



1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35

# Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)

**Document identifier:** draft-sstc-core-27

**Location:** <http://www.oasis-open.org/committees/security/docs>

**Publication date:** February 14th 2002

**Maturity Level:** Committee Working Draft

**Send comments to:** [security-services-comment@lists.oasis-open.org](mailto:security-services-comment@lists.oasis-open.org)

Note: Before sending a message to this list you must first subscribe; send an email message to [security-services-comment-request@lists.oasis-open.org](mailto:security-services-comment-request@lists.oasis-open.org) with the word "subscribe" as the body of the message.

**Editors:**

Phillip Hallam-Baker, VeriSign,  
Eve Maler, Sun Microsystems

**Contributors:**

Carlisle Adams, Entrust  
Scott Cantor, The Ohio State University  
Marc Chanliau, Netegrity  
Nigel Edwards, Hewlett-Packard  
Marlena Erdos, Tivoli  
Stephen Farrell, Baltimore Technologies  
Simon Godik, Crosslogic  
Jeff Hodges, Oblix  
Charles Knouse, Oblix  
Hal Lockhart, Entegriety Solutions  
Chris McLaren, Netegrity  
Prateek Mishra, Netegrity  
RL "Bob" Morgan, University of Washington  
Tim Moses, Entrust  
David Orchard, BEA  
Joe Pato, Hewlett Packard  
Darren Platt, RSA Security  
Irving Reid, Baltimore Technologies  
Krishna Sankar, Cisco Systems Inc

|    |   |           |
|----|---|-----------|
| 35 |   |           |
| 36 | <b>ASSERTIONS AND PROTOCOL FOR THE OASIS SECURITY ASSERTION MARKUP</b>              |           |
| 37 | <b>LANGUAGE (SAML)</b>  | <b>1</b>  |
| 38 | <b>1. INTRODUCTION</b>  | <b>6</b>  |
| 39 | 1.1. NOTATION   | 6         |
| 40 | 1.2. SCHEMA ORGANIZATION AND NAMESPACES   | 6         |
| 41 | 1.2.1. <i>Time Values.</i>  | 7         |
| 42 | 1.2.2. <i>Comparing SAML values</i>   | 7         |
| 43 | 1.3. SAML CONCEPTS (NON-NORMATIVE)  | 7         |
| 44 | 1.3.1. <i>Overview</i>  | 7         |
| 45 | 1.3.2. <i>SAML and URI-Based Identifiers</i>  | 9         |
| 46 | 1.3.3. <i>SAML and Extensibility</i>  | 9         |
| 47 | <b>2. SAML ASSERTIONS</b>   | <b>10</b> |
| 48 | 2.1. SCHEMA HEADER AND NAMESPACE DECLARATIONS                                       | 10        |
| 49 | 2.2. SIMPLE TYPES   | 10        |
| 50 | 2.2.1. <i>Simple Types IDType and IDReferenceType</i>                               | 10        |
| 51 | 2.2.2. <i>Simple Type DecisionType</i>  | 11        |
| 52 | 2.3. ASSERTIONS   | 11        |
| 53 | 2.3.1. <i>Element &lt;AssertionSpecifier&gt;</i>                                    | 11        |
| 54 | 2.3.2. <i>Element &lt;AssertionID&gt;</i>   | 12        |
| 55 | 2.3.3. <i>Element &lt;Assertion&gt;</i>   | 12        |
| 56 | 2.3.3.1. <i>Element &lt;Conditions&gt;</i>  | 13        |
| 57 | 2.3.3.1.1 <i>Attributes NotBefore and NotOnOrAfter</i>                              | 14        |
| 58 | 2.3.3.1.2 <i>Element &lt;Condition&gt;</i>  | 14        |
| 59 | 2.3.3.1.3 <i>Elements &lt;AudienceRestrictionCondition&gt; and &lt;Audience&gt;</i> | 14        |
| 60 | 2.3.3.1.4 <i>Elements &lt;TargetRestrictionCondition&gt; and &lt;Target&gt;</i>     | 15        |
| 61 | 2.3.3.2. <i>Elements &lt;Advice&gt; and &lt;AdviceElement&gt;</i>                   | 15        |
| 62 | 2.4. STATEMENTS   | 16        |

|    |  |    |
|----|--|----|
| 63 | <b>2.4.1. Element &lt;Statement&gt;</b>  | 16 |
| 64 | <b>2.4.2. Element &lt;SubjectStatement&gt;</b>   | 16 |
| 65 | 2.4.2.1. Element <Subject>   | 17 |
| 66 | 2.4.2.2. Element <NameIdentifier>  | 17 |
| 67 | 2.4.2.3. Elements <SubjectConfirmation>, <ConfirmationMethod>, and <SubjectConfirmationData> | 17 |
| 68 | <b>2.4.3. Element &lt;AuthenticationStatement&gt;</b>  | 18 |
| 69 | 2.4.3.1. Element <AuthenticationLocality>  | 19 |
| 70 | 2.4.3.2. Element <AuthorityBinding>  | 19 |
| 71 | <b>2.4.4. Element &lt;AuthorizationDecisionStatement&gt;</b>                                 | 20 |
| 72 | 2.4.4.1. Elements <Actions> and <Action>   | 21 |
| 73 | 2.4.4.2. Element <Evidence>  | 21 |
| 74 | <b>2.4.5. Element &lt;AttributeStatement&gt;</b>   | 21 |
| 75 | 2.4.5.1. Elements <AttributeDesignator> and <Attribute>                                      | 22 |
| 76 | 2.4.5.1.1. Element <AttributeValue>  | 22 |
| 77 | <b>3. SAML PROTOCOL</b>  | 24 |
| 78 | <b>3.1. SCHEMA HEADER AND NAMESPACE DECLARATIONS</b>   | 24 |
| 79 | <b>3.2. REQUESTS</b>   | 24 |
| 80 | <b>3.2.1. Complex Type RequestAbstractType</b>   | 24 |
| 81 | 3.2.1.1. Element <RespondWith>   | 25 |
| 82 | <b>3.2.2. Element &lt;Request&gt;</b>  | 26 |
| 83 | <b>3.2.3. Element &lt;AssertionArtifact&gt;</b>  | 27 |
| 84 | <b>3.3. QUERIES</b>  | 27 |
| 85 | <b>3.3.1. Element &lt;Query&gt;</b>  | 27 |
| 86 | <b>3.3.2. Element &lt;SubjectQuery&gt;</b>   | 27 |
| 87 | <b>3.3.3. Element &lt;AuthenticationQuery&gt;</b>  | 27 |
| 88 | <b>3.3.4. Element &lt;AttributeQuery&gt;</b>   | 28 |
| 89 | <b>3.3.5. Element &lt;AuthorizationDecisionQuery&gt;</b>                                     | 28 |
| 90 | <b>3.4. RESPONSES</b>  | 29 |
| 91 | <b>3.4.1. Complex Type ResponseAbstractType</b>  | 29 |
| 92 | <b>3.4.2. Element &lt;Response&gt;</b>   | 30 |

|     |   |           |
|-----|---|-----------|
| 93  | <b>3.4.3. Element &lt;Status&gt;</b>  | 30        |
| 94  | 3.4.3.1. Element <StatusCode>   | 31        |
| 95  | 3.4.3.2. Element <SubStatusCode>  | 31        |
| 96  | 3.4.3.3. Element <StatusMessage>  | 32        |
| 97  | 3.4.3.4. Element <StatusDetail>   | 32        |
| 98  | <b>3.4.4. Responses to &lt;AuthenticationQuery&gt; and &lt;AttributeQuery&gt;</b> | 32        |
| 99  | <b>4. SAML VERSIONING</b>   | <b>34</b> |
| 100 | <b>4.1. ASSERTION VERSION</b>   | 34        |
| 101 | <b>4.2. REQUEST VERSION</b>   | 34        |
| 102 | <b>4.3. RESPONSE VERSION</b>  | 35        |
| 103 | <b>5. SAML &amp; XML-SIGNATURE SYNTAX AND PROCESSING</b>                          | <b>36</b> |
| 104 | <b>5.1. SIGNING ASSERTIONS</b>  | 36        |
| 105 | <b>5.2. REQUEST/RESPONSE SIGNING</b>  | 37        |
| 106 | <b>5.3. SIGNATURE INHERITANCE</b>   | 37        |
| 107 | <b>5.3.1. Rationale</b>   | 37        |
| 108 | <b>5.3.2. Rules for SAML Signature Inheritance</b>                                | 37        |
| 109 | <b>5.4. XML SIGNATURE PROFILE</b>   | 37        |
| 110 | <b>5.4.1. Signing formats</b>   | 37        |
| 111 | <b>5.4.2. CanonicalizationMethod</b>  | 37        |
| 112 | <b>5.4.3. Transforms</b>  | 38        |
| 113 | <b>5.4.4. KeyInfo</b>   | 38        |
| 114 | <b>5.4.5. Binding between statements in a multi-statement assertion</b>           | 38        |
| 115 | <b>6. SAML EXTENSIONS</b>   | <b>39</b> |
| 116 | <b>6.1. ASSERTION SCHEMA EXTENSION</b>  | 39        |
| 117 | <b>6.2. PROTOCOL SCHEMA EXTENSION</b>   | 39        |
| 118 | <b>6.3. USE OF TYPE DERIVATION AND SUBSTITUTION GROUPS</b>                        | 40        |

|     |   |           |
|-----|---|-----------|
| 119 | <b>7. SAML-DEFINED IDENTIFIERS</b>                      | <b>41</b> |
| 120 | 7.1. CONFIRMATION METHOD IDENTIFIERS                    | 41        |
| 121 | 7.1.1. SAML Artifact:                                   | 41        |
| 122 | 7.1.2. SAML Artifact (SHA-1):                           | 41        |
| 123 | 7.1.3. Holder of Key:                                   | 41        |
| 124 | 7.1.4. Sender Vouches:                                  | 41        |
| 125 | 7.1.5. Password (Pass-Through):                         | 41        |
| 126 | 7.1.6. Password (One-Way-Function SHA-1):               | 42        |
| 127 | 7.1.7. Kerberos   | 42        |
| 128 | 7.1.8. SSL/TLS Certificate Based Client Authentication: | 42        |
| 129 | 7.1.9. Object Authenticator (SHA-1):                    | 42        |
| 130 | 7.1.10. PKCS#7  | 42        |
| 131 | 7.1.11. Cryptographic Message Syntax                    | 43        |
| 132 | 7.1.12. XML Digital Signature                           | 43        |
| 133 | 7.2. ACTION NAMESPACE IDENTIFIERS                       | 43        |
| 134 | 7.2.1. Read/Write/Execute/Delete/Control:               | 43        |
| 135 | 7.2.2. Read/Write/Execute/Delete/Control with Negation: | 43        |
| 136 | 7.2.3. Get/Head/Put/Post:                               | 44        |
| 137 | 7.2.4. UNIX File Permissions:                           | 44        |
| 138 | <b>8. SAML SCHEMA LISTINGS</b>                          | <b>45</b> |
| 139 | 8.1. ASSERTION SCHEMA                                   | 45        |
| 140 | 8.2. PROTOCOL SCHEMA                                    | 48        |
| 141 | <b>9. REFERENCES</b>                                    | <b>52</b> |
| 142 | <b>APPENDIX A. NOTICES</b>                              | <b>54</b> |
| 143 |   |           |

# 144 1. Introduction

145 This specification defines the syntax and semantics for XML-encoded SAML assertions, protocol  
146 requests, and protocol responses. These constructs are typically embedded in other structures for  
147 transport, such as HTTP form POSTs and XML-encoded SOAP messages. The SAML specification  
148 for bindings and profiles [SAMLBind] provides frameworks for this embedding and transport. Files  
149 containing just the SAML assertion schema [SAML-XSD] and protocol schema [SAML-PSD] are  
150 available.

151 The following sections describe how to understand the rest of this specification.

## 152 1.1. Notation

153 This specification uses schema documents conforming to W3C XML Schema [Schema1] and  
154 normative text to describe the syntax and semantics of XML-encoded SAML assertions and  
155 protocol messages.

156 The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD",  
157 "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be  
158 interpreted as described in IETF RFC 2119 [RFC2119]:

159 *"they MUST only be used where it is actually required for interoperability or to limit*  
160 *behavior which has potential for causing harm (e.g., limiting retransmissions)"*

161 These keywords are thus capitalized when used to unambiguously specify requirements over  
162 protocol and application features and behavior that affect the interoperability and security of  
163 implementations. When these words are not capitalized, they are meant in their natural-language  
164 sense.

165 Listings of SAML schemas appear like this.

166 `Example code listings appear like this.`

168 Conventional XML namespace prefixes are used throughout the listings in this specification to  
169 stand for their respective namespaces (see Section 1.2) as follows, whether or not a namespace  
170 declaration is present in the example:

- 171 • The prefix `saml`: stands for the SAML assertion namespace.
- 172 • The prefix `samlp`: stands for the SAML request-response protocol namespace.
- 173 • The prefix `ds`: stands for the W3C XML Signature namespace.
- 174 • The prefix `xsd`: stands for the W3C XML Schema namespace in example listings. In  
175 schema listings, this is the default namespace and no prefix is shown.

176 This specification uses the following typographical conventions in text: `<SAMLElement>`,  
177 `<ns:ForeignElement>`, `Attribute`, **Datatype**, `OtherCode`.

## 178 1.2. Schema Organization and Namespaces

179 The SAML assertion structures are defined in a schema [SAML-XSD] associated with the following  
180 XML namespace:[PHB1]

181 <http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-assertion-27.xsd>

182 The SAML request-response protocol structures are defined in a schema [SAML-PSD]  
183 associated with the following XML namespace:

184 <http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-protocol-27.xsd>

185                   **Note:** The SAML namespace names are temporary and will change when  
186                   SAML 1.0 is finalized.

187   The assertion schema is imported into the protocol schema. Also imported into both schemas is the  
188   schema for XML Signature [**XMLSig-XSD**], which is associated with the following XML namespace:  
189   <http://www.w3.org/2000/09/xmldsig#>

### 190   **1.2.1. Time Values.**

191   All SAML time values have the type **dateTime**, which is built in to the W3C XML Schema Datatypes  
192   specification [**Schema2**] and MUST be expressed in UTC form.

193   SAML applications SHOULD NOT rely on other applications supporting time resolution finer than  
194   milliseconds. Implementations MUST NOT generate time instants that specify leap seconds.

### 195   **1.2.2. Comparing SAML values**

196   Unless otherwise noted, all elements in SAML documents that have the XML Schema "string" type,  
197   or a type derived from that, MUST be compared using an exact binary comparison. In particular,  
198   SAML implementations and deployments MUST NOT depend on case-insensitive string  
199   comparisons, normalization or trimming of white space, or conversion of locale-specific formats  
200   such as numbers or currency. This requirement is intended to conform to the W3C Requirements  
201   for String Identity, Matching, and String Indexing [**W3C-CHAR**].

202   If an implementation is comparing values that are represented using different character encodings,  
203   the implementation MUST use a comparison method that returns the same result as converting  
204   both values to the Unicode character encoding (<http://www.unicode.org>), Normalization Form C  
205   [**UNICODE-C**] and then performing an exact binary comparison. This requirement is intended to  
206   conform to the W3C Character Model for the World Wide Web ([**W3C-CharMod**]), and in particular  
207   the rules for Unicode-normalized Text.

208   Applications that compare data received in SAML documents to data from external sources MUST  
209   take into account the normalization rules specified for XML. Text contained within elements is  
210   normalized so that line endings are represented using linefeed characters (ASCII code 10<sub>Decimal</sub>), as  
211   described in section 2.11 of the XML Recommendation [**XML**]. Attribute values defined as strings  
212   (or types derived from strings) are normalized as described in section 3.3.3 [**XML**] all white space  
213   characters are replaced with blanks (ASCII code 32<sub>Decimal</sub>).

214   The SAML specification does not define collation or sorting order for attribute or element values.  
215   SAML implementations MUST NOT depend on specific sorting orders for values, because these  
216   may differ depending on the locale settings of the hosts involved.

## 217   **1.3. SAML Concepts (Non-Normative)**

218   This section is informative only and is superseded by any contradicting information in the normative  
219   text in Sections 1.2 and following. A glossary of SAML terms and concepts [**SAMLGloss**] is  
220   available.

### 221   **1.3.1. Overview**

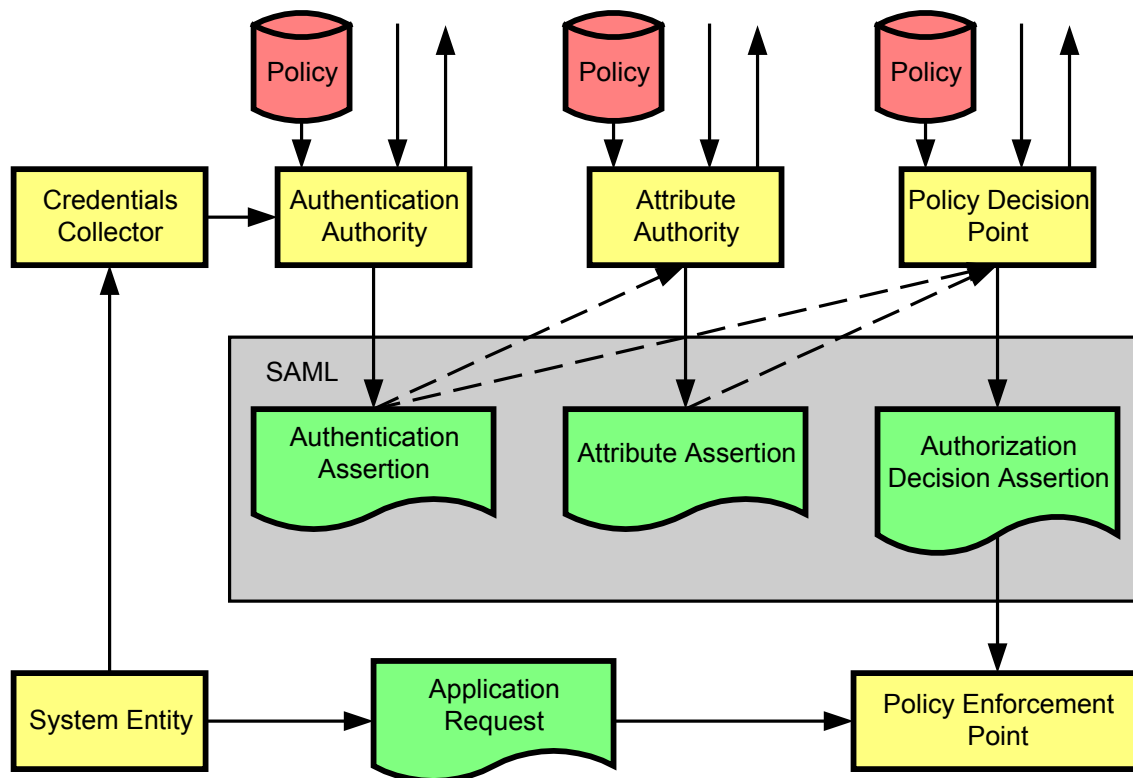
222   The Security Assertion Markup Language (SAML) is an XML-based framework for exchanging  
223   security information. This security information is expressed in the form of assertions about subjects,  
224   where a subject is an entity (either human or computer) that has an identity in some security  
225   domain. A typical example of a subject is a person, identified by his or her email address in a  
226   particular Internet DNS domain.

227 Assertions can convey information about authentication acts performed by subjects, attributes of  
 228 subjects, and authorization decisions about whether subjects are allowed to access certain  
 229 resources. Assertions are represented as XML constructs and have a nested structure, whereby a  
 230 single assertion might contain several different internal statements about authentication,  
 231 authorization, and attributes. Note that authentication assertions merely describe acts of  
 232 authentication that happened previously.

233 Assertions are issued by SAML authorities, namely, authentication authorities, attribute authorities,  
 234 and policy decision points. SAML defines a protocol by which clients can request assertions from  
 235 SAML authorities and get a response from them. This protocol, consisting of XML-based request  
 236 and response message formats, can be bound to many different underlying communications and  
 237 transport protocols; SAML currently defines one binding, to SOAP over HTTP.

238 SAML authorities can use various sources of information, such as external policy stores and  
 239 assertions that were received as input in requests, in creating their responses. Thus, while clients  
 240 always consume assertions, SAML authorities can be both producers and consumers of assertions.

241 The following model is conceptual only; for example, it does not account for real-world information  
 242 flow or the possibility of combining of authorities into a single system.



243

244

**Figure 1 The SAML Domain Model**

245 One major design goal for SAML is Single Sign-On (SSO), the ability of a user to authenticate in  
 246 one domain and use resources in other domains without re-authenticating. However, SAML can be  
 247 used in various configurations to support additional scenarios as well. Several profiles of SAML are  
 248 defined that support different styles of SSO and the securing of SOAP payloads.

249 The assertion and protocol data formats are defined in this specification. The bindings and profiles  
 250 are defined in a separate specification **[SAMLBind]**. A conformance program for SAML is defined  
 251 in the conformance specification **[SAMLConform]**. Security issues are discussed in a separate  
 252 security and privacy considerations specification **[SAMLSecure]**.



253 **1.3.2. SAML and URI-Based Identifiers**

254 SAML defines some identifiers to manage references to well-known concepts and sets of values.  
255 For example, the SAML-defined identifier for the Kerberos subject confirmation method is as  
256 follows:

257 **urn:ietf:rfc:1510**

258 For another example, the SAML-defined identifier for the set of possible actions on a resource  
259 consisting of Read/Write/Execute/Delete/Control is as follows:

260 **http://www.oasis-open.org/committees/security/docs/draft-sstc-core-26#rwedc**

261 These identifiers are defined as Uniform Resource Identifiers (URIs), but they are not necessarily  
262 able to be resolved to some Web resource. At times SAML authorities need to use identifier strings  
263 of their own design, for example, for assertion IDs or additional kinds of confirmation methods not  
264 covered by SAML-defined identifiers. In these cases, using a URI form is not required; if it is used, it  
265 is not required to be resolvable to some Web resource. However, using URIs – particularly URLs  
266 based on the `http:` scheme – is likely to mitigate problems with clashing identifiers to some  
267 extent.

268 The Read/Write/Execute/Delete/Control identifier above is an example of a namespace (not in the  
269 sense of an XML namespace). SAML uses this namespace mechanism to manage the universe of  
270 possible types of actions and possible names of attributes.

271 See section 7 for a list of SAML-defined identifiers.

272 **1.3.3. SAML and Extensibility**

273 The XML formats for SAML assertions and protocol messages have been designed to be  
274 extensible.

275 However, it is possible that the use of extensions will harm interoperability and therefore the use of  
276 extensions SHOULD be carefully considered.

## 277 2. SAML Assertions

278 An assertion is a package of information that supplies one or more statements made by an issuer.  
279 SAML allows issuers to make three different kinds of assertion statement:

- 280 • **Authentication:** The specified subject was authenticated by a particular means at a  
281 particular time.
- 282 • **Authorization Decision:** A request to allow the specified subject to access the specified  
283 resource has been granted or denied.
- 284 • **Attribute:** The specified subject is associated with the supplied attributes.

285 Assertions have a nested structure. A series of inner elements representing authentication  
286 statements, authorization decision statements, and attribute statements contain the specifics, while  
287 an outer generic assertion element provides information that is common to all of the statements.

### 288 2.1. Schema Header and Namespace Declarations

289 The following schema fragment defines the XML namespaces and other header information for the  
290 assertion schema:[PHB2]

```
291 <schema  
292   targetNamespace="http://www.oasis-open.org/committees/security/docs/draft-  
293   sstc-schema-assertion-27.xsd"  
294   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
295   xmlns:saml="http://www.oasis-open.org/committees/security/docs/draft-sstc-  
296   schema-assertion-27.xsd"  
297   xmlns="http://www.w3.org/2001/XMLSchema"  
298   elementFormDefault="unqualified">  
299   <import namespace="http://www.w3.org/2000/09/xmldsig#"  
300     schemaLocation="xmldsig-core-schema.xsd"/>[PHB3]  
301   <annotation>  
302     <documentation>draft-sstc-schema-assertion-27.xsd</documentation>  
303   </annotation>  
304   ...  
305 </schema>
```

### 306 2.2. Simple Types

307 The following sections define the SAML assertion-related simple types.

#### 308 2.2.1. Simple Types **IDType** and **IDReferenceType**

309 The **IDType** simple type is used to declare identifiers to assertions, requests, and responses. The  
310 **IDReferenceType** is used to reference identifiers of type **IDType**.

311 Values declared to be of type **IDType** MUST satisfy the following properties:

- 312 • Any party that assigns an identifier MUST ensure that there is negligible probability that that  
313 party or any other party will accidentally assign the same identifier to a different data object.
- 314 • Where a data object declares that it has a particular identifier, there MUST be exactly one  
315 such declaration.

316 The mechanism by which the application ensures that the identifier is unique is left to the  
317 implementation. In the case that a pseudorandom technique is employed, the probability of two  
318 randomly chosen identifiers being identical MUST be less than  $2^{-128}$  and SHOULD be less than  
319  $2^{-160}$ . This requirement MAY be met by applying Base64 encoding to a randomly chosen value 128  
320 or 160 bits in length.

321 It is OPTIONAL for an identifier based on **IDType** to be resolvable in principle to some resource. In  
322 the case that the identifier is resolvable in principle (for example, the identifier is in the form of a  
323 URI reference), it is OPTIONAL for the identifier to be dereferenceable.

324 The following schema fragment defines the **IDType** and **IDReferenceType** simple types:

```
325 <simpleType name="IDType">  
326   <restriction base="string"/>  
327 </simpleType>  
328 <simpleType name="IDReferenceType">  
329   <restriction base="string"/>  
330 </simpleType>
```

### 331 2.2.2. Simple Type DecisionType

332 The **DecisionType** simple type defines the possible values to be reported as the status of an  
333 authorization decision statement.

334 Permit

335       The specified action is permitted.

336 Deny

337       The specified action is denied.

338 Indeterminate

339       No assessment is made as to whether the specified action is permitted or denied.

340 The following schema fragment defines the **DecisionType** simple type:

```
341 <simpleType name="DecisionType">  
342   <restriction base="string">  
343     <enumeration value="Permit"/>  
344     <enumeration value="Deny"/>  
345     <enumeration value="Indeterminate"/>  
346   </restriction>  
347 </simpleType>
```

## 348 2.3. Assertions

349 The following sections define the SAML constructs that contain assertion information.

### 350 2.3.1. Element <AssertionSpecifier>

351 The <AssertionSpecifier> element specifies an assertion either by reference or by value. It  
352 contains one of the following elements:

353 <AssertionIDReference>

354       Specifies an assertion by reference to the value of the assertion's `AssertionID` attribute.

355 <Assertion>

356       Specifies an assertion by value.

357 The following schema fragment defines the <AssertionSpecifier> element and its  
358 **AssertionSpecifierType** complex type:

```
359 <element name="AssertionSpecifier" type="saml:AssertionSpecifierType"/>  
360 <complexType name="AssertionSpecifierType">  
361   <choice>  
362     <element ref="saml:AssertionIDReference"/>  
363     <element ref="saml:Assertion"/>  
364   </choice>  
365 </complexType>
```

## 366 2.3.2. Element <AssertionID>

367 The <AssertionID> element makes a reference to a SAML assertion by means of the value of  
368 the assertion's AssertionID attribute.

369 The following schema fragment defines the <AssertionID> element:

```
370 <element name="AssertionIDReference" type="saml:IDReferenceType"/>
```

## 371 2.3.3. Element <Assertion>

372 The <Assertion> element is of **AssertionType** complex type. This type specifies the basic  
373 information that is common to all assertions, including the following elements and attributes:

374 MajorVersion [Required]

375 The major version of this assertion. The identifier for the version of SAML defined in this  
376 specification is 1. Processing of this attribute is specified in Section 3.4.4.

377 MinorVersion [Required]

378 The minor version of this assertion. The identifier for the version of SAML defined in this  
379 specification is 0. Processing of this attribute is specified in Section 3.4.4.

380 AssertionID [Required]

381 The identifier for this assertion. It is of type **IDType**, and MUST follow the requirements  
382 specified by that type for identifier uniqueness.

383 Issuer [Required]

384 The issuer of the assertion. The name of the issuer is provided as a string. The issuer  
385 name SHOULD be unambiguous to the intended relying parties. SAML applications may  
386 use an identifier such as a URI that is designed to be unambiguous regardless of context.

387 IssueInstant [Required]

388 The time instant of issue in UTC as described in section 1.2.1.

389 <Conditions> [Optional]

390 Conditions that MUST be taken into account in assessing the validity of the assertion.

391 <Advice> [Optional]

392 Additional information related to the assertion that assists processing in certain situations  
393 but which MAY be ignored by applications that do not support its use.

394 <Signature> [Optional]

395 An XML Signature that authenticates the assertion, see section 5.

396 One or more of the following statement elements:

397 <Statement>

398 A statement defined in an extension schema.

399 <SubjectStatement>

400 A subject statement defined in an extension schema.

401 <AuthenticationStatement>

402 An authentication statement.

403 <AuthorizationDecisionStatement>

404 An authorization decision statement.

405 <AttributeStatement>

406 An attribute statement.

407 The following schema fragment defines the <Assertion> element and its **AssertionType**  
408 complex type:

```

409 <element name="Assertion" type="saml:AssertionType"/>
410 <complexType name="AssertionType">
411   <sequence>
412     <element ref="saml:Conditions" minOccurs="0"/>
413     <element ref="saml:Advice" minOccurs="0"/>
414     <choice maxOccurs="unbounded">
415       <element ref="saml:Statement"/>
416       <element ref="saml:SubjectStatement"/>
417       <element ref="saml:AuthenticationStatement"/>
418       <element ref="saml:AuthorizationDecisionStatement"/>
419       <element ref="saml:AttributeStatement"/>
420     </choice>
421     <element ref="ds:Signature" minOccurs="0"/>
422   </sequence>
423   <attribute name="MajorVersion" type="integer" use="required"/>
424   <attribute name="MinorVersion" type="integer" use="required"/>
425   <attribute name="AssertionID" type="saml:IDType" use="required"/>
426   <attribute name="Issuer" type="string" use="required"/>
427   <attribute name="IssueInstant" type="dateTime" use="required"/>
428 </complexType>

```

### 429 2.3.3.1. Element <Conditions>

430 If an assertion contains a <Conditions> element, the validity of the assertion is dependent on the  
431 conditions provided. Each condition evaluates to a status of Valid, Invalid, or  
432 Indeterminate. The validity status of an assertion is the conjunction of the validity status of each  
433 of the conditions it contains, as follows:

- 434 • If any condition evaluates to Invalid, the assertion status is Invalid.
- 435 • If no condition evaluates to Invalid and one or more conditions evaluate to  
436 Indeterminate, the assertion status is Indeterminate.
- 437 • If no conditions are supplied or all the specified conditions evaluate to Valid, the assertion  
438 status is Valid.

439 Note that an assertion that has validity status 'Valid' may not be trustworthy by reasons such as not  
440 being issued by a trustworthy issuer or not being authenticated by a trustworthy signature.

441 The <Conditions> element MAY be extended to contain additional conditions. If an element  
442 contained within a <Conditions> element is encountered that is not understood, the status of the  
443 condition MUST be evaluated to Indeterminate.

444 The <Conditions> element MAY contain the following elements and attributes:

445 NotBefore [Optional]

446 Specifies the earliest time instant at which the assertion is valid. The time value is encoded  
447 in UTC as described in section 1.2.1.

448 NotOnOrAfter [Optional]

449 Specifies the time instant at which the assertion has expired. The time value is encoded in  
450 UTC as described in section 1.2.1.

451 <Condition> [Any Number]

452 Provides an extension point allowing extension schemas to define new conditions.

453 <AudienceRestrictionCondition> [Any Number]

454 Specifies that the assertion is addressed to a particular audience.

455 <TargetRestrictionCondition> [Any Number]

456 The <TargetRestriction> condition is used to limit the use of the assertion to a particular  
457 relying party.

458 The following schema fragment defines the `<Conditions>` element and its **ConditionsType**  
459 complex type:

```
460 <element name="Conditions" type="saml:ConditionsType"/>
461 <complexType name="ConditionsType">
462   <choice minOccurs="0" maxOccurs="unbounded">
463     <element ref="saml:Condition"/>
464     <element ref="saml:AudienceRestrictionCondition"/>
465     <element ref="saml:TargetRestrictionCondition"/>
466   </choice>
467   <attribute name="NotBefore" type="dateTime" use="optional"/>
468   <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
469 </complexType>
```

#### 470 **2.3.3.1.1 Attributes *NotBefore* and *NotOnOrAfter***

471 The `NotBefore` and `NotOnOrAfter` attributes specify time limits on the validity of the assertion.

472 The `NotBefore` attribute specifies the time instant at which the validity interval begins. The  
473 `NotOnOrAfter` attribute specifies the time instant at which the validity interval has ended.

474 If the value for either `NotBefore` or `NotOnOrAfter` is omitted it is considered unspecified. If the  
475 `NotBefore` attribute is unspecified (and if any other conditions that are supplied evaluate to  
476 `Valid`), the assertion is valid at any time before the time instant specified by the `NotOnOrAfter`  
477 attribute. If the `NotOnOrAfter` attribute is unspecified (and if any other conditions that are supplied  
478 evaluate to `Valid`), the assertion is valid from the time instant specified by the `NotBefore`  
479 attribute with no expiry. If neither attribute is specified (and if any other conditions that are supplied  
480 evaluate to `Valid`), the assertion is valid at any time.

481 The `NotBefore` and `NotOnOrAfter` attributes are defined to have the **dateTime** simple type that  
482 is built in to the W3C XML Schema Datatypes specification [**Schema2**]. All time instants are  
483 specified in Universal Coordinated Time (UTC) as described in section 1.2.1.

484 Implementations MUST NOT generate time instants that specify leap seconds.

#### 485 **2.3.3.1.2 Element `<Condition>`**

486 The `<Condition>` element serves as an extension point for new conditions. Its  
487 **ConditionAbstractType** complex type is abstract; extension elements MUST use the `xsi:type`  
488 attribute to indicate the derived type.

489 The following schema fragment defines the `<Condition>` element and its  
490 **ConditionAbstractType** complex type:

```
491 <element name="Condition" type="saml:ConditionAbstractType"/>
492 <complexType name="ConditionAbstractType" abstract="true"/>
```

#### 493 **2.3.3.1.3 Elements `<AudienceRestrictionCondition>` and `<Audience>`**

494 The `<AudienceRestrictionCondition>` element specifies that the assertion is addressed to  
495 one or more specific audiences identified by `<Audience>` elements. Although a party that is outside  
496 the audiences specified is capable of drawing conclusions from an assertion, the issuer explicitly  
497 makes no representation as to accuracy or trustworthiness to such a party. It contains the following  
498 elements:

499 `<Audience>`

500 A URI that identifies an intended audience. The URI MAY identify a document that  
501 describes the terms and conditions of audience membership.

502 The `AudienceRestrictionCondition` evaluates to `Valid` if and only if the relying party is a  
503 member of one or more of the audiences specified.

504 The issuer of an assertion cannot prevent a party to whom it is disclosed from making a decision on  
505 the basis of the information provided. However, the <AudienceRestrictionCondition>  
506 element allows the issuer to state explicitly that no warranty is provided to such a party in a  
507 machine- and human-readable form. While there can be no guarantee that a court would uphold  
508 such a warranty exclusion in every circumstance, the probability of upholding the warranty  
509 exclusion is considerably improved.

510 The following schema fragment defines the <AudienceRestrictionCondition> element and  
511 its **AudienceRestrictionConditionType** complex type:

```
512 <element name="AudienceRestrictionCondition"  
513     type="saml:AudienceRestrictionConditionType"/>  
514 <complexType name="AudienceRestrictionConditionType">  
515     <complexContent>  
516         <extension base="saml:ConditionAbstractType">  
517             <sequence>  
518                 <element ref="saml:Audience" maxOccurs="unbounded"/>  
519             </sequence>  
520         </extension>  
521     </complexContent>  
522 </complexType>  
523 <element name="Audience" type="anyURI"/>
```

#### 524 **2.3.3.1.4 Elements <TargetRestrictionCondition> and <Target>**

525 The <TargetRestrictionCondition> element is used to limit the use of the assertion to a particular  
526 relying party. This is useful to prevent malicious forwarding of assertions to unintended recipients. It  
527 contains the following elements:

528 <Target>  
529 A URI that identifies an intended relying party.

530 The TargetRestrictionCondition evaluates to Valid if and only if one or more URIs identify the  
531 recipient or a resource managed by the recipient.

532 The following schema fragment defines the <TargetRestrictionCondition> element and its  
533 **TargetRestrictionConditionType** complex type:

```
534 <element name="TargetRestrictionCondition"  
535     type="saml:TargetRestrictionConditionType"/>  
536 <complexType name="TargetRestrictionConditionType">  
537     <complexContent>  
538         <extension base="saml:ConditionAbstractType">  
539             <sequence>  
540                 <element ref="saml:Target"  
541                     minOccurs="1" maxOccurs="unbounded"/>  
542             </sequence>  
543         </extension>  
544     </complexContent>  
545 </complexType>  
546 <element name="Target" type="anyURI"/>
```

#### 547 **2.3.3.2. Elements <Advice> and <AdviceElement>**

548 The <Advice> element contains any additional information that the issuer wishes to provide. This  
549 information MAY be ignored by applications without affecting either the semantics or the validity of  
550 the assertion.

551 The <Advice> element contains a mixture of zero or more <AssertionSpecifier> elements,  
552 <AdviceElement> elements, and elements in other namespaces, with lax schema validation in  
553 effect for these other elements.

554 Following are some potential uses of the <Advice> element:



- 555 • Include evidence supporting the assertion claims to be cited, either directly (through  
556 incorporating the claims) or indirectly (by reference to the supporting assertions).
- 557 • State a proof of the assertion claims.
- 558 • Specify the timing and distribution points for updates to the assertion.

559 The following schema fragment defines the <Advice> element and its **AdviceType** complex type,  
560 along with the <AdviceElement> element and its **AdviceAbstractType** complex type:

```
561 <element name="Advice" type="saml:AdviceType"/>
562 <complexType name="AdviceType">
563
564     <choice minOccurs="0" maxOccurs="unbounded">
565         <element ref="saml:AssertionSpecifier"/>
566         <element ref="saml:AdviceElement"/>
567         <any namespace="##other" processContents="lax"/>
568     </choice>
569
570 </complexType>
571 <element name="AdviceElement" type="saml:AdviceAbstractType"/>
572 <complexType name="AdviceAbstractType"/>
```

## 573 2.4. Statements

574 The following sections define the SAML constructs that contain statement information.

### 575 2.4.1. Element <Statement>

576 The <Statement> element is an extension point that allows other assertion-based applications to  
577 reuse the SAML assertion framework. Its **StatementAbstractType** complex type is abstract;  
578 extension elements **MUST** use the `xsi:type` attribute to indicate the derived type.

579 The following schema fragment defines the <Statement> element and its  
580 **StatementAbstractType** complex type:

```
581 <element name="Statement" type="saml:StatementAbstractType"/>
582 <complexType name="StatementAbstractType" abstract="true"/>
```

### 583 2.4.2. Element <SubjectStatement>

584 The <SubjectStatement> element is an extension point that allows other assertion-based  
585 applications to reuse the SAML assertion framework. It contains a <Subject> element that allows  
586 an issuer to describe a subject. Its **SubjectStatementAbstractType** complex type, which extends  
587 **StatementAbstractType**, is abstract; extension elements **MUST** use the `xsi:type` attribute to  
588 indicate the derived type.

589 The following schema fragment defines the <SubjectStatement> element and its  
590 **SubjectStatementAbstractType** abstract type:

```
591 <element name="SubjectStatement" type="saml:SubjectStatementAbstractType"/>
592 <complexType name="SubjectStatementAbstractType" abstract="true">
593     <complexContent>
594         <extension base="saml:StatementAbstractType">
595             <sequence>
596                 <element ref="saml:Subject"/>
597             </sequence>
598         </extension>
599     </complexContent>
600 </complexType>
```



### 601 2.4.2.1. Element <Subject>

602 The <Subject> element specifies the principal that is the subject of the statement. It contains  
603 either or both of the following elements:

604 <NameIdentifier>

605 An identification of a subject by its name and security domain.

606 <SubjectConfirmation>

607 Information that allows the subject to be authenticated.

608 If the <Subject> element contains both a <NameIdentifier> and a  
609 <SubjectConfirmation>, the issuer is asserting that if the relying party performs the specified  
610 <SubjectConfirmation>, it can be confident that the entity presenting the assertion to the  
611 relying party is the entity that the issuer associates with the <NameIdentifier>. A <Subject>  
612 element SHOULD NOT identify more than one principal.

613 The following schema fragment defines the <Subject> element and its **SubjectType** complex  
614 type:

```
615 <element name="Subject" type="saml:SubjectType"/>
616 <complexType name="SubjectType">
617 <choice>
618 <sequence>
619 <element ref="saml:NameIdentifier"/>
620 <element ref="saml:SubjectConfirmation" minOccurs="0"/>
621 </sequence>
622 <element ref="saml:SubjectConfirmation"/>
623 </choice>
624 </complexType>
```

### 625 2.4.2.2. Element <NameIdentifier>

626 The <NameIdentifier> element specifies a subject by a combination of a name and a security  
627 domain. It has the following attributes:

628 SecurityDomain [Optional]

629 The security domain governing the name of the subject.

630 Name [Required]

631 The name of the subject.

632 The interpretation of the security domain and the name are left to individual implementations,  
633 including issues of anonymity, pseudonymity, and the persistence of the identifier with respect to  
634 the asserting and relying parties.

635 The following schema fragment defines the <NameIdentifier> element and its  
636 **NameIdentifierType** complex type:

```
637 <element name="NameIdentifier" type="saml:NameIdentifierType"/>
638 <complexType name="NameIdentifierType">
639 <attribute name="SecurityDomain" type="string"/>
640 <attribute name="Name" type="string" use="required"/>
641 </complexType>
```

### 642 2.4.2.3. Elements <SubjectConfirmation>, <ConfirmationMethod>, and 643 <SubjectConfirmationData>

644 The <SubjectConfirmation> element specifies a subject by supplying data that allows the  
645 subject to be authenticated. It contains the following elements in order:

646 <ConfirmationMethod> [One or more]

647 A URI that identifies a protocol to be used to authenticate the subject. URIs identifying  
648 common authentication protocols are listed in Section 7.

649 <SubjectConfirmationData> [Optional]  
650 Additional authentication information to be used by a specific authentication protocol.

651 <ds:KeyInfo> [Optional]  
652 An XML Signature [XMLSig] element that specifies a cryptographic key held by the  
653 subject.

654 The following schema fragment defines the <SubjectConfirmation> element and its  
655 **SubjectConfirmationType** complex type, along with the <SubjectConfirmationData>  
656 element and the <ConfirmationMethod> element:

```
657 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>  
658 <complexType name="SubjectConfirmationType">  
659 <sequence>  
660 <element ref="saml:ConfirmationMethod" maxOccurs="unbounded"/>  
661 <element ref="saml:SubjectConfirmationData" minOccurs="0"/>  
662 <element ref="ds:KeyInfo" minOccurs="0"/>  
663 </sequence>  
664 </complexType>  
665 <element name="SubjectConfirmationData" type="string"/>  
666 <element name="ConfirmationMethod" type="anyURI"/>
```

### 667 2.4.3. Element <AuthenticationStatement>

668 The <AuthenticationStatement> element supplies a statement by the issuer that its subject  
669 was authenticated by a particular means at a particular time. It is of type  
670 **AuthenticationStatementType**, which extends **SubjectStatementAbstractType** with the addition  
671 of the following element and attributes:

672 AuthenticationMethod [Optional]  
673 A URI that specifies the type of authentication that took place. URIs identifying common  
674 authentication protocols are listed in Section 7.

675 AuthenticationInstant [Optional]  
676 Specifies the time at which the authentication took place. The time value is encoded in UTC  
677 as described in section 1.2.1.

678 <AuthenticationLocality> [Optional]  
679 Specifies the DNS domain name and IP address for the system entity from which the  
680 Subject was apparently authenticated.

681 <AuthorityBinding> [Any Number]  
682 Indicates that additional information about the subject of the statement may be available.

683 The following schema fragment defines the <AuthenticationStatement> element and its  
684 **AuthenticationStatementType** complex type:

```
685 <element name="AuthenticationStatement"  
686 type="saml:AuthenticationStatementType"/>  
687 <complexType name="AuthenticationStatementType">  
688 <complexContent>  
689 <extension base="saml:SubjectStatementAbstractType">  
690 <sequence>  
691 <element ref="saml:AuthenticationLocality" minOccurs="0"/>  
692 <element ref="saml:AuthorityBinding"  
693 minOccurs="0" maxOccurs="unbounded"/>  
694 </sequence>  
695 <attribute name="AuthenticationMethod" type="anyURI"/>  
696 <attribute name="AuthenticationInstant" type="dateTime"/>  
697 </extension>  
698 </complexContent>  
699 </complexType>
```

### 700 2.4.3.1. Element <AuthenticationLocality>

701 The <AuthenticationLocality> element specifies the DNS domain name and IP address for  
702 the system entity that was authenticated. It has the following attributes:

703 IPAddress [Optional]

704 The IP address of the system entity that was authenticated.

705 DNSAddress [Optional]

706 The DNS address of the system entity that was authenticated.

707 This element is entirely advisory, since both these fields are quite easily “spoofed” but current  
708 practice appears to require its inclusion.

709 The following schema fragment defines the <AuthenticationLocality> element and its  
710 **AuthenticationLocalityType** complex type:

```
711 <element name="AuthenticationLocality"  
712         type="saml:AuthenticationLocalityType"/>  
713 <complexType name="AuthenticationLocalityType">  
714   <attribute name="IPAddress" type="string" use="optional"/>  
715   <attribute name="DNSAddress" type="string" use="optional"/>  
716 </complexType>
```

### 717 2.4.3.2. Element <AuthorityBinding>

718 The <AuthorityBinding> element may be used to indicate to a relying party receiving an  
719 AuthenticationStatement that a SAML authority may be available to provide additional information  
720 about the subject of the statement. A single SAML authority may advertise its presence over  
721 multiple protocol bindings, at multiple locations, and as more than one kind of authority by sending  
722 multiple elements as needed.

723 AuthorityKind [Required]

724 The type of SAML authority (Authentication, Attribute, or Authorization Decision) advertised  
725 by the element. The kind of authority corresponds to the derived type of SubjectQuery that  
726 the authority expects to receive (and is likely to be able to successfully answer) at the  
727 location being advertised. For example, a value of "attribute" means that an  
728 <AttributeQuery> is expected.

729 Location [Required]

730 A URI describing how to locate and communicate with the authority, the exact syntax of  
731 which depends on the protocol binding in use. For example, a binding based on HTTP will  
732 be a web URL, while a binding based on SMTP might use the "mailto" scheme.

733 Binding [Required]

734 A URI identifying the SAML protocol binding to use in communicating with the authority. All  
735 SAML protocol bindings will have an assigned URI.

736 The following schema fragment defines the <AuthorityBinding> element and its  
737 **AuthorityBindingType** complex type and **AuthorityKindType** simple type:

```
738 <element name="AuthorityBinding" type="saml:AuthorityBindingType"/>  
739 <complexType name="AuthorityBindingType">  
740   <attribute name="AuthorityKind" type="saml:AuthorityKindType"  
741           use="required"/>  
742   <attribute name="Location" type="anyURI" use="required"/>  
743   <attribute name="Binding" type="anyURI" use="required"/>  
744 </complexType>  
745 <simpleType name="AuthorityKindType">  
746   <restriction base="string">  
747     <enumeration value="authentication"/>  
748     <enumeration value="attribute"/>  
749     <enumeration value="authorization"/>
```

```
750     </restriction>
751 </simpleType>
```

#### 752 2.4.4. Element <AuthorizationDecisionStatement>

753 The <AuthorizationDecisionStatement> element supplies a statement by the issuer that the  
754 request for access by the specified subject to the specified resource has resulted in the specified  
755 decision on the basis of some optionally specified evidence.

756 The resource is identified by means of a URI. In order for the assertion to be interpreted correctly  
757 and securely the issuer and relying party MUST interpret each URI in a consistent manner. Failure  
758 to achieve a consistent URI interpretation can result in different authorization decisions depending  
759 on the encoding of the resource URI. Rules for normalizing URIs are to be found in [RFC 2396]§6

760 *In general, the rules for equivalence and definition of a normal form, if any, are scheme*  
761 *dependent. When a scheme uses elements of the common syntax, it will also use the common*  
762 *syntax equivalence rules, namely that the scheme and hostname are case insensitive and a*  
763 *URL with an explicit ":port", where the port is the default for the scheme, is equivalent to one*  
764 *where the port is elided.*

765 To avoid ambiguity resulting from variations in URI encoding SAML applications SHOULD employ  
766 the URI normalized form wherever possible as follows:

- 767 • The assertion issuer SHOULD encode all resource URIs in normalized form.
- 768 • Relying parties SHOULD convert resource URIs to normalized form prior to processing.

769 Inconsistent URI interpretation can also result from differences between the URI syntax and the  
770 semantics of an underlying file system. Particular care is required if URIs are employed to specify  
771 an access control policy language. The following security conditions should be satisfied by the  
772 system which employs SAML assertions:

- 773 • Parts of the URI syntax are case sensitive. If the underlying file system is case insensitive a  
774 requestor SHOULD NOT be able to gain access to a denied resource by changing the case  
775 of a part of the resource URI.
- 776 • Many file systems support mechanisms such as logical paths and symbolic links which  
777 allow users to establish logical equivalences between file system entries. A requestor  
778 SHOULD NOT be able to gain access to a denied resource by creating such an  
779 equivalence.

780 The <AuthorizationDecisionStatement> element is of type  
781 **AuthorizationDecisionStatementType**, which extends **SubjectStatementAbstractType** with the  
782 addition of the following elements (in order) and attributes:

783 **Resource** [Required]

784 A URI identifying the resource to which access authorization is sought.

785 **Decision** [Required]

786 The decision rendered by the issuer with respect to the specified resource. The value is of  
787 the **DecisionType** simple type.

788 **<Actions>** [Required]

789 The set of actions authorized to be performed on the specified resource.

790 **<Evidence>** [Any Number]

791 A set of assertions that the issuer relied on in making the decision.

792 The following schema fragment defines the <AuthorizationDecisionStatement> element  
793 and its **AuthorizationDecisionStatementType** complex type:

```
794 <element name="AuthorizationDecisionStatement"
795 type="saml:AuthorizationDecisionStatementType"/>
```

```

796 <complexType name="AuthorizationDecisionStatementType">
797   <complexContent>
798     <extension base="saml:SubjectStatementAbstractType">
799       <sequence>
800         <element ref="saml:Actions"/>
801         <element ref="saml:Evidence" minOccurs="0"
802           maxOccurs="unbounded"/>
803       </sequence>
804       <attribute name="Resource" type="anyURI" use="required" />
805       <attribute name="Decision" type="saml:DecisionType"
806         use="required"/>
807     </extension>
808   </complexContent>
809 </complexType>

```

#### 810 2.4.4.1. Elements <Actions> and <Action>

811 The <Actions> element specifies the set of actions on the specified resource for which permission  
812 is sought. It has the following element and attribute:

813 Namespace [Optional]

814 A URI representing the namespace in which the names of specified actions are to be  
815 interpreted. If this element is absent, the namespace [http://www.oasis-](http://www.oasis-open.org/committees/security/docs/draft-sstc-core-26#rwdc-negation)  
816 [open.org/committees/security/docs/draft-sstc-core-26#rwdc-negation](http://www.oasis-open.org/committees/security/docs/draft-sstc-core-26#rwdc-negation) specified in section  
817 7.2.2 is in effect.

818 <Action> [One or more]

819 An action sought to be performed on the specified resource.

820 The following schema fragment defines the <Actions> element, its **ActionsType** complex type,  
821 and the <Action> element:

```

822 <element name="Actions" type="saml:ActionsType"/>
823 <complexType name="ActionsType">
824   <sequence>
825     <element ref="saml:Action" maxOccurs="unbounded"/>
826   </sequence>
827   <attribute name="Namespace" type="anyURI" use="optional"/>
828 </complexType>
829 <element name="Action" type="string"/>

```

#### 830 2.4.4.2. Element <Evidence>

831 The <Evidence> element contains an assertion that the issuer relied on in issuing the  
832 authorization decision. It has the **AssertionSpecifierType** complex type.

833 The provision of an assertion as evidence MAY affect the reliance agreement between the  
834 requestor and the Authorization Authority. For example, in the case that the requestor presented an  
835 assertion to the Authorization Authority in a request, the Authorization Authority MAY use that  
836 assertion as evidence in making its response without endorsing the assertion as valid either to the  
837 requestor or any third party.

838 The following schema fragment defines the <Evidence> element:

```

839 <element name="Evidence" type="saml:AssertionSpecifierType"/>

```

#### 840 2.4.5. Element <AttributeStatement>

841 The <AttributeStatement> element supplies a statement by the issuer that the specified  
842 subject is associated with the specified attributes. It is of type **AttributeStatementType**, which  
843 extends **SubjectStatementAbstractType** with the addition of the following element:

844 <Attribute> [One or More]  
845 The <Attribute> element specifies an attribute of the subject.

846 The following schema fragment defines the <AttributeStatement> element and its  
847 **AttributeStatementType** complex type:

```
848 <element name="AttributeStatement" type="saml:AttributeStatementType"/>  
849 <complexType name="AttributeStatementType">  
850 <complexContent>  
851 <extension base="saml:SubjectStatementAbstractType">  
852 <sequence>  
853 <element ref="saml:Attribute" maxOccurs="unbounded"/>  
854 </sequence>  
855 </extension>  
856 </complexContent>  
857 </complexType>
```

#### 858 2.4.5.1. Elements <AttributeDesignator> and <Attribute>

859 The <AttributeDesignator> element identifies an attribute name within an attribute  
860 namespace. It has the **AttributeDesignatorType** complex type. It is used in an attribute assertion  
861 query to request that attribute values within a specific namespace be returned (see 3.3.4 for more  
862 information). The <AttributeDesignator> element contains the following XML attributes:

863 AttributeNamespace [Optional]  
864 The namespace in which the AttributeName elements are interpreted.

865 AttributeName [Optional]  
866 The name of the attribute.

867 The following schema fragment defines the <AttributeDesignator> element and its  
868 **AttributeDesignatorType** complex type:

```
869 <element name="AttributeDesignator" type="saml:AttributeDesignatorType"/>  
870 <complexType name="AttributeDesignatorType">  
871 <attribute name="AttributeName" type="string" use="required"/>  
872 <attribute name="AttributeNamespace" type="anyURI" use="required"/>  
873 </complexType>
```

874 The <Attribute> element supplies the value for an attribute of an assertion subject. It has the  
875 **AttributeType** complex type, which extends **AttributeDesignatorType** with the addition of the  
876 following element:

877 <AttributeValue> [Any Number]  
878 The value of the attribute.

879 The following schema fragment defines the <Attribute> element and its **AttributeType** complex  
880 type:

```
881 <element name="Attribute" type="saml:AttributeType"/>  
882 <complexType name="AttributeType">  
883 <complexContent>  
884 <extension base="saml:AttributeDesignatorType">  
885 <sequence>  
886 <element ref="saml:AttributeValue" maxOccurs="unbounded"/>  
887 </sequence>  
888 </extension>  
889 </complexContent>  
890 </complexType>
```

##### 891 2.4.5.1.1 Element <AttributeValue>

892 The <AttributeValue> element supplies the value of a specified attribute. It is of the **anyType**  
893 simple type, which allows any well-formed XML to appear as the content of the element.

894 If the data content of an AttributeValue element is of a XML Schema simple type (e.g. interger,  
895 string, etc) the data type MAY be declared explicitly by means of an `xsi:type` declaration in the  
896 `<AttributeValue>` element. If the attribute value contains structured data the necessary data  
897 elements may be defined in an extension schema introduced by means of the `xmlns=` mechanism.

898 The following schema fragment defines the `<AttributeValue>` element:

```
899 <element name="AttributeValue" type="anyType"/>
```



## 3. SAML Protocol

900

901 SAML assertions MAY be generated and exchanged using a variety of protocols. The bindings and  
902 profiles specification for SAML [**SAMLBind**] describes specific means of transporting assertions  
903 using existing widely deployed protocols.

904 SAML-aware requestors MAY in addition use the SAML request-response protocol defined by the  
905 <Request> and <Response> elements. The requestor sends a <Request> element to a SAML  
906 authority, and the authority generates a <Response> element, as shown in Figure 2.



907

908

Figure 2: SAML Request-Response Protocol

### 3.1. Schema Header and Namespace Declarations

909

910 The following schema fragment defines the XML namespaces and other header information for the  
911 protocol schema:

```
912 <schema  
913   targetNamespace="http://www.oasis-open.org/committees/security/docs/draft-  
914   sstc-schema-protocol-27.xsd"  
915   xmlns="http://www.w3.org/2001/XMLSchema"  
916   xmlns:samlp="http://www.oasis-open.org/committees/security/docs/draft-sstc-  
917   schema-protocol-27.xsd"  
918   xmlns:saml="http://www.oasis-open.org/committees/security/docs/draft-sstc-  
919   schema-assertion-27.xsd"  
920   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
921   elementFormDefault="unqualified">  
922   <import namespace="http://www.oasis-open.org/committees/security/docs/draft-  
923   sstc-schema-assertion-27.xsd"  
924     schemaLocation="draft-sstc-schema-assertion-27.xsd"/>  
925   <import namespace="http://www.w3.org/2000/09/xmldsig#"  
926     schemaLocation="xmldsig-core-schema.xsd"/>  
927   <annotation>  
928     <documentation>draft-sstc-schema-protocol-27.xsd</documentation>  
929   </annotation>  
930   ...  
931 </schema>
```

932

### 3.2. Requests

933

934 The following sections define the SAML constructs that contain request information.

#### 3.2.1. Complex Type RequestAbstractType

935

936 All SAML requests are of types that are derived from the abstract **RequestAbstractType** complex  
937 type. This type defines common attributes and elements that are associated with all SAML  
938 requests:

939 RequestID [Required]

940

941

942

An identifier for the request. It is of type **IDType**, and MUST follow the requirements specified by that type for identifier uniqueness. The values of the RequestID attribute in a request and the InResponseTo attribute in the corresponding response MUST match.



943 MajorVersion [Required]  
 944 The major version of this request. The identifier for the version of SAML defined in this  
 945 specification is 1. Processing of this attribute is specified in Section 3.4.2.

946 MinorVersion [Required]  
 947 The minor version of this request. The identifier for the version of SAML defined in this  
 948 specification is 0. Processing of this attribute is specified in Section 3.4.2.

949 IssueInstant [Required]  
 950 The time instant of issue of the request. The time value is encoded in UTC as described in  
 951 section 1.2.1.

952 <RespondWith> [Any Number]  
 953 Each <RespondWith> element specifies a type of response that is acceptable to the  
 954 requestor.

955 <Signature> [Optional]  
 956 An XML Signature that authenticates the assertion, see section 5.

957 The following schema fragment defines the **RequestAbstractType** complex type:

```

958 <complexType name="RequestAbstractType" abstract="true">
959   <sequence>
960     <element ref="samlp:RespondWith"
961       minOccurs="0" maxOccurs="unbounded"/>
962     <element ref="ds:Signature" minOccurs="0"/>
963   </sequence>
964   <attribute name="RequestID" type="saml:IDType" use="required"/>
965   <attribute name="MajorVersion" type="integer" use="required"/>
966   <attribute name="MinorVersion" type="integer" use="required"/>
967   <attribute name="IssueInstant" type="dateTime" use="required"/>
968 </complexType>

```

### 969 3.2.1.1. Element <RespondWith>

970 The <RespondWith> element specifies a type of response that is acceptable to the requestor. If  
 971 no <RespondWith> element is specified the default is SingleStatement.

972 The <RespondWith> element specifies the type(s) of response that is acceptable to the requestor.  
 973 Multiple <RespondWith> elements MAY be specified to indicate that the requestor is capable of  
 974 processing multiple requests.

975 <RespondWith> elements are used to inform the responder of the type of assertion statements  
 976 that the requestor is capable of processing. The Responder MUST use this information to ensure  
 977 that it generates responses consistent with information found in the <RespondWith> element of  
 978 the Request.

979 NOTE: Inability to find assertions that meet <RespondWith> criteria should be treated identical to  
 980 any other query for which no assertions are available. In both cases a status of success would  
 981 normally be returned in the Response message, but no assertions to be found therein.

982 <RespondWith> element values are URIs. A requestor MAY use an XML schema identifier as a  
 983 <RespondWith> element value to inform the responder that the specified SAML extension schema  
 984 is supported. <RespondWith> values defined in this document are specified as URI fragment  
 985 identifiers, the nominal base for these identifier values being the SAML protocol schema identifier  
 986 URI.

987 Acceptable values for the <RespondWith> element are:

988 #SingleStatement  
 989 An assertion carrying exactly one statement element.

- 990 #MultipleStatement
- 991     An assertion carrying at least one statement element.
- 992 #AuthenticationStatement
- 993     An assertion carrying an Authentication statement.
- 994 #AuthorizationDecisionStatement
- 995     An assertion carrying an Authorization Decision statement.
- 996 #AttributeStatement
- 997     An assertion carrying an Attribute statement.
- 998 *Schema URI*
- 999     An assertion containing additional elements from the specified schema.

1000 The following schema fragment defines the <RespondWith> element:

```
1001 <element name="RespondWith" type="anyURI"/>
```

### 1002 3.2.2. Element <Request>

1003 The <Request> element specifies a SAML request. It provides either a query or a request for a  
 1004 specific assertion identified by <AssertionIDReference> or <AssertionArtifact>. It has  
 1005 the complex type **RequestType**, which extends **RequestAbstractType** by adding a choice of one  
 1006 of the following elements:

- 1007 <Query>
- 1008     An extension point that allows extension schemas to define new types of query.
- 1009 <SubjectQuery>
- 1010     An extension point that allows extension schemas to define new types of query that specify
- 1011     a single SAML subject.
- 1012 <AuthenticationQuery>
- 1013     Makes a query for authentication information.
- 1014 <AttributeQuery>
- 1015     Makes a query for attribute information.
- 1016 <AuthorizationDecisionQuery>
- 1017     Makes a query for an authorization decision.
- 1018 <AssertionIDReference> [One or more]
- 1019     Requests assertions by reference to its assertion identifier.
- 1020 <AssertionArtifact> [One or more]
- 1021     Requests assertions by supplying an assertion artifact that represents it.

1022 The following schema fragment defines the <Request> element and its **RequestType** complex  
 1023 type:

```
1024 <element name="Request" type="samlp:RequestType"/>
1025 <complexType name="RequestType">
1026   <complexContent>
1027     <extension base="samlp:RequestAbstractType">
1028       <choice>
1029         <element ref="samlp:Query"/>
1030         <element ref="samlp:SubjectQuery"/>
1031         <element ref="samlp:AuthenticationQuery"/>
1032         <element ref="samlp:AttributeQuery"/>
1033         <element ref="samlp:AuthorizationDecisionQuery"/>
1034         <element ref="saml:AssertionIDReference" maxOccurs="unbounded"/>
1035         <element ref="samlp:AssertionArtifact" maxOccurs="unbounded"/>
1036       </choice>
1037     </extension>
1038   </complexContent>
```

1039 `</complexType>`

### 1040 **3.2.3. Element <AssertionArtifact>**

1041 The <AssertionArtifact> element is used to specify the assertion artifact that represents an  
1042 assertion.

1043 The following schema fragment defines the <AssertionArtifact> element:

```
1044 <element name="AssertionArtifact" type="string"/>
```

## 1045 **3.3. Queries**

1046 The following sections define the SAML constructs that contain query information.

### 1047 **3.3.1. Element <Query>**

1048 The <Query> element is an extension point that allows new SAML queries to be defined. Its  
1049 **QueryAbstractType** is abstract; extension elements MUST use the `xsi:type` attribute to indicate  
1050 the derived type. **QueryAbstractType** is the base type from which all SAML query elements are  
1051 derived.

1052 The following schema fragment defines the <Query> element and its **QueryAbstractType**  
1053 complex type:

```
1054 <element name="Query" type="saml:QueryAbstractType"/>  
1055 <complexType name="QueryAbstractType" abstract="true"/>
```

### 1056 **3.3.2. Element <SubjectQuery>**

1057 The <SubjectQuery> element is an extension point that allows new SAML queries that specify a  
1058 single SAML subject. Its **SubjectQueryAbstractType** complex type is abstract; extension elements  
1059 MUST use the `xsi:type` attribute to indicate the derived type. **SubjectQueryAbstractType** adds  
1060 the <Subject> element.

1061 The following schema fragment defines the <SubjectQuery> element and its  
1062 **SubjectQueryAbstractType** complex type:

```
1063 <element name="SubjectQuery" type="saml:SubjectQueryAbstractType"/>  
1064 <complexType name="SubjectQueryAbstractType" abstract="true">  
1065   <complexContent>  
1066     <extension base="saml:QueryAbstractType">  
1067       <sequence>  
1068         <element ref="saml:Subject"/>  
1069       </sequence>  
1070     </extension>  
1071   </complexContent>  
1072 </complexType>
```

### 1073 **3.3.3. Element <AuthenticationQuery>**

1074 The <AuthenticationQuery> element is used to make the query “What authentication  
1075 assertions are available for this subject?” A successful response will be in the form of assertions  
1076 containing authentication statements. This element is of type **AuthenticationQueryType**, which  
1077 extends **SubjectQueryAbstractType** with the addition of the following element:

1078 <ConfirmationMethod> [Optional]

1079 A filter for possible responses. If it is present, the query made is “What authentication  
1080 assertions do you have for this subject with the supplied confirmation method?”

1081 In response to an authentication query, a responder returns assertions with authentication  
1082 statements as follows: The <Subject> element in the returned assertions MUST be identical to  
1083 the <Subject> element of the query. If the <ConfirmationMethod> element is present in the  
1084 query, at least one <ConfirmationMethod> element in the response MUST match. It is  
1085 OPTIONAL for the complete set of all such matching assertions to be returned in the response.

1086 The following schema fragment defines the <AuthenticationQuery> type and its  
1087 **AuthenticationQueryType** complex type:

```
1088 <element name="AuthenticationQuery" type="samlp:AuthenticationQueryType"/>  
1089 <complexType name="AuthenticationQueryType">  
1090 <complexContent>  
1091 <extension base="samlp:SubjectQueryAbstractType">  
1092 <sequence>  
1093 <element ref="saml:ConfirmationMethod" minOccurs="0"/>  
1094 </sequence>  
1095 </extension>  
1096 </complexContent>  
1097 </complexType>
```

### 1098 3.3.4. Element <AttributeQuery>

1099 The <AttributeQuery> element is used to make the query "Return the requested attributes for  
1100 this subject." A successful response will be in the form of assertions containing attribute statements.  
1101 This element is of type **AttributeQueryType**, which extends **SubjectQueryAbstractType** with the  
1102 addition of the following element and attribute:

1103 Resource [Optional]

1104 The Resource attribute if present specifies that the attribute query is made in response to a  
1105 specific authorization decision relating to the resource. The responder MAY use the  
1106 resource attribute to establish the scope of the request.

1107 If the resource attribute is specified and the responder does not wish to support resource-  
1108 specific attribute queries, or if the resource value provided is invalid or unrecognized, then it  
1109 SHOULD respond with a SAML status of "Error.Receiver.ResourceNotRecognized".

1110 <AttributeDesignator> [Any Number] (see Section 2.4.5.1)

1111 Each <AttributeDesignator> element specifies an attribute whose value is to be  
1112 returned. If no attributes are specified, the list of desired attributes is implicit and  
1113 application-specific.

1114 The following schema fragment defines the <AttributeQuery> element and its  
1115 **AttributeQueryType** complex type:

```
1116 <element name="AttributeQuery" type="samlp:AttributeQueryType"/>  
1117 <complexType name="AttributeQueryType">  
1118 <complexContent>  
1119 <extension base="samlp:SubjectQueryAbstractType">  
1120 <sequence>  
1121 <element ref="saml:AttributeDesignator"  
1122 minOccurs="0" maxOccurs="unbounded"/>  
1123 </sequence>  
1124 <attribute name="Resource" type="anyURI" use="optional"/>  
1125 </extension>  
1126 </complexContent>  
1127 </complexType>
```

### 1128 3.3.5. Element <AuthorizationDecisionQuery>

1129 The <AuthorizationDecisionQuery> element is used to make the query "Should these  
1130 actions on this resource be allowed for this subject, given this evidence?" A successful response  
1131 will be in the form of assertions containing authorization decision statements. This element is of

1132 type **AuthorizationDecisionQueryType**, which extends **SubjectQueryAbstractType** with the  
1133 addition of the following elements and attribute:

1134 Resource [Required]  
1135 A URI indicating the resource for which authorization is requested.

1136 <Actions> [Required]  
1137 The actions for which authorization is requested.

1138 <Evidence> [Any Number]  
1139 An assertion that the responder MAY rely on in making its response.

1140 The following schema fragment defines the <AuthorizationDecisionQuery> element and its  
1141 **AuthorizationDecisionQueryType** complex type:

```
1142 <element name="AuthorizationDecisionQuery"  
1143 type="samlp:AuthorizationDecisionQueryType"/>  
1144 <complexType name="AuthorizationDecisionQueryType">  
1145 <complexContent>  
1146 <extension base="samlp:SubjectQueryAbstractType">  
1147 <sequence>  
1148 <element ref="saml:Actions"/>  
1149 <element ref="saml:Evidence"  
1150 minOccurs="0" maxOccurs="unbounded"/>  
1151 </sequence>  
1152 <attribute name="Resource" type="anyURI" use="required"/>  
1153 </extension>  
1154 </complexContent>  
1155 </complexType>
```

## 1156 3.4. Responses

1157 The following sections define the SAML constructs that contain response information.

### 1158 3.4.1. Complex Type ResponseAbstractType

1159 All SAML responses are of types that are derived from the abstract **ResponseAbstractType**  
1160 complex type. This type defines common attributes and elements that are associated with all SAML  
1161 responses:

1162 ResponseID [Required]  
1163 An identifier for the response. It is of type **IDType**, and MUST follow the requirements  
1164 specified by that type for identifier uniqueness.

1165 InResponseTo [Required]  
1166 A reference to the identifier of the request to which the response corresponds. The value of  
1167 this attribute MUST match the value of the corresponding RequestID attribute.

1168 MajorVersion [Required]  
1169 The major version of this response. The identifier for the version of SAML defined in this  
1170 specification is 1. Processing of this attribute is specified in Section 3.4.4.

1171 MinorVersion [Required]  
1172 The minor version of this response. The identifier for the version of SAML defined in this  
1173 specification is 0. Processing of this attribute is specified in Section 3.4.4.

1174 IssueInstant [Optional]  
1175 The time instant of issue of the request. The time value is encoded in UTC as described in  
1176 section 1.2.1.

1177 <Signature> [Optional]  
1178 An XML Signature that authenticates the assertion, see section 5.

1179 The following schema fragment defines the **ResponseAbstractType** complex type:

```
1180 <complexType name="ResponseAbstractType" abstract="true">
1181 <sequence>
1182 <element ref="ds:Signature" minOccurs="0"/>
1183 </sequence>
1184 <attribute name="ResponseID" type="saml:IDType" use="required"/>
1185 <attribute name="InResponseTo" type="saml:IDReferenceType"
1186 use="required"/>
1187 <attribute name="MajorVersion" type="integer" use="required"/>
1188 <attribute name="MinorVersion" type="integer" use="required"/>
1189 <attribute name="IssueInstant" type="dateTime" use="required"/>
1190 </complexType>
```

### 1191 3.4.2. Element <Response>

1192 The <Response> element specifies the status of the corresponding SAML request and a list of  
1193 zero or more assertions that answer the request. It has the complex type **ResponseType**, which  
1194 extends **ResponseAbstractType** by adding the following elements (in an unbounded mixture):

1195 <Status> [Required] (see Section 3.4.3)

1196 A code representing the status of the corresponding request.

1197 <Assertion> [Any Number] (see Section 2.3.3)

1198 Specifies an assertion by value.

1199 The following schema fragment defines the <Response> element and its **ResponseType** complex  
1200 type:

```
1201 <element name="Response" type="samlp:ResponseType"/>
1202 <complexType name="ResponseType">
1203 <complexContent>
1204 <extension base="samlp:ResponseAbstractType">
1205 <sequence>
1206 <element ref="samlp:Status"/>
1207 <element ref="saml:Assertion"
1208 minOccurs="0" maxOccurs="unbounded"/>
1209 </sequence>
1210 </extension>
1211 </complexContent>
1212 </complexType>
```

### 1213 3.4.3. Element <Status>

1214 The <Status> element :

1215 <StatusCode> [Required]

1216 A code representing the status of the corresponding request.

1217 <StatusMessage> [Any Number]

1218 A message which MAY be returned to an operator.

1219 <StatusDetail> [Optional]

1220 Specifies additional information concerning an error condition.

1221 The following schema fragment defines the <Status> element and its **StatusType** complex type:

```
1222 <element name="Status" type="samlp:StatusType"/>
1223 <complexType name="StatusType">
1224 <sequence>
1225 <element ref="samlp:StatusCode"/>
1226 <element ref="samlp:StatusMessage"
1227 minOccurs="0" maxOccurs="unbounded"/>
1228 <element ref="samlp:StatusDetail" minOccurs="0"/>
```

1229       </sequence>  
1230       </complexType>

### 1231   **3.4.3.1. Element <StatusCode>**

1232   The <StatusCode> element specifies a code representing the status of the corresponding request  
1233   and an option sub code providing more specific information concerning a particular error status:

1234   Value [Required]

1235       The status code value as defined below.

1236   <SubStatusCode> [Optional]

1237       An optional subordinate status code value that provides more specific information on an  
1238       error condition.

1239   The following **StatusCode** values are defined:

1240   Success

1241       The request succeeded.

1242   VersionMismatch

1243       The receiver could not process the request because the version was incorrect.

1244   Receiver

1245       The request could not be performed due to an error at the receiving end.

1246   Sender

1247       The request could not be performed due to an error in the sender or in the request

1248   The following schema fragment defines the <StatusCode> element and its **StatusCodeType**  
1249   complex type and the **StatusCodeEnumType** simple type:

```
1250   <element name="StatusCode" type="samlp:StatusCodeType"/>  
1251   <complexType name="StatusCodeType">  
1252     <sequence>  
1253       <element ref="samlp:SubStatusCode" minOccurs="0"/>  
1254     </sequence>  
1255     <attribute name="Value" type="samlp:StatusCodeEnumType" use="required"/>  
1256   </complexType>  
1257   <simpleType name="StatusCodeEnumType">  
1258     <restriction base="QName">  
1259       <enumeration value="samlp:Success"/>  
1260       <enumeration value="samlp:VersionMismatch"/>  
1261       <enumeration value="samlp:Receiver"/>  
1262       <enumeration value="samlp:Sender"/>  
1263     </restriction>  
1264   </simpleType>
```

### 1265   **3.4.3.2. Element <SubStatusCode>**

1266   The <SubStatusCode> element specifies an additional code representing the status of the  
1267   corresponding request:

1268   Value [Required]

1269       The status code value as defined below.

1270   <SubStatusCode> [Optional]

1271       An optional subordinate status code value that provides an additional level of specific  
1272       information on an error condition.

1273   The following **SubStatusCode** values are defined, additional codes MAY be defined in future  
1274   versions of the SAML specification:



1275 RequestVersionTooHigh  
 1276       The protocol version specified in the request is a major upgrade from the highest protocol  
 1277       version supported by the responder.

1278 RequestVersionTooLow  
 1279       The responder cannot respond to the particular request using the SAML version specified  
 1280       in the request because it is too low.

1281 RequestVersionDeprecated  
 1282       The responder does not respond to any requests with the protocol version specified in the  
 1283       request.

1284 TooManyResponses  
 1285       The response would contain more elements than the responder will return.

1286 The following schema fragment defines the <SubStatusCode> element and its  
 1287 **SubStatusCodeType** complex type:

```
1288 <element name="SubStatusCode" type="samlp:SubStatusCodeType"/>
1289 <complexType name="SubStatusCodeType">
1290   <sequence>
1291     <element ref="samlp:SubStatusCode" minOccurs="0"/>
1292   </sequence>
1293   <attribute name="Value" type="QName" use="required"/>
1294 </complexType>
```

### 1295 3.4.3.3. Element <StatusMessage>

1296 The <StatusMessage> element specifies a message that MAY be returned to an operator:

1297 The following schema fragment defines the <StatusMessage> element and its  
 1298 **StatusMessageType** complex type:

```
1299 <element name="StatusMessage" type="string"/>
```

### 1300 3.4.3.4. Element <StatusDetail>

1301 The <StatusDetail> element MAY be used to specify additional information concerning an error  
 1302 condition.

1303 The following schema fragment defines the <StatusDetail> element and its **StatusDetailType**  
 1304 complex type:

```
1305 <element name="StatusDetail" type="samlp:StatusDetailType"/>
1306 <complexType name="StatusDetailType">
1307   <sequence>
1308     <any namespace="##any"
1309       processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
1310   </sequence>
1311 </complexType>
```

## 1312 3.4.4. Responses to <AuthenticationQuery> and <AttributeQuery>

1313 Responses to Authentication and Attribute queries are constructed by matching against the  
 1314 <saml:Subject> element found within the <AuthenticationQuery> or <AttributeQuery>  
 1315 elements. In response to these queries, every assertion returned by a SAML responder MUST  
 1316 contain at least one statement whose <saml:Subject> element **strongly matches** the  
 1317 <saml:Subject> element found in the query.

1318 A <saml:Subject> element S1 strongly matches S2 if and only if:

- 1319 1 If S2 includes a <saml:NameIdentifier> element, then S1 must include an identical  
 1320 <saml:NameIdentifier> element.



1321 2 If S2 includes a `<saml:SubjectConfirmation>` element, then S1 must include an  
1322 identical `<saml:SubjectConfirmation>` element.

## 4. SAML Versioning

1323

1324 SAML version information appears in the following elements:

- 1325 • <Assertion>
- 1326 • <Request>
- 1327 • <Response>

1328 The version numbering of the SAML assertion is independent of the version number of the SAML  
1329 request-response protocol. The version information for each consists of a major version number  
1330 and a minor version number, both of which are integers. In accordance with industry practice a  
1331 version number SHOULD be presented to the user in the form *Major.Minor*. This document defines  
1332 SAML Assertions 1.0 and SAML Protocol 1.0.

1333 The version number  $Major_B.Minor_B$  is higher than the version number  $Major_A.Minor_A$  if and only if:

1334  $Major_B > Major_A \vee ( ( Major_B = Major_A ) \wedge Minor_B > Minor_A )$

1335 Each revision of SAML SHALL assign version numbers to assertions, requests, and responses that  
1336 are the same as or higher than the corresponding version number in the SAML version that  
1337 immediately preceded it.

1338 New versions of SAML SHALL assign new version numbers as follows:

- 1339 • **Documentation change:**  $( Major_B = Major_A ) \wedge ( Minor_B > Minor_A )$   
1340 If the major and minor version numbers are unchanged, the new version *B* only introduces  
1341 changes to the documentation that raise no compatibility issues with an implementation of  
1342 version *A*.
- 1343 • **Minor upgrade:**  $( Major_B = Major_A ) \wedge ( Minor_B > Minor_A )$   
1344 If the major version number of versions *A* and *B* are the same and the minor version  
1345 number of *B* is higher than that of *A*, the new SAML version MAY introduce changes to the  
1346 SAML schema and semantics but any changes that are introduced in *B* SHALL be  
1347 compatible with version *A*.
- 1348 • **Major upgrade:**  $Major_B > Major_A$   
1349 If the major version of *B* number is higher than the major version of *A*, Version *B* MAY  
1350 introduce changes to the SAML schema and semantics that are incompatible with *A*.

### 4.1. Assertion Version

1351

1352 A SAML application MUST NOT issue any assertion whose version number is not supported.

1353 A SAML application MUST reject any assertion whose major version number is not supported.

1354 A SAML application MAY reject any assertion whose version number is higher than the highest  
1355 supported version.

### 4.2. Request Version

1356

1357 A SAML application SHOULD issue requests that specify the highest SAML version supported by  
1358 both the sender and recipient.

1359 If the SAML application does not know the capabilities of the recipient it should assume that it  
1360 supports the highest SAML version supported by the sender.

### 1361 **4.3. Response Version**

1362 A SAML application MUST NOT issue responses that specify a higher SAML version number than  
1363 the corresponding request.

1364 A SAML application MUST NOT issue a response that has a major version number that is lower  
1365 than the major version number of the corresponding request except to report the error  
1366 `RequestVersionTooHigh`.

1367 Incompatible protocol versions MAY cause the following errors to be reported:

1368 `RequestVersionTooHigh`

1369       The protocol version specified in the request is a major upgrade from the highest protocol  
1370       version supported by the responder.

1371 `RequestVersionTooLow`

1372       The responder cannot respond to the particular request using the SAML version specified  
1373       in the request because it is too low.

1374 `RequestVersionDeprecated`

1375       The responder does not respond to any requests with the protocol version specified in the  
1376       request.

1377 **5. SAML & XML-Signature Syntax and**  
1378 **Processing**

1379 SAML Assertions, Request and Response messages may be signed, with the following benefits:

- 1380
- 1381 • An Assertion signed by the issuer (AP). This supports :
    - 1382 (1) Message integrity
    - 1383 (2) Authentication of the issuer to a relying party
    - 1384 (3) If the signature is based on the issuer's public-private key pair, then it also provides for non-repudiation of origin.
  - 1385 • A SAML request or a SAML response message signed by the message originator. This supports :
    - 1386 (1) Message integrity
    - 1387 (2) Authentication of message origin to a destination
    - 1388 (3) If the signature is based on the originator's public-private key pair, then it also provides for non-repudiation of origin.
- 1389
- 1390

1391 Note :

- 1392 • SAML documents may be the subject of signatures from different packaging contexts. **[XMLSig]** provides a framework for signing in XML and is the framework of choice. However, signing may also take place in the context of S/MIME or Java objects that contain SAML documents. One goal is to ensure compatibility with this type of "foreign" digital signing.
- 1393
- 1394
- 1395
- 1396
- 1397 • It is useful to characterize situations when a digital signature is NOT required in SAML.

1398 Assertions:

1399 The asserting party has provided the assertion to the relying party, authenticated by means  
1400 other than digital signature and the channel is secure. In other words, the RP has obtained the  
1401 assertion from the AP directly (no intermediaries) through a secure channel and the AP has  
1402 authenticated to the RP.

1403 Request/Response messages:

1404 The originator has authenticated to the destination and the destination has obtained the  
1405 assertion directly from the originator (no intermediaries) through secure channel(s).

1406 Many different techniques are available for "direct" authentication and secure channel between  
1407 two parties. The list includes SSL, HMAC, password-based login etc. Also the security  
1408 requirement depends on the communicating applications and the nature of the assertion  
1409 transported.

1410 All other contexts require the use of digital signature for assertions and request and response  
1411 messages. Specifically:

- 1412 (1) An assertion obtained by a relying party from an entity other than the asserting party MUST  
1413 be signed by the issuer.
- 1414 (2) A SAML message arriving at a destination from an entity other than the originating site  
1415 MUST be signed by the origin site.

1416 **5.1. Signing Assertions**

1417 All SAML assertions MAY be signed using the XML Signature. This is reflected in the assertion  
1418 schema – Section 2.3.

## 1419 **5.2. Request/Response Signing**

1420 All SAML requests and responses MAY be signed using the XML Signature. This is reflected in the  
1421 schema – Section 3.2 & 3.4.

## 1422 **5.3. Signature Inheritance**

### 1423 **5.3.1. Rationale**

1424 SAML assertions may be embedded within request or response messages or other XML  
1425 messages, which may be signed. Request or response messages may themselves be contained  
1426 within other messages that are based on other XML messaging frameworks (e.g., SOAP) and the  
1427 composite object may be the subject of a signature. Another possibility is that SAML assertions or  
1428 request/response messages are embedded within a non-XML messaging object (e.g., MIME  
1429 package) and signed.

1430 In such a case, the SAML sub-message (Assertion, request, response) may be viewed as inheriting  
1431 a signature from the "super-signature" over the enclosing object, provided certain constraints are  
1432 met.

1433 (1) An assertion may be viewed as inheriting a signature from a super signature, if the super  
1434 signature applies all the elements within the assertion.

1435 A SAML request or response may be viewed as inheriting a signature from a super signature, if the  
1436 super signature applies to all of the elements within the response.

### 1437 **5.3.2. Rules for SAML Signature Inheritance**

1438 Signature inheritance occurs when SAML message (assertion/request/response) is not signed but  
1439 is enclosed within signed SAML such that the signature applies to all of the elements within the  
1440 message. In such a case, the SAML message is said to inherit the signature and may be  
1441 considered equivalent to the case where it is explicitly signed. The SAML message inherits the  
1442 "closest enclosing signature".

1443 But if SAML messages need to be passed around by themselves, or embedded in other messages,  
1444 they would need to be signed as per section 5.1

## 1445 **5.4. XML Signature Profile**

1446 The XML Signature [**XMLSig**] specification calls out a general XML syntax for signing data with  
1447 many flexibilities and choices. This section details the constraints on these facilities so that SAML  
1448 processors do not have to deal with the full generality of XML Signature processing.

### 1449 **5.4.1. Signing formats**

1450 XML Signature has three ways of representing signature in a document viz: enveloping, enveloped  
1451 and detached.

1452 SAML assertions and protocols MUST use the enveloped signatures for signing assertions and  
1453 protocols. SAML processors should support use of RSA signing and verification for public key  
1454 operations.

### 1455 **5.4.2. CanonicalizationMethod**

1456 XML Signature REQUIRES the Canonical XML (omits comments)  
1457 (<http://www.w3.org/TR/2001/REC-xml-c14n-20010315>). SAML implementations SHOULD use  
1458 Canonical XML with no comments.

1459 **5.4.3. Transforms**

1460 [XMLSig] REQUIRES the enveloped signature transform  
1461 <http://www.w3.org/2000/09/xmlsig#enveloped-signature>

1462 **5.4.4. KeyInfo**

1463 SAML does not restrict or impose any restrictions in this area. Therefore following [XMLSig]  
1464 keyInfo may be absent.

1465 **5.4.5. Binding between statements in a multi-statement assertion**

1466 Use of signing does not affect semantics of statements within assertions in any way, as stated in  
1467 this document Sections 1 through 4.

## 1468 **6. SAML Extensions**

1469 The SAML schemas support extensibility. An example of an application that extends SAML  
1470 assertions is the XTAML system for management of embedded trust roots [XTAML]. The following  
1471 sections explain how to use the extensibility features in SAML to create extension schemas.

1472 Note that elements in the SAML schemas are not blocked from substitution, so that all SAML  
1473 elements MAY serve as the head element of a substitution group. Also, types are not defined as  
1474 *final*, so that all SAML types MAY be extended and restricted. The following sections discuss  
1475 only elements that have been specifically designed to support extensibility.

### 1476 **6.1. Assertion Schema Extension**

1477 The SAML assertion schema is designed to permit separate processing of the assertion package  
1478 and the statements it contains, if the extension mechanism is used for either part.

1479 The following elements are intended specifically for use as extension points in an extension  
1480 schema; their types are set to *abstract*, so that the use of an `xsi:type` attribute with these  
1481 elements is REQUIRED:

- 1482 • `<Assertion>`
- 1483 • `<Condition>`
- 1484 • `<Statement>`
- 1485 • `<SubjectStatement>`
- 1486 • `<AdviceElement>`

1487 In addition, the following elements that are directly usable as part of SAML MAY be extended:

- 1488 • `<AuthenticationStatement>`
- 1489 • `<AuthorizationDecisionStatement>`
- 1490 • `<AttributeStatement>`
- 1491 • `<AudienceRestrictionCondition>`

1492 Finally, the following elements are defined to allow elements from arbitrary namespaces within  
1493 them, which serves as a built-in extension point without requiring an extension schema:

- 1494 • `<AttributeValue>`
- 1495 • `<Advice>`

### 1496 **6.2. Protocol Schema Extension**

1497 The following elements are intended specifically for use as extension points in an extension  
1498 schema; their types are set to *abstract*, so that the use of an `xsi:type` attribute with these  
1499 elements is REQUIRED:

- 1500 • `<Query>`
- 1501 • `<SubjectQuery>`

1502 In addition, the following elements that are directly usable as part of SAML MAY be extended:

- 1503 • `<Request>`

- 1504 • <AuthenticationQuery>
- 1505 • <AuthorizationDecisionQuery>
- 1506 • <AttributeQuery>
- 1507 • <Response>

### 1508 6.3. Use of Type Derivation and Substitution Groups

1509 W3C XML Schema [**Schema1**] provides two principal mechanisms for specifying an element of an  
1510 extended type: type derivation and substitution groups.

1511 For example, a <Statement> element can be assigned the type **NewStatementType** by means of  
1512 the `xsi:type` attribute. For such an element to be schema-valid, **NewStatementType** needs to be  
1513 derived from **StatementType**. The following example of a SAML assertion assumes that the  
1514 extension schema (represented by the `new:` prefix) has defined this new type:

```
1515 <saml:Assertion ...>  
1516   <saml:Statement xsi:type="new:NewStatementType">  
1517     ...  
1518   </saml:Statement>  
1519 </saml:Assertion>
```

1520 Alternatively, the extension schema can define a <NewStatement> element that is a member of a  
1521 substitution group that has <Statement> as a head element. For the substituted element to be  
1522 schema-valid, it needs to have a type that matches or is derived from the head element's type. The  
1523 following is an example of an extension schema fragment that defines this new element:

```
1524 <xsd:element "NewStatement" type="new:NewStatementType"  
1525   substitutionGroup="saml:Statement"/>
```

1526 The substitution group declaration allows the <NewStatement> element to be used anywhere the  
1527 SAML <Statement> element can be used. The following is an example of a SAML assertion that  
1528 uses the extension element:

```
1529 <saml:Assertion ...>  
1530   <new:NewStatement>  
1531     ...  
1532   </new:NewStatement>  
1533 </saml:Assertion>
```

1534 The choice of extension method has no effect on the semantics of the XML document but does  
1535 have implications for interoperability.

1536 The advantages of type derivation are as follows:

- 1537 • A document can be more fully interpreted by a parser that does not have access to the  
1538 extension schema because a "native" SAML element is available.
- 1539 • At the time of writing, some W3C XML Schema validators do not support substitution  
1540 groups, whereas the `xsi:type` attribute is widely supported.

1541 The advantage of substitution groups is that a document can be explained without the need to  
1542 explain the functioning of the `xsi:type` attribute.



## 1543 **7. SAML-Defined Identifiers**

1544 The following sections define URI-based identifiers for common authentication protocols and  
1545 actions.

1546 Where possible an existing URN is used to specify a protocol. In the case of IETF protocols the  
1547 URN of the most current RFC that specifies the protocol is used. URIs created specifically for  
1548 SAML have the initial stem:

1549 <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-27>

### 1550 **7.1. Confirmation Method Identifiers**

1551 The following identifiers MAY be used in the <ConfirmationMethod> element (see Section  
1552 2.4.2.3) to refer to common authentication protocols.

#### 1553 **7.1.1. SAML Artifact:**

1554 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-26#artifact>

1555 <SubjectConfirmationData>: *Base64 ( Artifact )*

1556 The subject of the assertion is the party that can present the SAML Artifact value specified in  
1557 <SubjectConfirmationData>.

#### 1558 **7.1.2. SAML Artifact (SHA-1):**

1559 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-26#artifact-sha1>

1560 <SubjectConfirmationData>: *Base64 ( SHA1 ( Artifact ) )*

1561 The subject of the assertion is the party that can present a SAML Artifact such that the SHA1 digest  
1562 of the specified artifact matches the value specified in <SubjectConfirmationData>.

#### 1563 **7.1.3. Holder of Key:**

1564 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-26#Holder-Of-Key>

1565 <ds:KeyInfo>: Any cryptographic key

1566 The subject of the assertion is the party that can demonstrate that it is the holder of the private  
1567 component of the key specified in <ds:KeyInfo>.

#### 1568 **7.1.4. Sender Vouches:**

1569 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-26#sender-vouches>

1570 Indicates that no other information is available about the context of use of the assertion. The  
1571 Relying party SHOULD utilize other means to determine if it should process the assertion further.

#### 1572 **7.1.5. Password (Pass-Through):**

1573 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-26#password>

1574 <SubjectConfirmationData>: *Base64 ( Password )*

1575 The subject of the assertion is the party that can present the password value specified in  
1576 <SubjectConfirmationData>.  
1577 The username of the subject is specified by means of the <NameIdentifier> element.

### 1578 **7.1.6. Password (One-Way-Function SHA-1):**

1579 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-26#password-sha1>

1580 <SubjectConfirmationData>: *Base64 ( SHA1 ( Password )*)

1581 The subject of the assertion is the party that can present the password such that the SHA1 digest of  
1582 the specified password matches the value specified in <SubjectConfirmationData>.

1583 The username of the subject is specified by means of the <NameIdentifier> element.

### 1584 **7.1.7. Kerberos**

1585 **URI:** urn:ietf:rfc:1510

1586 <SubjectConfirmationData>: A Kerberos Ticket

1587 The subject is authenticated by means of the Kerberos protocol **[RFC 1510]**, an instantiation of the  
1588 Needham-Schroeder symmetric key authentication mechanism **[Needham78]**.

### 1589 **7.1.8. SSL/TLS Certificate Based Client Authentication:**

1590 **URI:** urn:ietf:rfc:2246

1591 <ds:KeyInfo>: Any cryptographic key

### 1592 **7.1.9. Object Authenticator (SHA-1):**

1593 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-26#object-sha1>

1594 <SubjectConfirmationData>: *Base64 ( SHA1 ( Object )*)

1595 This authenticator element is the result of computing a digest, using the SHA-1 hash algorithm. It is  
1596 used when the subject can be represented as a binary string, for example when it is an XML  
1597 document or the disk image of executable code. Any preprocessing of the subject prior to  
1598 computation of the digest is out of scope. The name of the subject should be conveyed in an  
1599 accompanying NameIdentifier element.

### 1600 **7.1.10. PKCS#7**

1601 **URI:** urn:ietf:rfc:2315

1602 <SubjectConfirmationData>: *Base64 ( PKCS#7 ( Object )*)

1603 This authenticator element is signed data in PKCS#7 format **[PKCS#7]**. The posited identity of the  
1604 signer must be conveyed in an accompanying NameIdentifier element. This subject type may be  
1605 included in the subject field of an authentication query, in which case the corresponding response  
1606 indicates whether the posited signer is, indeed, the signer. It may be included in an attribute query,  
1607 in which case, the requested attribute values for the subject authenticated by the signed data are  
1608 returned. It may be included in an authorization query, in which case, the access request  
1609 represented by the signed data shall be identified by the accompanying object element, and the

1610 corresponding authorization decision assertion indicates whether the signer is authorized for the  
1611 access request represented by the object element.

### 1612 **7.1.11. Cryptographic Message Syntax**

1613 **URI:** urn:ietf:rfc:2630

1614 <SubjectConfirmationData>: *Base64* ( CMS ( *Object* ) )

1615 This authenticator element is signed data in CMS format [CMS]. See also 7.1.10

### 1616 **7.1.12. XML Digital Signature**

1617 **URI:** urn:ietf:rfc:3075

1618 <SubjectConfirmationData>: *Base64* ( XML-SIG ( *Object* ) )

1619 <ds:KeyInfo>: A cryptographic signing key

1620 This authenticator element is signed data in XML Signature format. See also 7.1.10

## 1621 **7.2. Action Namespace Identifiers**

1622 The following identifiers MAY be used in the `ActionNamespace` attribute (see Section 2.4.4.1) to  
1623 refer to common sets of actions to perform on resources.

### 1624 **7.2.1. Read/Write/Execute/Delete/Control:**

1625 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-26#rwedc>

1626 Defined actions:

1627 `Read Write Execute Delete Control`

1628 These actions are interpreted in the normal manner, i.e.

1629 `Read`

1630 `The subject may read the resource`

1631 `Write`

1632 `The subject may modify the resource`

1633 `Execute`

1634 `The subject may execute the resource`

1635 `Delete`

1636 `The subject may delete the resource`

1637 `Control`

1638 `The subject may specify the access control policy for the resource`

### 1639 **7.2.2. Read/Write/Execute/Delete/Control with Negation:**

1640 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-26#rwedc-negation>

1641 Defined actions:

1642 `Read Write Execute Delete Control ~Read ~Write ~Execute ~Delete ~Control`

1643 The actions specified in section 7.2.1 are interpreted in the same manner described there. Actions  
1644 prefixed with a tilde ~ are negated permissions and are used to affirmatively specify that the stated

1645 permission is denied. Thus a subject described as being authorized to perform the action `~Read` is  
1646 affirmatively denied read permission.

1647 An application MUST NOT authorize both an action and its negated form.

### 1648 **7.2.3. Get/Head/Put/Post:**

1649 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-26#ghpp>

1650 Defined actions:

1651 GET HEAD PUT POST

1652 These actions bind to the corresponding HTTP operations. For example a subject authorized to  
1653 perform the GET action on a resource is authorized to retrieve it.

1654 The GET and HEAD actions loosely correspond to the conventional read permission and the PUT  
1655 and POST actions to the write permission. The correspondence is not exact however since a HTTP  
1656 GET operation may cause data to be modified and a POST operation may cause modification to a  
1657 resource other than the one specified in the request. For this reason a separate Action URI  
1658 specifier is provided.

### 1659 **7.2.4. UNIX File Permissions:**

1660 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-26#unix>

1661 The defined actions are the set of UNIX file access permissions expressed in the numeric (octal)  
1662 notation.

1663 The action string is a four digit numeric code:

1664 *extended user group world*

1665 Where the *extended* access permission has the value

1666 +2 if sgid is set

1667 +4 if suid is set

1668 The *user group* and *world* access permissions have the value

1669 +1 if execute permission is granted

1670 +2 if write permission is granted

1671 +4 if read permission is granted

1672 For example `0754` denotes the UNIX file access permission: user read, write and execute, group  
1673 read and execute and world read.

## 8. SAML Schema Listings

1674

1675 The following sections contain complete listings of the assertion and protocol schemas for SAML.

### 8.1. Assertion Schema

1676

1677 Following is a complete listing of the SAML assertion schema [SAML-XSD].

```
1678 <?xml version="1.0" encoding="UTF-8"?>
1679 <!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Phill Hallam-Baker
1680 (VeriSign Inc.) -->
1681 <schema
1682     targetNamespace="http://www.oasis-open.org/committees/security/docs/draft-
1683     sstc-schema-assertion-27.xsd"
1684     xmlns="http://www.w3.org/2001/XMLSchema" xmlns:saml="http://www.oasis-
1685     open.org/committees/security/docs/draft-sstc-schema-assertion-27.xsd"
1686     xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
1687     elementFormDefault="unqualified">
1688     <import namespace="http://www.w3.org/2000/09/xmldsig#"
1689         schemaLocation="xmldsig-core-schema.xsd"/>
1690     <annotation>
1691         <documentation>draft-sstc-schema-assertion-27.xsd</documentation>
1692     </annotation>
1693     <simpleType name="IDType">
1694         <restriction base="string"/>
1695     </simpleType>
1696     <simpleType name="IDReferenceType">
1697         <restriction base="string"/>
1698     </simpleType>
1699     <simpleType name="DecisionType">
1700         <restriction base="string">
1701             <enumeration value="Permit"/>
1702             <enumeration value="Deny"/>
1703             <enumeration value="Indeterminate"/>
1704         </restriction>
1705     </simpleType>
1706     <element name="AssertionSpecifier" type="saml:AssertionSpecifierType"/>
1707     <complexType name="AssertionSpecifierType">
1708         <choice>
1709             <element ref="saml:AssertionIDReference"/>
1710             <element ref="saml:Assertion"/>
1711         </choice>
1712     </complexType>
1713     <element name="AssertionIDReference" type="saml:IDReferenceType"/>
1714     <element name="Assertion" type="saml:AssertionType"/>
1715     <complexType name="AssertionType">
1716         <sequence>
1717             <element ref="saml:Conditions" minOccurs="0"/>
1718             <element ref="saml:Advice" minOccurs="0"/>
1719             <choice maxOccurs="unbounded">
1720                 <element ref="saml:Statement"/>
1721                 <element ref="saml:SubjectStatement"/>
1722                 <element ref="saml:AuthenticationStatement"/>
1723                 <element ref="saml:AuthorizationDecisionStatement"/>
1724                 <element ref="saml:AttributeStatement"/>
1725             </choice>
1726             <element ref="ds:Signature" minOccurs="0"/>
1727         </sequence>
1728         <attribute name="MajorVersion" type="integer" use="required"/>
1729         <attribute name="MinorVersion" type="integer" use="required"/>
1730         <attribute name="AssertionID" type="saml:IDType" use="required"/>

```

```

1731     <attribute name="Issuer" type="string" use="required"/>
1732     <attribute name="IssueInstant" type="dateTime" use="required"/>
1733 </complexType>
1734 <element name="Conditions" type="saml:ConditionsType"/>
1735 <complexType name="ConditionsType">
1736     <choice minOccurs="0" maxOccurs="unbounded">
1737         <element ref="saml:Condition"/>
1738         <element ref="saml:AudienceRestrictionCondition"/>
1739     </choice>
1740     <attribute name="NotBefore" type="dateTime" use="optional"/>
1741     <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
1742 </complexType>
1743 <element name="Condition" type="saml:ConditionAbstractType"/>
1744 <complexType name="ConditionAbstractType" abstract="true"/>
1745 <element name="AudienceRestrictionCondition"
1746     type="saml:AudienceRestrictionConditionType"/>
1747 <complexType name="AudienceRestrictionConditionType">
1748     <complexContent>
1749         <extension base="saml:ConditionAbstractType">
1750             <sequence>
1751                 <element ref="saml:Audience" maxOccurs="unbounded"/>
1752             </sequence>
1753         </extension>
1754     </complexContent>
1755 </complexType>
1756 <element name="Audience" type="anyURI"/>
1757 <element name="TargetRestrictionCondition"
1758     type="saml:TargetRestrictionConditionType"/>
1759 <complexType name="TargetRestrictionConditionType">
1760     <complexContent>
1761         <extension base="saml:ConditionAbstractType">
1762             <sequence>
1763                 <element ref="saml:Target"
1764                     minOccurs="1" maxOccurs="unbounded"/>
1765             </sequence>
1766         </extension>
1767     </complexContent>
1768 </complexType>
1769 <element name="Target" type="anyURI"/>
1770 <element name="Advice" type="saml:AdviceType"/>
1771 <complexType name="AdviceType">
1772     <choice minOccurs="0" maxOccurs="unbounded">
1773         <element ref="saml:AssertionSpecifier"/>
1774         <element ref="saml:AdviceElement"/>
1775         <any namespace="##other" processContents="lax"/>
1776     </choice>
1777 </complexType>
1778 </complexType>
1779 <complexType name="AdviceElement" type="saml:AdviceAbstractType"/>
1780 <complexType name="AdviceAbstractType"/>
1781 <element name="Statement" type="saml:StatementAbstractType"/>
1782 <complexType name="StatementAbstractType" abstract="true"/>
1783 <element name="SubjectStatement" type="saml:SubjectStatementAbstractType"/>
1784 <complexType name="SubjectStatementAbstractType" abstract="true">
1785     <complexContent>
1786         <extension base="saml:StatementAbstractType">
1787             <sequence>
1788                 <element ref="saml:Subject"/>
1789             </sequence>
1790         </extension>
1791     </complexContent>
1792 </complexType>
1793 </complexType>

```

```

1794 <element name="Subject" type="saml:SubjectType"/>
1795 <complexType name="SubjectType">
1796   <choice>
1797     <sequence>
1798       <element ref="saml:NameIdentifier"/>
1799       <element ref="saml:SubjectConfirmation" minOccurs="0"/>
1800     </sequence>
1801     <element ref="saml:SubjectConfirmation"/>
1802   </choice>
1803 </complexType>
1804 <element name="NameIdentifier" type="saml:NameIdentifierType"/>
1805 <complexType name="NameIdentifierType">
1806   <attribute name="SecurityDomain" type="string"/>
1807   <attribute name="Name" type="string" use="required"/>
1808 </complexType>
1809 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>
1810 <complexType name="SubjectConfirmationType">
1811   <sequence>
1812     <element ref="saml:ConfirmationMethod" maxOccurs="unbounded"/>
1813     <element ref="saml:SubjectConfirmationData" minOccurs="0"/>
1814     <element ref="ds:KeyInfo" minOccurs="0"/>
1815   </sequence>
1816 </complexType>
1817 <element name="SubjectConfirmationData" type="string"/>
1818 <element name="ConfirmationMethod" type="anyURI"/>
1819 <element name="AuthenticationStatement"
1820   type="saml:AuthenticationStatementType"/>
1821 <complexType name="AuthenticationStatementType">
1822   <complexContent>
1823     <extension base="saml:SubjectStatementAbstractType">
1824       <sequence>
1825         <element ref="saml:AuthenticationLocality" minOccurs="0"/>
1826         <element ref="saml:AuthorityBinding"
1827           minOccurs="0" maxOccurs="unbounded"/>
1828       </sequence>
1829       <attribute name="AuthenticationMethod" type="anyURI"/>
1830       <attribute name="AuthenticationInstant" type="dateTime"/>
1831     </extension>
1832   </complexContent>
1833 </complexType>
1834 <element name="AuthenticationLocality"
1835   type="saml:AuthenticationLocalityType"/>
1836 <complexType name="AuthenticationLocalityType">
1837   <attribute name="IPAddress" type="string" use="optional"/>
1838   <attribute name="DNSAddress" type="string" use="optional"/>
1839 </complexType>
1840 <element name="AuthorityBinding" type="saml:AuthorityBindingType"/>
1841 <complexType name="AuthorityBindingType">
1842   <attribute name="AuthorityKind" type="saml:AuthorityKindType"
1843     use="required"/>
1844   <attribute name="Location" type="anyURI" use="required"/>
1845   <attribute name="Binding" type="anyURI" use="required"/>
1846 </complexType>
1847 <simpleType name="AuthorityKindType">
1848   <restriction base="string">
1849     <enumeration value="authentication"/>
1850     <enumeration value="attribute"/>
1851     <enumeration value="authorization"/>
1852   </restriction>
1853 </simpleType>
1854 <element name="AuthorizationDecisionStatement"
1855   type="saml:AuthorizationDecisionStatementType"/>
1856 <complexType name="AuthorizationDecisionStatementType">

```

```

1857     <complexContent>
1858         <extension base="saml:SubjectStatementAbstractType">
1859             <sequence>
1860                 <element ref="saml:Actions"/>
1861                 <element ref="saml:Evidence"
1862                     minOccurs="0" maxOccurs="unbounded"/>
1863             </sequence>
1864             <attribute name="Resource" type="anyURI" use="required"/>
1865             <attribute name="Decision"
1866                 type="saml:DecisionType" use="required"/>
1867         </extension>
1868     </complexContent>
1869 </complexType>
1870 <element name="Actions" type="saml:ActionsType"/>
1871 <complexType name="ActionsType">
1872     <sequence>
1873         <element ref="saml:Action" maxOccurs="unbounded"/>
1874     </sequence>
1875     <attribute name="Namespace" type="anyURI" use="optional"/>
1876 </complexType>
1877 <element name="Action" type="string"/>
1878 <element name="Evidence" type="saml:AssertionSpecifierType"/>
1879 <element name="AttributeStatement" type="saml:AttributeStatementType"/>
1880 <complexType name="AttributeStatementType">
1881     <complexContent>
1882         <extension base="saml:SubjectStatementAbstractType">
1883             <sequence>
1884                 <element ref="saml:Attribute" maxOccurs="unbounded"/>
1885             </sequence>
1886         </extension>
1887     </complexContent>
1888 </complexType>
1889 <element name="AttributeDesignator" type="saml:AttributeDesignatorType"/>
1890 <complexType name="AttributeDesignatorType">
1891     <attribute name="AttributeName" type="string" use="required"/>
1892     <attribute name="AttributeNamespace" type="anyURI" use="required"/>
1893 </complexType>
1894 <element name="Attribute" type="saml:AttributeType"/>
1895 <complexType name="AttributeType">
1896     <complexContent>
1897         <extension base="saml:AttributeDesignatorType">
1898             <sequence>
1899                 <element ref="saml:AttributeValue" maxOccurs="unbounded"/>
1900             </sequence>
1901         </extension>
1902     </complexContent>
1903 </complexType>
1904 <element name="AttributeValue" type="saml:anyType"/>
1905 </schema>

```

## 8.2. Protocol Schema

1906

1907 Following is a complete listing of the SAML protocol schema [SAML-XML].

1908

```

1909 <?xml version="1.0" encoding="UTF-8"?>
1910 <!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Phill Hallam-Baker
1911 (VeriSign Inc.) -->
1912 <schema
1913     targetNamespace="http://www.oasis-open.org/committees/security/docs/draft-
1914     sstc-schema-protocol-27.xsd"
1915     xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
1916     xmlns:saml="http://www.oasis-open.org/committees/security/docs/draft-sstc-
1917     schema-assertion-27.xsd"

```

1916



```

1917     xmlns:samlp="http://www.oasis-open.org/committees/security/docs/draft-sstc-
1918 schema-protocol-27.xsd"
1919     xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="unqualified">
1920     <import
1921       namespace="http://www.oasis-open.org/committees/security/docs/draft-sstc-
1922 schema-assertion-27.xsd"
1923       schemaLocation="draft-sstc-schema-assertion-27.xsd"/>
1924     <import namespace="http://www.w3.org/2000/09/xmlsig#"
1925       schemaLocation="xmlsig-core-schema.xsd"/>
1926     <annotation>
1927       <documentation>draft-sstc-schema-protocol-27.xsd</documentation>
1928     </annotation>
1929     <complexType name="RequestAbstractType" abstract="true">
1930       <sequence>
1931         <element ref="samlp:RespondWith"
1932           minOccurs="0" maxOccurs="unbounded"/>
1933         <element ref="ds:Signature" minOccurs="0"/>
1934       </sequence>
1935       <attribute name="RequestID" type="saml:IDType" use="required"/>
1936       <attribute name="MajorVersion" type="integer" use="required"/>
1937       <attribute name="MinorVersion" type="integer" use="required"/>
1938       <attribute name="IssueInstant" type="dateTime" use="required"/>
1939     </complexType>
1940     <element name="RespondWith" type="anyURI"/>
1941     <element name="Request" type="samlp:RequestType"/>
1942     <complexType name="RequestType">
1943       <complexContent>
1944         <extension base="samlp:RequestAbstractType">
1945           <choice>
1946             <element ref="samlp:Query"/>
1947             <element ref="samlp:SubjectQuery"/>
1948             <element ref="samlp:AuthenticationQuery"/>
1949             <element ref="samlp:AttributeQuery"/>
1950             <element ref="samlp:AuthorizationDecisionQuery"/>
1951             <element ref="saml:AssertionID" maxOccurs="unbounded"/>
1952             <element ref="samlp:AssertionArtifact" maxOccurs="unbounded"/>
1953           </choice>
1954         </extension>
1955       </complexContent>
1956     </complexType>
1957     <element name="AssertionArtifact" type="string"/>
1958     <element name="Query" type="samlp:QueryAbstractType"/>
1959     <complexType name="QueryAbstractType" abstract="true">
1960     <element name="SubjectQuery" type="samlp:SubjectQueryAbstractType"/>
1961     <complexType name="SubjectQueryAbstractType" abstract="true">
1962       <complexContent>
1963         <extension base="samlp:QueryAbstractType">
1964           <sequence>
1965             <element ref="saml:Subject"/>
1966           </sequence>
1967         </extension>
1968       </complexContent>
1969     </complexType>
1970     <element name="AuthenticationQuery" type="samlp:AuthenticationQueryType"/>
1971     <complexType name="AuthenticationQueryType">
1972       <complexContent>
1973         <extension base="samlp:SubjectQueryAbstractType">
1974           <sequence>
1975             <element ref="saml:ConfirmationMethod" minOccurs="0"/>
1976           </sequence>
1977         </extension>
1978       </complexContent>
1979     </complexType>

```

```

1980 <element name="AttributeQuery" type="saml:AttributeQueryType"/>
1981 <complexType name="AttributeQueryType">
1982   <complexContent>
1983     <extension base="saml:SubjectQueryAbstractType">
1984       <sequence>
1985         <element ref="saml:AttributeDesignator"
1986           minOccurs="0" maxOccurs="unbounded"/>
1987       </sequence>
1988       <attribute name="Resource" type="anyURI" use="optional"/>
1989     </extension>
1990   </complexContent>
1991 </complexType>
1992 <element name="AuthorizationDecisionQuery"
1993   type="saml:AuthorizationDecisionQueryType"/>
1994 <complexType name="AuthorizationDecisionQueryType">
1995   <complexContent>
1996     <extension base="saml:SubjectQueryAbstractType">
1997       <sequence>
1998         <element ref="saml:Actions"/>
1999         <element ref="saml:Evidence"
2000           minOccurs="0" maxOccurs="unbounded"/>
2001       </sequence>
2002       <attribute name="Resource" type="anyURI" use="required"/>
2003     </extension>
2004   </complexContent>
2005 </complexType>
2006 <complexType name="ResponseAbstractType" abstract="true">
2007   <sequence>
2008     <element ref="ds:Signature" minOccurs="0"/>
2009   </sequence>
2010   <attribute name="ResponseID" type="saml:IDType" use="required"/>
2011   <attribute name="InResponseTo" type="saml:IDReferenceType"
2012     use="required"/>
2013   <attribute name="MajorVersion" type="integer" use="required"/>
2014   <attribute name="MinorVersion" type="integer" use="required"/>
2015   <attribute name="IssueInstant" type="dateTime" use="required"/>
2016 </complexType>
2017
2018 <element name="Response" type="saml:ResponseType"/>
2019 <complexType name="ResponseType">
2020   <complexContent>
2021     <extension base="saml:ResponseAbstractType">
2022       <sequence>
2023         <element ref="saml:Status"/>
2024         <element ref="saml:Assertion"
2025           minOccurs="0" maxOccurs="unbounded"/>
2026       </sequence>
2027     </extension>
2028   </complexContent>
2029 </complexType>
2030 <element name="Status" type="saml:StatusType"/>
2031 <complexType name="StatusType">
2032   <sequence>
2033     <element ref="saml:StatusCode"/>
2034     <element ref="saml:StatusMessage"
2035       minOccurs="0" maxOccurs="unbounded"/>
2036     <element ref="saml:StatusDetail" minOccurs="0"/>
2037   </sequence>
2038 </complexType>
2039 <element name="StatusCode" type="saml:StatusCodeType"/>
2040 <complexType name="StatusCodeType">
2041   <sequence>
2042     <element ref="saml:SubStatusCode" minOccurs="0"/>

```

```
2043     </sequence>
2044     <attribute name="Value" type="samlp:StatusCodeEnumType" use="required"/>
2045 </complexType>
2046 <simpleType name="StatusCodeEnumType">
2047   <restriction base="QName">
2048     <enumeration value="samlp:Success"/>
2049     <enumeration value="samlp:VersionMismatch"/>
2050     <enumeration value="samlp:Receiver"/>
2051     <enumeration value="samlp:Sender"/>
2052   </restriction>
2053 </simpleType>
2054 <element name="SubStatusCode" type="samlp:SubStatusCodeType"/>
2055 <complexType name="SubStatusCodeType">
2056   <sequence>
2057     <element ref="samlp:SubStatusCode" minOccurs="0"/>
2058   </sequence>
2059   <attribute name="Value" type="QName" use="required"/>
2060 </complexType>
2061 <element name="StatusMessage" type="string"/>
2062 <element name="StatusDetail" type="samlp:StatusDetailType"/>
2063 <complexType name="StatusDetailType">
2064   <sequence>
2065     <any namespace="##any"
2066         processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
2067   </sequence>
2068 </complexType>
2069 </schema>
2070
```

## 9. References

2071

- 2072 [Needham78] R. Needham et al., *Using Encryption for Authentication in Large Networks*  
2073 *of Computers*, Communications of the ACM, Vol. 21 (12), pp. 993-999,  
2074 December 1978.
- 2075 [Kern-84] B. Kernighan, Rob Pike *The UNIX Programming Environment*, (March  
2076 1984) Prentice Hall Computer Books;
- 2077 [PKCS1] B. Kaliski, *PKCS #1: RSA Encryption Version 2.0*, RSA Laboratories, also  
2078 IETF RFC 2437, October 1998. <http://www.ietf.org/rfc/rfc2437.txt>
- 2079 [PKCS7] B. Kaliski., "PKCS #7: Cryptographic Message Syntax, Version 1.5.", RFC  
2080 2315, March 1998.
- 2081 [RFC 1510] J. Kohl, C. Neuman. *The Kerberos Network Authentication Requestor (V5)*.  
2082 September 1993. <http://www.ietf.org/rfc/rfc1510.txt>
- 2083 [RFC 2246] T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. January 1999.  
2084 <http://www.ietf.org/rfc/rfc2246.txt>
- 2085 [RFC 2630] R. Housley. Cryptographic Message Syntax. June 1999.  
2086 <http://www.ietf.org/rfc/rfc630.txt>
- 2087 [RFC 2648] R. Moats. *A URN Namespace for IETF Documents*. August 1999.  
2088 <http://www.ietf.org/rfc/rfc2648.txt>
- 2089 [RFC 3075] D. Eastlake, J. Reagle, D. Solo. XML-Signature Syntax and Processing.  
2090 March 2001. <http://www.ietf.org/rfc/rfc3075.txt>
- 2091 [RFC2104] H. Krawczyk et al., *HMAC: Keyed Hashing for Message Authentication*,  
2092 <http://www.ietf.org/rfc/rfc2104.txt>, IETF RFC 2104, February 1997.
- 2093 [RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,  
2094 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997
- 2095 [SAMLBind] P. Mishra et al., *Bindings and Profiles for the OASIS Security Assertion*  
2096 *Markup Language (SAML)*, [http://www.oasis-](http://www.oasis-open.org/committees/security/docs/draft-sstc-bindings-model-07.pdf)  
2097 [open.org/committees/security/docs/draft-sstc-bindings-model-07.pdf](http://www.oasis-open.org/committees/security/docs/draft-sstc-bindings-model-07.pdf),  
2098 OASIS, December 2001.
- 2099 [SAMLConform] **TBS**
- 2100 [SAMLGloss] J. Hodges et al., *Glossary for the OASIS Security Assertion Markup*  
2101 *Language (SAML)*, [http://www.oasis-](http://www.oasis-open.org/committees/security/docs/draft-sstc-glossary-02.pdf)  
2102 [open.org/committees/security/docs/draft-sstc-glossary-02.pdf](http://www.oasis-open.org/committees/security/docs/draft-sstc-glossary-02.pdf), OASIS,  
2103 December 2001.
- 2104 [SAMPL-XSD] P. Hallam-Baker et al., *SAML protocol schema*, [http://www.oasis-](http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-protocol-21.xsd)  
2105 [open.org/committees/security/docs/draft-sstc-schema-protocol-21.xsd](http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-protocol-21.xsd),  
2106 OASIS, December 2001.
- 2107 [SAMLSecure] **TBS**
- 2108 [SAML-XSD] P. Hallam-Baker et al., *SAML assertion schema*, [http://www.oasis-](http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-assertion-21.xsd)  
2109 [open.org/committees/security/docs/draft-sstc-schema-assertion-21.xsd](http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-assertion-21.xsd),  
2110 OASIS, December 2001.
- 2111 [Schema1] H. S. Thompson et al., *XML Schema Part 1: Structures*,  
2112 <http://www.w3.org/TR/xmlschema-1/>, World Wide Web Consortium  
2113 Recommendation, May 2001.
- 2114 [Schema2] P. V. Biron et al., *XML Schema Part 2: Datatypes*,  
2115 <http://www.w3.org/TR/xmlschema-2/>, World Wide Web Consortium  
2116 Recommendation, May 2001.
- 2117 [XMLEnc] *XML Encryption Specification*, In development.

|      |                      |   |
|------|----------------------|---|
| 2118 | <b>[XMLSig]</b>      | D. Eastlake et al., <i>XML-Signature Syntax and Processing</i> ,<br><a href="http://www.w3.org/TR/xmlsig-core/">http://www.w3.org/TR/xmlsig-core/</a> , World Wide Web Consortium.                    |
| 2119 |                      |   |
| 2120 | <b>[XMLSig-XSD]</b>  | XML Signature Schema available from <a href="http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd">http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd</a> . |
| 2121 |                      |   |
| 2122 | <b>[XTAML]</b>       | P. Hallam-Baker, <i>XML Trust Axiom Markup Language 1.0</i> ,<br><a href="http://www.xmltrustcenter.org/">http://www.xmltrustcenter.org/</a> , VeriSign Inc. September 2001.                          |
| 2123 |                      |   |
| 2124 | <b>[W3C-CHAR]</b>    | <a href="http://www.w3.org/TR/WD-charreq">http://www.w3.org/TR/WD-charreq</a>   |
| 2125 | <b>[UNICODE-C]</b>   | <a href="http://www.unicode.org/unicode/reports/tr15/tr15-21.html">http://www.unicode.org/unicode/reports/tr15/tr15-21.html</a>   |
| 2126 | <b>[W3C-CharMod]</b> | <a href="http://www.w3.org/TR/charmod/">http://www.w3.org/TR/charmod/</a>   |
| 2127 | <b>[XML]</b>         | <a href="http://www.w3.org/TR/REC-xml">http://www.w3.org/TR/REC-xml</a>   |
| 2128 | <b>[RFC 2396]</b>    | <a href="http://www.ietf.org/rfc/rfc2396.txt?">http://www.ietf.org/rfc/rfc2396.txt?</a>   |

2129

## Appendix A. Notices

2130 OASIS takes no position regarding the validity or scope of any intellectual property or other rights  
2131 that might be claimed to pertain to the implementation or use of the technology described in this  
2132 document or the extent to which any license under such rights might or might not be available;  
2133 neither does it represent that it has made any effort to identify any such rights. Information on  
2134 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS  
2135 website. Copies of claims of rights made available for publication and any assurances of licenses to  
2136 be made available, or the result of an attempt made to obtain a general license or permission for  
2137 the use of such proprietary rights by implementors or users of this specification, can be obtained  
2138 from the OASIS Executive Director.

2139 OASIS invites any interested party to bring to its attention any copyrights, patents or patent  
2140 applications, or other proprietary rights which may cover technology that may be required to  
2141 implement this specification. Please address the information to the OASIS Executive Director.

2142 Copyright © The Organization for the Advancement of Structured Information Standards [OASIS]  
2143 2001. All Rights Reserved.

2144 This document and translations of it may be copied and furnished to others, and derivative works  
2145 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,  
2146 published and distributed, in whole or in part, without restriction of any kind, provided that the above  
2147 copyright notice and this paragraph are included on all such copies and derivative works. However,  
2148 this document itself may not be modified in any way, such as by removing the copyright notice or  
2149 references to OASIS, except as needed for the purpose of developing OASIS specifications, in  
2150 which case the procedures for copyrights defined in the OASIS Intellectual Property Rights  
2151 document must be followed, or as required to translate it into languages other than English.

2152 The limited permissions granted above are perpetual and will not be revoked by OASIS or its  
2153 successors or assigns.

2154 This document and the information contained herein is provided on an "AS IS" basis and OASIS  
2155 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO  
2156 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY  
2157 RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A  
2158 PARTICULAR PURPOSE.

Page: 6

[PHB1] This will need to be edited before final release

Page: 10

[PHB2] Update with final name spaces

Page: 10

[PHB3] schema location bug