:

- A signed document
- Authorization
- Other trust assertions
- (optionally) A public key

A Trust Assertion is authenticated. A Trust Assertion that is authenticated by means of an XML Signature may be archived in a repository, providing evidence to support non-repudiation.

The specification is divided into two tiers:

Tier 1 Direct Assertions
Bind extended attributes to a public key.

Tier 2 Meta-Assertions
Contain a statement concerning the validity of another statement

Clients may implement only specification level that meets their needs.

The XML elements that a Trust Assertion may contain are divided into five categories as follows.

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.

Assertion 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.

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

Reissue.
In cases where a validity interval is specified information MAY be provided to allow a replacement assertion to be obtained.

Evidence.
Assertions MAY specify the evidence used to make an assertion. For example an assertion might include the Certificate and Certificate Revocation List used to determine the validity of a Public Key binding stated in the claim.

Dividing the elements that Trust Assertions may contain ensures that applications behave correctly when processing elements that are not implemented:

Basic Information.
All applications MUST be capable of correctly processing all basic information elements.

Conditions.
If an application is not able to process an assertion condition the application MUST assign the status `Indeterminate` to that element.

Claims.
All claims are asserted jointly and severally. If an assertion makes more than one claim additional claims MAY be ignored.

Reissue.
Reissue elements MAY be ignored.

Evidence.
Evidence MAY always be ignored.

2.1 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.

2.1.1 Unique Identifier

Each assertion contains exactly one unique 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 rules for lexical comparison and equivalence of URIs are specified in Appendix A .

2.1.2 Issuer

The name of the issuer of the assertion.

2.1.3 Issue Date

The date and time at which the assertion was issued.

2.1.4 Validity Interval

The validity interval MAY specify the earliest instant at which the claims are asserted and/or the earliest instant at which the assertion claims are no longer asserted.

2.1.5 Assertion Status

The status of an assertion may be Valid, Invalid or Indeterminate. If the status is not explicitly specified it is asserted to be Valid.

2.2 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.

2.2.1 Online Verification

In certain circumstances it is desirable to make a claim that is conditional on obtaining online verification at the time of use.

In many business applications where layered services (e.g. insurance) are tied to an assertion it is not the information itself that is acted on but the acceptance of responsibility. In other applications the status of an assertion might be exceptionally volatile requiring verification each time it is used. Use of a template assertion means that the statement is bound to a single assertion id rather than one for each query response.

For example the following assertion specifies that its status MUST be verified by reference to the specified service to be considered trustworthy:

```
<KeyAssertion>
   <AssertionID>urn:taxi:assert.verisign.test/2000-10-
11/x4</AssertionID>
   <Status>Indeterminate</Status>
   <Conditions>
      <Verify>
         <string>http://reissue1.verisign.test/2000-10-11/x4</string>
      </Verify>
   <Conditions>
   <…>…<…/>
</KeyAssertion>
```

The actual status of the assertion is determined by means of the Tier4 status declaration service.

2.2.2 Audience Restriction

TASS 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
- 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.

2.3 Claim: Authorization

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.

In the TASS architecture 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.

Example: Authorizing Web Site Access

An enterprise maintains multiple web services, access to which is authenticated by means of a PKI based authentication protocol (e.g. SSL or equivalent). Authorization data for all the services is maintained in a single database (in a larger enterprise multiple databases might be separately administered).

When Alice attempts to access the Payroll Service, the Payroll Service queries the Assertion Service to verify the validity of Alice's authentication data (her public key) and whether she has authorization (Figure 1).

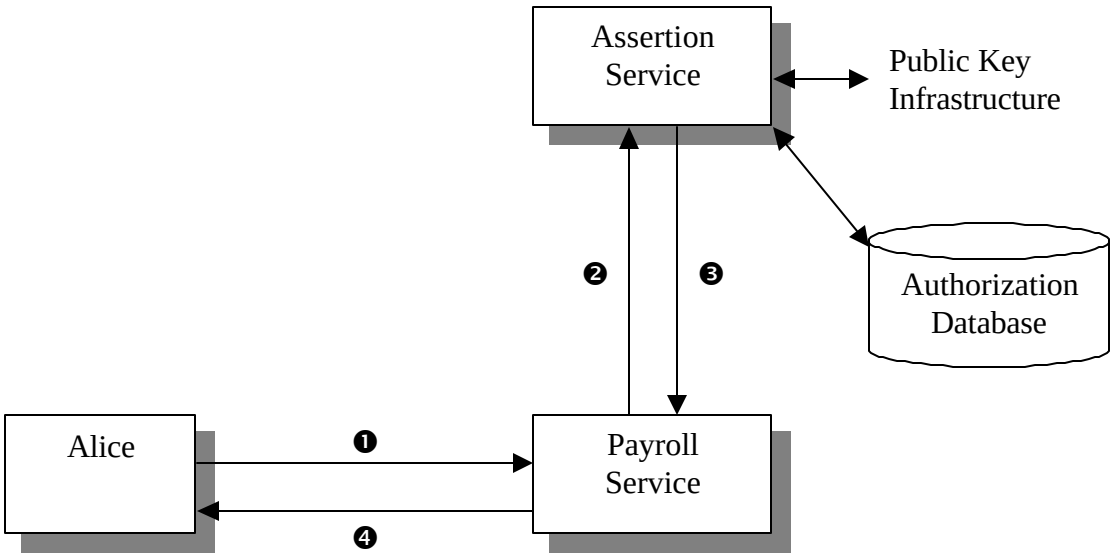

Figure 1: Assertion Service Distributing Authentication and Authorization Data

❶ Alice requests access to the quotes pages of the Payroll Service, the request is authenticated with public key *K*.

The URL of the quotes pages is `http://www.info.test/quotes`

❷ The Payroll Service asks the assertion service whether a request authenticated with public key *K* is authorized to access the resource `urn:names.info.test/2000-09-20/roles/Pay`/1

❸ The Payroll Service responds with a set of resources that public key *K* is authorized to access. The response is constructed using information from a locally maintained authorization database and a remote Public Key Infrastructure.

❹ The request is authorized and the Payroll Service returns the quote data requested.

The Request made by the Payroll Service is:

```
<AssertKey>
   <KeyAssertion>
      <AssertionID>urn:taxi:assert.verisign.test/2000-10-
11/x2</AssertionID>
      <Status>Indeterminate</Status>
      <KeyInfo>
         <KeyValue>
            <RSAKeyValue>
               <Modulus>998/T2PUN8HQlnhf9YIKdMHHGM7HkJwA56UD0a1oY
               q7EfdxSXAidruAszNqBoOqfarJIsfcVKLob1hGnQ/l6xw
               </Modulus>
               <Exponent>AQAB</Exponent>
```

```
            </RSAKeyValue>
         </KeyValue>
         <KeyName>mailto:Alice@cryptographer.test</KeyName>
      </KeyInfo>
      <Resources>
         <string>urn:names.info.test/2000-09-20/roles/Pay/1</string>
      </Resources>
   </KeyAssertion>
</AssertKey>
```

The Response from the Assertion Service is:

```
<AssertKeyResult>
   <KeyAssertion>
      <AssertionID>urn:taxi:assert.verisign.test/2000-10-
11/x2</AssertionID>
      <Status>Indeterminate</Status>
      <KeyInfo>
         <KeyValue>
            <RSAKeyValue>
               <Modulus>998/T2PUN8HQlnhf9YIKdMHHGM7HkJwA56UD0a1oY
               q7EfdxSXAidruAszNqBoOqfarJIsfcVKLob1hGnQ/l6xw
               </Modulus>
               <Exponent>AQAB</Exponent>
            </RSAKeyValue>
         </KeyValue>
         <KeyName>mailto:Alice@cryptographer.test</KeyName>
      </KeyInfo>
      <Resources>
         <string>urn:names.info.test/2000-09-20/roles/ceo</string>
         <string>urn:names.info.test/2000-09-20/roles/Pay</string>
      </Resources>
   </KeyAssertion>
</AssertKeyResult>
```

2.4 Claim: Key Delegation

TASS supports a limited delegation mechanism to support online/offline key management. Delegation is ‘all or nothing’, that is the key to which authority is delegated is directly equivalent to the signing key for the period of the assertion. Constrained delegation where the delegated key has signing authority within a limited name space, corresponding to mutual key recognition agreements between peers (i.e. cross certification) is NOT SUPPORTED.

If a KeyAssertion element contains one or more `Delegation` elements the assertion states that the public key identified by the `KeyInfo` statement (the Delegate key) is directly equivalent to public key used to sign the `KeyAssertion`.

A Delegation Assertion MUST be signed with a Digital Signature. The `KeyInfo` element contained within the `KeyAssertion` must specify the `KeyValue` of the key to which signing authority is delegated.

A Delegation Assertion MAY limit the authority delegated to the delegate key as follows

- The authority to authorize further delegations MAY be restricted by means of the ChainLength parameter. A delegate key MUST NOT sign delegation assertions that have a chain length greater to or equal to the chain length specified. If the chain length is 0 or less the delegate key MUST NOT sign delegation assertions.

Example: Root Key Management

A device contains an embedded trust root that is used to authenticate exchanges with a Tier 2 Trust Service. The risk of compromise of the embedded root is minimized by means of a two stage Delegation scheme as follows:

Offline Root: The private key corresponding to the embedded public key is maintained 'offline' - in secure hardware that is kept in a secure vault under stringent security precautions. The embedded root is never activated in an online system connected to a network of any kind. The offline root is only used to authenticate Delegation Assertions for online roots.

Online Root: The private key corresponding to the online root is also maintained in secure hardware. The online root is used to authenticate protocol exchanges with the Tier 2 Trust Service.

The Key Assertion authenticating the Offline Root is as follows:

```
<KeyAssertion>
   <AssertionID>urn:taxi:assert.verisign.test/2000-10-
11/x5</AssertionID>
   <Status>Valid</Status>
   <KeyBinding>
      <KeyInfo>
         <KeyValue>
            <RSAKeyValue>
               <Modulus>998/T2PUN8HQlnhf9YIKdMHHGM7HkJwA56UD0a1oY
               q7EfdxSXAidruAszNqBoOqfarJIsfcVKLob1hGnQ/l6xw
               </Modulus>
               <Exponent>AQAB</Exponent>
            </RSAKeyValue>
         </KeyValue>
         <KeyName>mailto:Alice@cryptographer.test</KeyName>
      </KeyInfo>
   </KeyBinding>
   <Delegate>
      <ChainLength>1</ChainLength>
   </Delegate>
</KeyAssertion>
```

Note that the Offline Root does not specify a `ValidityInterval`. Although there is a risk that the Offline Root might be compromised there is no means of mitigating the consequences of the compromise or otherwise recovering from them except by out of band means.

The Key Assertion authenticating the Online Root is as follows:

```
<KeyAssertion>
```

```
   <AssertionID>urn:taxi:assert.verisign.test/2000-10-
11/x5</AssertionID>
   <Status>Valid</Status>
   <KeyBinding>
      <KeyInfo>
         <KeyValue>
            <RSAKeyValue>
               <Modulus>998/T2PUN8HQlnhf9YIKdMHHGM7HkJwA56UD0a1oY
               q7EfdxSXAidruAszNqBoOqfarJIsfcVKLob1hGnQ/l6xw
               </Modulus>
               <Exponent>AQAB</Exponent>
            </RSAKeyValue>
         </KeyValue>
         <KeyName>mailto:Alice@cryptographer.test</KeyName>
      </KeyInfo>
   </KeyBinding>
   <Delegate>
      <ChainLength>0</ChainLength>
   </Delegate>
</KeyAssertion>
```

2.5 Claim: Assertion Status

The fourth tier of the XKMS/X-TASS architecture defines assertions that make assertions about assertions. Such Meta-Assertions provide equivalent functionality to the Certificate Revocation List and Online Certificate Status Protocols of PKIX.

TASS supports two types of status query:

- Querying the status of a statement made by an assertion
- Querying the status of the assertion itself

The distinction between the two types of query is not always material.

When liability insurance or other layered services are bound to an assertion there is however a considerable difference between repeating the question and asking if a previous answer is still valid.

A second application of meta-assertions is distribution of assertion status information between distribution points. Figure 1 above shows an architecture in which Trust services acting as local trust distribution points communicate with a Registration Service via the Tier 4 protocol but support only tier 1 & 2 queries from end users.

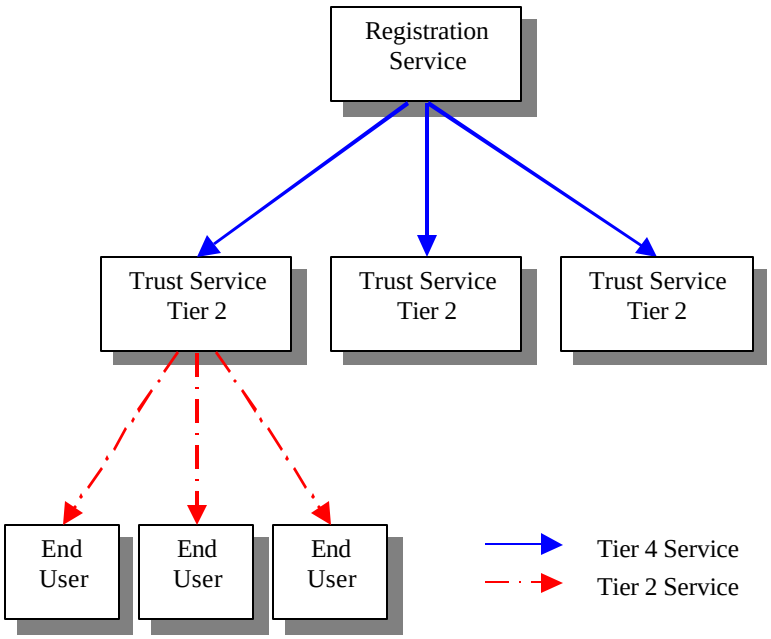

Figure 2: Distribution of Trust Data Between Distribution Points

A third application of meta-assertions is to alert systems that have previously obtained an assertion of a change in the status of an assertion they might otherwise continue to rely on.

The `Declare` Assertion is used to specify the status of a set of TASS assertions. The Declare assertion is designed to permit the status of individual assertions or collections of assertions to be specified.

The Declare assertion is designed to support both dynamic and static signing of responses. Dynamic responses are signed in response to specific requests. Static responses are signed in advance of any request and MAY anticipate multiple requests.

```
<DeclareAssertion>
   <AssertionID>urn:taxi:status.verisign.test/2000-11-
1/20481292</AssertionID>
   <Declare>
      <Status>Valid</Status>
      <First>urn:taxi:assert.verisign.test/2000-10-09/</First>
      <Last>urn:taxi:assert.verisign.test/2000-10-16/</Last>
      <Terminal>False</Terminal>
   </Declare>
   <Declare>
      <Status>Invalid</Status>
      <First>urn:taxi:assert.verisign.test/2000-10-11/x3</First>
      <Last>urn:taxi:assert.verisign.test/2000-10-11/x3</Last>
      <Terminal>True</Terminal>
   </Declare>
</DeclareAssertion>
```

2.6 Other Claims

X-TASS is conceived as a general trust assertion framework. The basic building blocks provided by the X-TASS framework may be applied in a variety of e-commerce applications, for example:

2.6.1 Invoices and Receipts,

An assertion specifying that a particular amount is due or was paid on a particular date.

A standard format for presentation of invoices would permit many aspects of bills payable processing to be automated. This would allow invoice assertions transmitted via email to be automatically routed to accounts processing software (e.g. SAP, Quicken Microsoft Money), allowing 'single click' authorization by the payee. A standard format for receipts would permit automated reconciliation of accounts.

The assertion claims would require specification of the parties (names, account details, etc. etc.), the sum involved (currency, payment terms, due date, etc. etc.), description of the goods concerned (invoice number) and information to permit payment to be effected.

2.6.2 Financial Instrument

In general any financial instrument may be reduced to an assertion that may be represented in the X-TASS architecture. Transfer of the instrument to another party may be achieved through use of Tier 4 status assertions, the status of the old ownership assertion is reported as `Invalid` and a new ownership assertion issued.

2.6.3 Bill of Lading

A bill of lading is a negotiable document that carries title to a cargo carried in transit. A shipper has fully discharged its duty when the cargo is transferred to the first person that presents a valid bill of lading at the port. In many cases a cargo is sold while in transit.

Bills of Lading are physical documents and the bearer of the document has title to the cargo. Replacement of the paper bill of lading with an electronic document offers a significant reduction in processing costs since there is no need to transport the bill of lading by courier in advance of the cargo. In addition replacement of paper title with electronic form facilitates sale and resale of the cargo in transit.

2.6.4 Letter of Credit

A letter of credit is a financial instrument that specifies that a certain sum shall be transferred to a beneficiary on the presentation of specific documentary evidence. The letter of credit is a unique form of financial instrument since the issuer is explicitly required to transfer the sum once the requirements are met regardless of the circumstances. The letter of credit is thus a particularly attractive model for e-commerce since the issuer is required to act in the manner of a machine.

2.6.5 Peer to Peer Trust

The key delegation mechanism specified in the X-TASS core is by design minimal. The only delegates all the rights of the master key to the child with the sole restrictions supported being on further delegation.

The X-TASS framework could be used to specify a full feature peer-to-peer delegation mechanism in which each peer specifies precisely the functions for which a peer is recognized to act. Such a mechanism would inevitably be considerably more complex than the simple mechanism specified in this document.

2.7 Assertion Reissue

Assertions MAY specify a validity interval. Such an assertion MAY typically be reissued before it expires. TASS provides a mechanism whereby clients are informed of the location(s) from which the reissued assertion SHOULD be available if it is reissued and the earliest and latest times at which the assertion SHOULD be obtained.

For example the following assertion will be reissued from two locations on 11th September 2001 and will be available for a week:

```
<KeyAssertion>
   <AssertionID>urn:taxi:assert.verisign.test/2000-10-
11/x2</AssertionID>
   <Reissue>
      <Earliest>2001-9-11</Earliest>
      <Latest>2001-9-18</Latest>
      <Location>
         <string>http://reissue1.verisign.test/2000-10-11/x2<string>
         <string>http://reissue2.verisign.test/2000-10-11/x2<string>
      </Location>
   </Reissue>
   <…>…<…/>
</KeyAssertion>
```

Clients SHOULD attempt to avoid overloading reissue servers by scheduling the download of the reissued assertion for the earliest moment it is available. This MAY be achieved by scheduling the download at a randomly chosen instant between the earliest and latest time instant specified.

2.8 Evidence

Evidence for an assertion MAY be provided contained in the `<Evidence>` element. Applications using the framework MAY define evidence elements. If an application encounters an element contained within a `<Evidence>` element that is not understood it MUST be ignored.

3 Message Set

All protocol exchanges are a remote procedure call that consists of a single request message sent by the client to the service followed by a single response message sent by the service to the client.

The presentation of the message set within the protocol is omitted from this document. We anticipate the use of object exchange layer such as the SOAP/WDSL layer used in XKMS or the XML Protocol specification in progress at the World Wide Web Consortium.

3.1 Framework Elements

The following data elements are used in the message set:

3.1.1 `AssertionID`

The `AssertionID` element defines a unique identifier for the assertion.

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

3.1.2 `Issuer`

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

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

3.1.3 `DateTime`

The time instant of issue. The element is defined by the following XML schema:

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

3.1.4 `ValidityInterval`

The `ValidityInterval` structure specifies limits on the validity of the assertion.

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

Member	Type	Description
`NotBefore`	`DateTime`	Time instant at which the validity interval begins

NotAfter	DateTime	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 *day*T12:03:02 to *day*T12:05:12 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.

3.1.5 Conditions

The Conditions element specifies restrictions on the validity of the assertion and 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>
         <element name="Verify" >
           <complexType >
              <sequence>
                  <element name="string" type="string"
                           minOccurs="0" maxOccurs="unbounded"/>
               </sequence>
            </complexType>
         </element>
     </sequence>
  </complexType>
</element>
```

Where the sub-elements have the following meaning

Identifier	Type	Description
`Audiences`	`URI []`	Specifies the set of audiences to which the assertion is addressed.
`Verify`	`URI []`	If non empty the assertion is a template assertion and the validity of the assertion MUST be specified as Invalid. The set of URIs specifies locations from which the actual status of the assertion is available.

3.1.6 `Evidence`

The `Evidence` element permits evidence supporting the assertion claims to be cited, either directly (through incorporating the claims) or indirectly (by reference to the supporting assertions. Currently no evidence elements are defined.

```
<element name="Evidence">
   <complexType>
      <sequence>

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

3.2 KeyAssertion

3.2.1 `Delegation`

The `Delegation` element specifies that the public key specified in the binding . The element is defined by the following XML schema:

```
<element name="Delegate">
   <complexType>
      <sequence>
         <element name="ChainLength" type="Integer"/>
     </sequence>
  </complexType>
</element>
```

Where the sub-elements have the following meaning

Identifier	Type	Description
`ChainLength`	`Integer`	Maximum number of delegations from the assertion. If 0 further delegation is not permitted.

The delegation mechanism is discussed in detail in section 2.3 above.

3.2.2 Authorization

The Authorization element specifies rights identifiers (encoded as URIs) for a particular Key Binding.

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

Where the sub-elements have the following meaning

Identifier	Type	Description
ChainLength	Integer	Maximum number of delegations from the assertion. If 0 further delegation is not permitted.

The delegation mechanism is discussed in detail in section 2.3 above.

3.2.3 KeyAssertion

The KeyAssertion element is an extended form of the XKMS `KeyBinding` element. In addition to the elements defined by the XKMS specification a `KeyAssertion` MUST carry an assertion identifier that MUST satisfy the uniqueness property. Delegations, Resources and Conditions element MAY be specified. The interpretation of these elements is described in Section 2 above.

```
<element name="KeyAssertion">
   <complexType>
      <sequence>
         <!-- Basic Information -->
         <element name="AssertionID" type="s0:This"/>
         <element name="Issuer" type="string"/>
         <element name="DateTime" type="timeInstant"/>
         <element name="ValidityInterval" type="s1:ValidityInterval"/>
         <element name="Status" type="s1:Status"/>

         <!-- Conditions -->
         <element name="Conditions" type="s0:Conditions"/>

         <!-- Claims -->
         <element name="KeyBinding" type="s1:KeyBinding">
         <element name="Authorize" type="s0:Authorize"
                         minOccurs="0" maxOccurs="unbounded"/>
         <element name="Delegate" type="s0:Delegate"
                         minOccurs="0" maxOccurs="unbounded"/>

         <!-- Reissue -->
         <element name="Reissue" type="s0:Reissue"/>
```

```
         <!-- Evidence -->
         <element name="Evidence"/>
      </sequence>
  </complexType>
</element>
```

3.3 DeclareAssertion

The DeclareAssertion permits a Trust service to communicate changes in the status of a previously issued assertion. The primary application of the Tier 4 protocol is to support communication of trust between a network of servers providing trust services.

Unlike the Services for tiers 1 through 3 the Tier 4 service is designed to permit the service to respond using static data that has been previously signed with a digital signature.

The response to a Declaration request MAY contain more than one declaration, each of which MAY specify the status of more than one assertion. To establish the validity of a particular assertion the client searches the set of declarations to find a match.

3.3.1 Declare

The Declare element asserts the validity of an assertion identified by a URI as follows:

- A Declare structure *matches* a URI if and only if the URI is greater or equal to the attribute First and less than or equal to the value Last.
- A Declare structure is *terminal* if the value of the Terminal attribute is true.
- The validity asserted by a Declare structure is Valid if the value of the Valid attribute is True and Invalid otherwise.

```
<simpleType name="Bool" base="string">
   <enumeration value="True"/>
   <enumeration value="False"/>
</simpleType>

<element name="Declare">
   <complexType>
      <sequence>
         <element name="Status" type="s0:Status"/>
         <element name="First" type="String"/>
         <element name="Last" type="String"/>
         <element name="Terminal" type="s0:Bool"/>
      </sequence>
  </complexType>
</element>
```

3.3.2 DeclareAssertion

The DeclareAssertion structure specifies the common assertion elements and an array of Declare structures.

The order in which elements are entered in the array of Declare structures is significant:

- The validity of the assertion is determined by the attribute `valid` of the matching terminal declare element that meets those criteria that has the lowest index value, if present.
- Otherwise the validity of the assertion is determined by the attribute `valid` of the matching declare element that has the highest index value, if present.
- Otherwise the validity of the assertion is `Undefined`.

```
<element name="DeclareAssertion">
   <complexType>
      <sequence>
         <!-- Basic Information -->
         <element name=“AssertionID” type="s0:This"/>
         <element name="Issuer" type="string"/>
         <element name="DateTime" type="timeInstant"/>
         <element name="ValidityInterval" type="s1:ValidityInterval"/>
         <element name="Status" type="s1:Status"/>

         <!-- Conditions -->
         <element name="Conditions" type="s0:Conditions"/>

         <!-- Claims -->
         <element name="" type="s0:Declare"
                         minOccurs="0" maxOccurs="unbounded"/>/>

         <!-- Reissue -->
         <element name="Reissue" type="s0:Reissue"/>

         <!-- Evidence -->
         <element name="Evidence"/>
     </sequence>
  </complexType>
</element>
```

4 Acknowlegements

The author would like to acknowledge the following people who provided assistance in this work: Warwick Ford, Thane Plambeck (VeriSign), Barbara Fox, Blair Dillaway, Brian LaMacchia (Microsoft), Jeremy Epstein, Joe Lapp (webMethods), David Solo (CitiGroup), Mack Hicks (Bank of America), Stephen Farrell (Baltimore Technologies), Prateek Mishra (Netegrity).

Appendix A Assertion Naming Infrastructure

The Trust Assertion Architecture makes extensive use of URIs to identify assertions, resources and audiences. Although the concepts of hierarchy, ownership and equality are implicit in the URI structure and are widely used, a single authoritative definition of these concepts with respect to URIs is not currently available.

The use of a URI as an object *identifier* is a superset from the use of a URI as an object *locator*. The Trust Assertion Infrastructure introduces objects such as audiences and authorization roles that carry distinct semantics even though there is no means of locating or even resolving them.

4.1 Lexical Comparison

The comparison functions used in the Trust Assertion Infrastructure are strictly lexical and are applied *without reference* to the semantics of the underlying URI name space. The rules for lexical comparison of URIs described here differ in some respects to the rules for semantic equivalence of URIs specified in RFC 2396 [RFC2396].

Use of lexical comparison functions ensures that the comparison functions are defined even though the application may not understand the resolution semantics of the underlying name space. The complexity of client implementations is reduced through application of the following rules:

- The forward slash character '`/`' is *always* interpreted as a separator for different levels in the name space hierarchy. No other character is interpreted as a separator.

- Comparison is always performed within the ASCII character set encoding of the URI.

- Characters describes as escaped, reserved and unreserved in RFC 2396 are always regarded as being so.

RFC 2396 defines rules for semantic equivalence of URIs. To simplify client implementation the following forms of URI are differentiated:

- A URI that specifies the default port explicitly is NOT equivalent to a URI that specified the default port implicitly (i.e. `http://site.test/` is distinct from `http://site.test:80/`).

Differentiating between explicitly and implicitly defined port numbers ensures that lexical comparison is consistent even though a client may not understand the resolution semantics of a URL scheme.

The following forms of URI are never differentiated:

- A URI that does end in a forward slash character '`/`' is directly equivalent to the same URI with a slash character appended at the end.

- A URI in which a character is escaped is directly equivalent to one in which the character is not escaped. Where more than one means of character escape is defined for the same character no distinction is made on the basis of the escape mechanism chosen.

Applying these rules the following URIs are not differentiated.

```
http://site.test/my+resource
http://site.test/my%20resource
http://site.test/my+resource/
http://site.test/my%20resource
```

4.1.1 Lexical Comparison

Lexical inequality operators (greater than, less than, equal to) are defined by applying pair wise comparisons to successive un-escaped octets of each URI starting with the first and ending with the last as follows:

- If $a = b =$ '' then $A = B$ **otherwise**
- If $a =$ '' then $A < B$ **otherwise**
- If $b =$ '' then $A > B$ **otherwise**
- If $a > b$ then $A > B$ **otherwise**
- If $a < b$ then $A < B$ **otherwise**
- $a = b \neq$ '', the next characters are compared

Examples: Applying the lexical comparison procedure defined above the following URIs are presented in lexical order, least first:

```
http://www.site.test/A1
http://www.site.test/A1/%20aaa      is equal to:
http://www.site.test/A1/+aaa
http://www.site.test/A1/aaaa
http://www.site.test/A1/zzzz
http://www.site.test/A2
http://www.site.test/A2/aaaa
```

4.1.2 Superiority

A URI *A* has the property lexical superiority with respect to the URI *B* if and only if:

- $B = A \wedge$ '/' $\wedge C$ **or**
- $B = A \wedge C$ and $A = D \wedge$ '/'

Where the symbol ^ represents string concatenation.

The symbol ▷ is used to denote the relationship *A* is superior to *B* as $A \triangleright B$.

Examples

The URI:

```
http://www.site.test/A1
```

Is lexically superior to the following URIs

```
http://www.site.test/A1/aaaa
http://www.site.test/A1/zzzz
http://www.site.test/A1/aaaa/aaaa
```

It is not superior to the following URIs:

```
http://www.site.test:80/A1/aaaa
http://www.site.test/a1/aaaa
http://www.site.test/
```

4.1.3 Membership of an Interval

A URI A is said to be a member of the interval X:Y if and only if:

- $A \geq X$ and $X \leq Y$

4.2 Ownership

Administrative ownership of an identifier is the power to bind semantics to that identifier.

The Domain Name System provides a distributed means of defining administrative ownership. The owner of the domain name `xyz.test` has the ability to configure services such as mail, web and ftp to be bound to that name through configuration of the DNS system.

URNs permit other namespaces to be incorporated into the URI name space. Many namespaces (e.g. ISBN) explicitly define the concept of namespace ownership.

Appendix B References

[RFC-2396] T. Berners-Lee, R. Fielding and L. Masinter. *Uniform Resource Identifiers (URI): Generic Syntax* RFC 2396, August 1998, Internet Engineering Taskforce. http://www.rfc-editor.org/rfc/rfc2396.txt.

[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

[WDSL] 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

[XKMS] P. Hallam-Baker, J. Epstein, B. Fox, *XML Key Management Specification (XKMS) 1.0*, November 27th 2000.

[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-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/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/xmlschema-2/

Appendix C Legal Notices

Copyright

Intellectual Property Statement

Neither the authors of this document, nor their companies take any 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 do they represent that they have made any effort to identify any such rights.

Disclaimer

This document and the information contained herein is provided on an "AS IS" basis and THE AUTHORS AND THEIR COMPANIES DISCLAIM 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.