

1

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

2      **Document identifier:** draft-sstc-core-28

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

4      **Publication date:** March 15th 2002~~March 7th 2002~~

5      **Maturity Level:** Committee Working Draft

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

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

8      **Editors:**

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

11     **Contributors:**

12     ~~Carlisle Adams, Entrust~~

13     ~~Scott Cantor, The Ohio State University~~

14     ~~Marc Chanliau, Netegrity~~

15     ~~Nigel Edwards, Hewlett-Packard~~

16     ~~Marlena Erdos, Tivoli~~

17     ~~Stephen Farrell, Baltimore Technologies~~

18     ~~Simon Godik, Crosslogic~~

19     ~~Jeff Hedges, Oblix~~

20     ~~Charles Knouse, Oblix~~

21     ~~Hal Lockhart, Entegrity Solutions~~

22     ~~Chris McLaren, Netegrity~~

23     ~~Prateek Mishra, Netegrity~~

24     ~~RL "Bob" Morgan, University of Washington~~

25     ~~Tim Moses, Entrust~~

26     ~~David Orchard, BEA~~

27     ~~Joe Pato, Hewlett Packard~~

28     ~~Darren Platt, RSA Security~~

29     ~~Irving Reid, Baltimore Technologies~~

30     ~~Krishna Sankar, Cisco Systems Inc~~

31

32

33

34

35

35		
36	<b><u>ASSERTIONS AND PROTOCOL FOR THE OASIS SECURITY ASSERTION MARKUP</u></b>	
37	<b><u>LANGUAGE (SAML)</u></b>	<b><u>1</u></b>
38	<b><u>1. INTRODUCTION</u></b>	<b><u>10</u></b>
39	<b><u>1.1. NOTATION</u></b>	<b><u>10</u></b>
40	<b><u>1.2. SCHEMA ORGANIZATION AND NAMESPACES</u></b>	<b><u>10</u></b>
41	<b><u>    1.2.1. Time Values.</u></b>	<b><u>11</u></b>
42	<b><u>    1.2.2. Comparing SAML values</u></b>	<b><u>11</u></b>
43	<b><u>    1.3. SAML CONCEPTS (NON-NORMATIVE)</u></b>	<b><u>11</u></b>
44	<b><u>        1.3.1. Overview</u></b>	<b><u>11</u></b>
45	<b><u>        1.3.2. SAML and URI-Based Identifiers</u></b>	<b><u>13</u></b>
46	<b><u>        1.3.3. SAML and Extensibility</u></b>	<b><u>13</u></b>
47	<b><u>2. SAML ASSERTIONS</u></b>	<b><u>14</u></b>
48	<b><u>    2.1. SCHEMA HEADER AND NAMESPACE DECLARATIONS</u></b>	<b><u>14</u></b>
49	<b><u>    2.2. SIMPLE TYPES</u></b>	<b><u>14</u></b>
50	<b><u>        2.2.1. Simple Types IDType and IDReferenceType</u></b>	<b><u>14</u></b>
51	<b><u>        2.2.2. Simple Type DecisionType</u></b>	<b><u>15</u></b>
52	<b><u>    2.3. ASSERTIONS</u></b>	<b><u>15</u></b>
53	<b><u>        2.3.1. Element &lt;AssertionSpecifier&gt;</u></b>	<b><u>15</u></b>
54	<b><u>        2.3.2. Element &lt;AssertionID&gt;</u></b>	<b><u>16</u></b>
55	<b><u>        2.3.3. Element &lt;Assertion&gt;</u></b>	<b><u>16</u></b>
56	<b><u>            2.3.3.1. Element &lt;Conditions&gt;</u></b>	<b><u>17</u></b>
57	<b><u>                2.3.3.1.1 Attributes NotBefore and NotOnOrAfter</u></b>	<b><u>18</u></b>
58	<b><u>                2.3.3.1.2 Element &lt;Condition&gt;</u></b>	<b><u>18</u></b>
59	<b><u>                2.3.3.1.3 Elements &lt;AudienceRestrictionCondition&gt; and &lt;Audience&gt;</u></b>	<b><u>18</u></b>
60	<b><u>                2.3.3.1.4 Elements &lt;TargetRestrictionCondition&gt; and &lt;Target&gt;</u></b>	<b><u>19</u></b>
61	<b><u>            2.3.3.2. Elements &lt;Advice&gt; and &lt;AdviceElement&gt;</u></b>	<b><u>19</u></b>
62	<b><u>    2.4. STATEMENTS</u></b>	<b><u>20</u></b>

63	<b><u>2.4.1. Element &lt;Statement&gt;</u></b>	20
64	<b><u>2.4.2. Element &lt;SubjectStatement&gt;</u></b>	20
65	<b><u>    2.4.2.1. Element &lt;Subject&gt;</u></b>	21
66	<b><u>    2.4.2.2. Element &lt;NameIdentifier&gt;</u></b>	21
67	<b><u>    2.4.2.3. Elements &lt;SubjectConfirmation&gt;, &lt;ConfirmationMethod&gt;, and &lt;SubjectConfirmationData&gt;</u></b>	22
68	<b><u>2.4.3. Element &lt;AuthenticationStatement&gt;</u></b>	23
69	<b><u>    2.4.3.1. Element &lt;AuthenticationLocality&gt;</u></b>	23
70	<b><u>    2.4.3.2. Element &lt;AuthorityBinding&gt;</u></b>	24
71	<b><u>2.4.4. Element &lt;AuthorizationDecisionStatement&gt;</u></b>	25
72	<b><u>    2.4.4.1. Elements &lt;Actions&gt; and &lt;Action&gt;</u></b>	26
73	<b><u>    2.4.4.2. Element &lt;Evidence&gt;</u></b>	27
74	<b><u>2.4.5. Element &lt;AttributeStatement&gt;</u></b>	27
75	<b><u>    2.4.5.1. Elements &lt;AttributeDesignator&gt; and &lt;Attribute&gt;</u></b>	27
76	<b><u>        2.4.5.1.1. Element &lt;AttributeValue&gt;</u></b>	28
77	<b><u>3. SAML PROTOCOL</u></b>	29
78	<b><u>3.1. SCHEMA HEADER AND NAMESPACE DECLARATIONS</u></b>	29
79	<b><u>3.2. REQUESTS</u></b>	29
80	<b><u>    3.2.1. Complex Type RequestAbstractType</u></b>	29
81	<b><u>        3.2.1.1. Element &lt;RespondWith&gt;</u></b>	30
82	<b><u>    3.2.2. Element &lt;Request&gt;</u></b>	31
83	<b><u>    3.2.3. Element &lt;AssertionArtifact&gt;</u></b>	32
84	<b><u>3.3. QUERIES</u></b>	32
85	<b><u>    3.3.1. Element &lt;Query&gt;</u></b>	32
86	<b><u>    3.3.2. Element &lt;SubjectQuery&gt;</u></b>	32
87	<b><u>    3.3.3. Element &lt;AuthenticationQuery&gt;</u></b>	33
88	<b><u>    3.3.4. Element &lt;AttributeQuery&gt;</u></b>	33
89	<b><u>    3.3.5. Element &lt;AuthorizationDecisionQuery&gt;</u></b>	34
90	<b><u>3.4. RESPONSES</u></b>	34
91	<b><u>    3.4.1. Complex Type ResponseAbstractType</u></b>	34
92	<b><u>    3.4.2. Element &lt;Response&gt;</u></b>	35

93	<b><u>3.4.3. Element &lt;Status&gt;</u></b>	36
94	<u>    3.4.3.1. Element &lt;StatusCode&gt;</u>	36
95	<u>    3.4.3.2. Element &lt;SubStatusCode&gt;</u>	37
96	<u>    3.4.3.3. Element &lt;StatusMessage&gt;</u>	37
97	<u>    3.4.3.4. Element &lt;StatusDetail&gt;</u>	38
98	<b><u>3.4.4. Responses to &lt;AuthenticationQuery&gt; and &lt;AttributeQuery&gt;</u></b>	38
99	<b><u>4. SAML VERSIONING</u></b>	38
100	<u>    4.1. ASSERTION VERSION</u>	39
101	<u>    4.2. REQUEST VERSION</u>	39
102	<u>    4.3. RESPONSE VERSION</u>	40
103	<b><u>5. SAML &amp; XML-SIGNATURE SYNTAX AND PROCESSING</u></b>	41
104	<u>    5.1. SIGNING ASSERTIONS</u>	41
105	<u>    5.2. REQUEST /RESPONSE SIGNING</u>	42
106	<u>    5.3. SIGNATURE INHERITANCE</u>	42
107	<u>        5.3.1. Rationale</u>	42
108	<u>        5.3.2. Rules for SAML Signature Inheritance</u>	42
109	<u>    5.4. XML SIGNATURE PROFILE</u>	42
110	<u>        5.4.1. Signing formats</u>	42
111	<u>        5.4.2. CanonicalizationMethod</u>	42
112	<u>        5.4.3. Transforms</u>	43
113	<u>        5.4.4. KeyInfo</u>	43
114	<u>        5.4.5. Binding between statements in a multi-statement assertion</u>	43
115	<b><u>6. SAML EXTENSIONS</u></b>	44
116	<u>    6.1. ASSERTION SCHEMA EXTENSION</u>	44
117	<u>    6.2. PROTOCOL SCHEMA EXTENSION</u>	44
118	<u>    6.3. USE OF TYPE DERIVATION AND SUBSTITUTION GROUPS</u>	45

119	<b><u>7. SAML-DEFINED IDENTIFIERS</u></b>	<b>46</b>
120	<b><u>7.1. CONFIRMATION METHOD IDENTIFIERS</u></b>	<b>46</b>
121	<b><u>    7.1.1. SAML Artifact:</u></b>	<b>46</b>
122	<b><u>    7.1.2. SAML Artifact (SHA-1):</u></b>	<b>47</b>
123	<b><u>    7.1.3. Holder of Key:</u></b>	<b>47</b>
124	<b><u>    7.1.4. Sender Vouches:</u></b>	<b>47</b>
125	<b><u>    7.1.5. Password (Pass-Through):</u></b>	<b>47</b>
126	<b><u>    7.1.6. Password (One-Way-Function SHA-1):</u></b>	<b>47</b>
127	<b><u>    7.1.7. Kerberos</u></b>	<b>47</b>
128	<b><u>    7.1.8. SSL/TLS Certificate Based Client Authentication:</u></b>	<b>48</b>
129	<b><u>    7.1.9. Object Authenticator (SHA-1):</u></b>	<b>48</b>
130	<b><u>    7.1.10. PKCS#7</u></b>	<b>48</b>
131	<b><u>    7.1.11. Cryptographic Message Syntax</u></b>	<b>48</b>
132	<b><u>    7.1.12. XML Digital Signature</u></b>	<b>48</b>
133	<b><u>7.2. ACTION NAMESPACE IDENTIFIERS</u></b>	<b>49</b>
134	<b><u>    7.2.1. Read/Write/Execute/Delete/Control:</u></b>	<b>49</b>
135	<b><u>    7.2.2. Read/Write/Execute/Delete/Control with Negation:</u></b>	<b>49</b>
136	<b><u>    7.2.3. Get/Head/Put/Post:</u></b>	<b>49</b>
137	<b><u>    7.2.4. UNIX File Permissions:</u></b>	<b>50</b>
138	<b><u>8. SAML SCHEMA LISTINGS</u></b>	<b>51</b>
139	<b><u>    8.1. ASSERTION SCHEMA</u></b>	<b>51</b>
140	<b><u>    8.2. PROTOCOL SCHEMA</u></b>	<b>55</b>
141	<b><u>9. REFERENCES</u></b>	<b>59</b>
142	<b><u>APPENDIX A. NOTICES</u></b>	<b>63</b>
143	<b><u>ASSERTIONS AND PROTOCOL FOR THE OASIS SECURITY ASSERTION MARKUP</u></b>	
144	<b><u>LANGUAGE (SAML)</u></b>	<b>1</b>

145	<b>1. INTRODUCTION</b>	6
146	<b>1.1. NOTATION</b>	6
147	<b>1.2. SCHEMA ORGANIZATION AND NAMESPACES</b>	6
148	<b>1.2.1. Time Values.</b>	7
149	<b>1.2.2. Comparing SAML values</b>	7
150	<b>1.3. SAML CONCEPTS (NON NORMATIVE)</b>	7
151	<b>1.3.1. Overview</b>	7
152	<b>1.3.2. SAML and URI-Based Identifiers</b>	9
153	<b>1.3.3. SAML and Extensibility</b>	9
154	<b>2. SAML ASSERTIONS</b>	10
155	<b>2.1. SCHEMA HEADER AND NAMESPACE DECLARATIONS</b>	10
156	<b>2.2. SIMPLE TYPES</b>	10
157	<b>2.2.1. Simple Types IDType and IDReferenceType</b>	10
158	<b>2.2.2. Simple Type DecisionType</b>	11
159	<b>2.3. ASSERTIONS</b>	11
160	<b>2.3.1. Element &lt;AssertionSpecifier&gt;</b>	11
161	<b>2.3.2. Element &lt;AssertionID&gt;</b>	12
162	<b>2.3.3. Element &lt;Assertion&gt;</b>	12
163	<b>2.3.3.1. Element &lt;Conditions&gt;</b>	13
164	<b>2.3.3.1.1. Attributes NotBefore and NotOnOrAfter</b>	14
165	<b>2.3.3.1.2. Element &lt;Condition&gt;</b>	14
166	<b>2.3.3.1.3. Elements &lt;AudienceRestrictionCondition&gt; and &lt;Audience&gt;</b>	14
167	<b>2.3.3.1.4. Elements &lt;TargetRestrictionCondition&gt; and &lt;Target&gt;</b>	15
168	<b>2.3.3.2. Elements &lt;Advice&gt; and &lt;AdviceElement&gt;</b>	15
169	<b>2.4. STATEMENTS</b>	16
170	<b>2.4.1. Element &lt;Statement&gt;</b>	16
171	<b>2.4.2. Element &lt;SubjectStatement&gt;</b>	16
172	<b>2.4.2.1. Element &lt;Subject&gt;</b>	17

173	<b>2.4.2.2. Element &lt;NameIdentifier&gt;</b>	17
174	<b>2.4.2.3. Elements &lt;SubjectConfirmation&gt;, &lt;ConfirmationMethod&gt;, and &lt;SubjectConfirmationData&gt;</b>	17
175	<b>2.4.3. Element &lt;AuthenticationStatement&gt;</b>	18
176	<b>2.4.3.1. Element &lt;AuthenticationLocality&gt;</b>	19
177	<b>2.4.3.2. Element &lt;AuthorityBinding&gt;</b>	19
178	<b>2.4.4. Element &lt;AuthorizationDecisionStatement&gt;</b>	20
179	<b>2.4.4.1. Elements &lt;Actions&gt; and &lt;Action&gt;</b>	21
180	<b>2.4.4.2. Element &lt;Evidence&gt;</b>	21
181	<b>2.4.5. Element &lt;AttributeStatement&gt;</b>	21
182	<b>2.4.5.1. Elements &lt;AttributeDesignator&gt; and &lt;Attribute&gt;</b>	22
183	<b>2.4.5.1.1. Element &lt;AttributeValue&gt;</b>	22
184	<b>3. SAML PROTOCOL</b>	24
185	<b>3.1. SCHEMA HEADER AND NAMESPACE DECLARATIONS</b>	24
186	<b>3.2. REQUESTS</b>	24
187	<b>3.2.1. Complex Type RequestAbstractType</b>	24
188	<b>3.2.1.1. Element &lt;RespondWith&gt;</b>	25
189	<b>3.2.2. Element &lt;Request&gt;</b>	26
190	<b>3.2.3. Element &lt;AssertionArtifact&gt;</b>	27
191	<b>3.3. QUERIES</b>	27
192	<b>3.3.1. Element &lt;Query&gt;</b>	27
193	<b>3.3.2. Element &lt;SubjectQuery&gt;</b>	27
194	<b>3.3.3. Element &lt;AuthenticationQuery&gt;</b>	27
195	<b>3.3.4. Element &lt;AttributeQuery&gt;</b>	28
196	<b>3.3.5. Element &lt;AuthorizationDecisionQuery&gt;</b>	28
197	<b>3.4. RESPONSES</b>	29
198	<b>3.4.1. Complex Type ResponseAbstractType</b>	29
199	<b>3.4.2. Element &lt;Response&gt;</b>	30
200	<b>3.4.3. Element &lt;Status&gt;</b>	30
201	<b>3.4.3.1. Element &lt;StatusCode&gt;</b>	31
202	<b>3.4.3.2. Element &lt;SubStatusCode&gt;</b>	31
203	<b>3.4.3.3. Element &lt;StatusMessage&gt;</b>	32

204	3.4.3.4. Element <StatusDetail>	32
205	<b>3.4.4. Responses to &lt;AuthenticationQuery&gt; and &lt;AttributeQuery&gt;</b>	32
206	<b>4. SAML VERSIONING</b>	34
207	<b>4.1. ASSERTION VERSION</b>	34
208	<b>4.2. REQUEST VERSION</b>	34
209	<b>4.3. RESPONSE VERSION</b>	35
210	<b>5. SAML &amp; XML SIGNATURE SYNTAX AND PROCESSING</b>	36
211	<b>5.1. SIGNING ASSERTIONS</b>	36
212	<b>5.2. REQUEST/RESPONSE SIGNING</b>	37
213	<b>5.3. SIGNATURE INHERITANCE</b>	37
214	<b>5.3.1. Rationale</b>	37
215	<b>5.3.2. Rules for SAML Signature Inheritance</b>	37
216	<b>5.4. XML SIGNATURE PROFILE</b>	37
217	<b>5.4.1. Signing formats</b>	37
218	<b>5.4.2. CanonicalizationMethod</b>	37
219	<b>5.4.3. Transforms</b>	38
220	<b>5.4.4. KeyInfo</b>	38
221	<b>5.4.5. Binding between statements in a multi-statement assertion</b>	38
222	<b>6. SAML EXTENSIONS</b>	39
223	<b>6.1. ASSERTION SCHEMA EXTENSION</b>	39
224	<b>6.2. PROTOCOL SCHEMA EXTENSION</b>	39
225	<b>6.3. USE OF TYPE DERIVATION AND SUBSTITUTION GROUPS</b>	40
226	<b>7. SAML-DEFINED IDENTIFIERS</b>	41
227	<b>7.1. CONFIRMATION METHOD IDENTIFIERS</b>	41

228	<b><u>7.1.1. SAML Artifact:</u></b>	41
229	<b><u>7.1.2. SAML Artifact (SHA-1):</u></b>	41
230	<b><u>7.1.3. Holder of Key:</u></b>	41
231	<b><u>7.1.4. Sender Vouches:</u></b>	41
232	<b><u>7.1.5. Password (Pass Through):</u></b>	41
233	<b><u>7.1.6. Password (One Way Function SHA-1):</u></b>	42
234	<b><u>7.1.7. Kerberos</u></b>	42
235	<b><u>7.1.8. SSL/TLS Certificate Based Client Authentication:</u></b>	42
236	<b><u>7.1.9. Object Authenticator (SHA-1):</u></b>	42
237	<b><u>7.1.10. PKCS#7</u></b>	42
238	<b><u>7.1.11. Cryptographic Message Syntax</u></b>	43
239	<b><u>7.1.12. XML Digital Signature</u></b>	43
240	<b><u>7.2. ACTION NAMESPACE IDENTIFIERS</u></b>	43
241	<b><u>7.2.1. Read/Write/Execute/Delete/Control:</u></b>	43
242	<b><u>7.2.2. Read/Write/Execute/Delete/Control with Negation:</u></b>	43
243	<b><u>7.2.3. Get/Head/Put/Post:</u></b>	44
244	<b><u>7.2.4. UNIX File Permissions:</u></b>	44
245	<b><u>8. SAML SCHEMA LISTINGS</u></b>	45
246	<b><u>8.1. ASSERTION SCHEMA</u></b>	45
247	<b><u>8.2. PROTOCOL SCHEMA</u></b>	48
248	<b><u>9. REFERENCES</u></b>	52
249	<b><u>APPENDIX A. NOTICES</u></b>	54
250		

# 1. Introduction

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

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

## 1.1. Notation

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

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

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

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

Listings of SAML schemas appear like this.

Example code listings appear like this.

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

?? The prefix saml: stands for the SAML assertion namespace.

?? The prefix samlp: stands for the SAML request-response protocol namespace.

?? The prefix ds: stands for the W3C XML Signature namespace.

?? The prefix xsd: stands for the W3C XML Schema namespace in example listings. In schema listings, this is the default namespace and no prefix is shown.

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

## 1.2. Schema Organization and Namespaces

The SAML assertion structures are defined in a schema associated with the following XML namespace:

<http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-assertion-28.xsd>

The SAML request-response protocol structures are defined in a schema associated with the following XML namespace:

<http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-protocol-28.xsd>

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

292       The assertion schema is imported into the protocol schema. Also imported into both schemas is the  
293       schema for XML Signature , which is associated with the following XML namespace:

294       <http://www.w3.org/2000/09/xmldsig#>

### 295       **1.2.1. Time Values.**

296       All SAML time values have the type **dateTime**, which is built in to the W3C XML Schema Datatypes  
297       specification and MUST be expressed in UTC form.

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

### 300       **1.2.2. Comparing SAML values**

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

307       If an implementation is comparing values that are represented using different character encodings,  
308       the implementation MUST use a comparison method that returns the same result as converting  
309       both values to the Unicode character encoding (<http://www.unicode.org>), Normalization Form C  
310       and then performing an exact binary comparison. This requirement is intended to conform to the  
311       W3C Character Model for the World Wide Web (), and in particular the rules for Unicode-  
312       normalized Text.

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

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

## 322       **1.3. SAML Concepts (Non-Normative)**

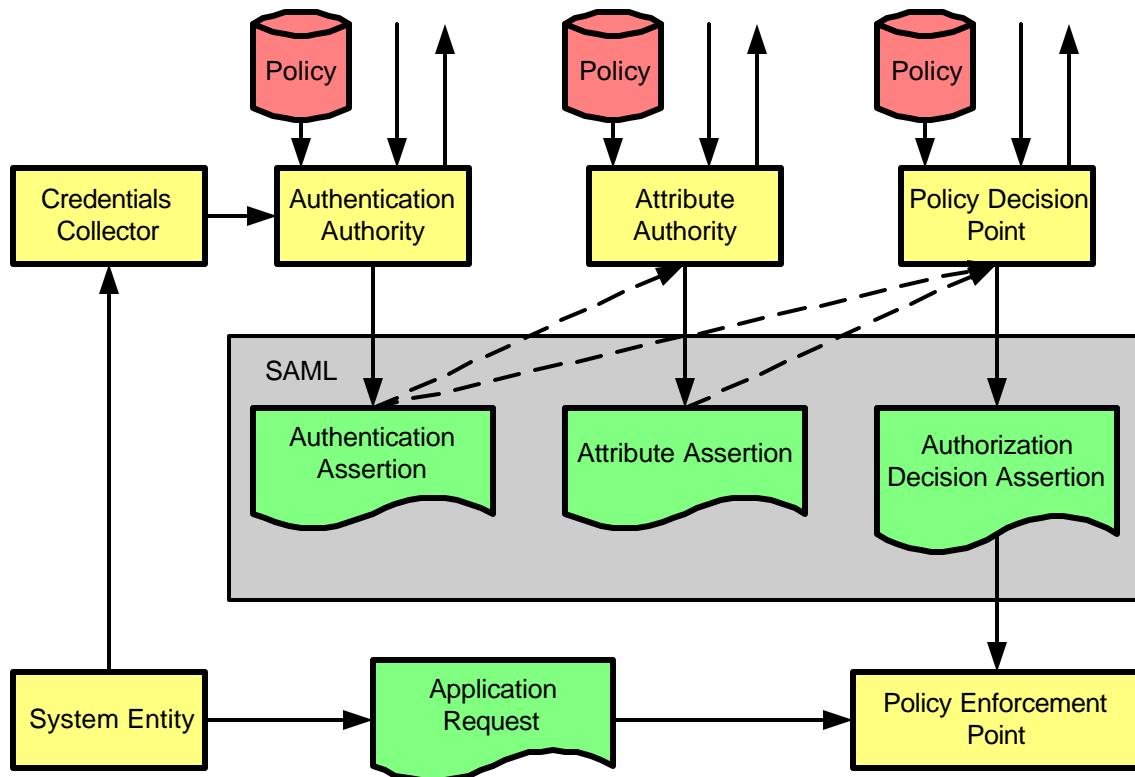
323       This section is informative only and is superseded by any contradicting information in the normative  
324       text in Sections 1.2 and following. A glossary of SAML terms and concepts is available.

### 325       **1.3.1. Overview**

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

331       Assertions can convey information about authentication acts performed by subjects, attributes of  
332       subjects, and authorization decisions about whether subjects are allowed to access certain

333 resources. Assertions are represented as XML constructs and have a nested structure, whereby a  
 334 single assertion might contain several different internal statements about authentication,  
 335 authorization, and attributes. Note that assertions containing authentication statements  
 336 merely describe acts of authentication that happened previously.  
 337 Assertions are issued by SAML authorities, namely, authentication authorities, attribute authorities,  
 338 and policy decision points. SAML defines a protocol by which clients can request assertions from  
 339 SAML authorities and get a response from them. This protocol, consisting of XML-based request  
 340 and response message formats, can be bound to many different underlying communications and  
 341 transport protocols; SAML currently defines one binding, to SOAP over HTTP.  
 342 SAML authorities can use various sources of information, such as external policy stores and  
 343 assertions that were received as input in requests, in creating their responses. Thus, while clients  
 344 always consume assertions, SAML authorities can be both producers and consumers of assertions.  
 345 The following model is conceptual only; for example, it does not account for real-world information  
 346 flow or the possibility of combining of authorities into a single system.



347  
 348 **Figure 1 The SAML Domain Model**  
 349 One major design goal for SAML is Single Sign-On (SSO), the ability of a user to authenticate in  
 350 one domain and use resources in other domains without re-authenticating. However, SAML can be  
 351 used in various configurations to support additional scenarios as well. Several profiles of SAML are  
 352 defined that support different styles of SSO and the securing of SOAP payloads.  
 353 The assertion and protocol data formats are defined in this specification. The bindings and profiles  
 354 are defined in a separate specification . A conformance program for SAML is defined in the  
 355 conformance specification . Security issues are discussed in a separate security and privacy  
 356 considerations specification .

357    **1.3.2. SAML and URI-Based Identifiers**

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

361    **urn:ietf:rfc:1510**

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

364    **<http://www.oasis-open.org/committees/security/docs/draft-sstc-core-28#rwedc>**

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

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

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

376    **1.3.3. SAML and Extensibility**

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

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

## 381 2. SAML Assertions

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

- 384 ?? **Authentication**: The specified subject was authenticated by a particular means at a  
385 particular time.
- 386 ?? **Authorization Decision**: A request to allow the specified subject to access the specified  
387 resource has been granted or denied.
- 388 ?? **Attribute**: The specified subject is associated with the supplied attributes.

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

### 392 2.1. Schema Header and Namespace Declarations

393 The following schema fragment defines the XML namespaces and other header information for the  
394 assertion schema:

```
395 <schema
396   targetNamespace="http://www.oasis-open.org/committees/security/docs/draft-
397   sstc-schema-assertion-28.xsd"
398   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
399   xmlns:saml="http://www.oasis-open.org/committees/security/docs/draft-sstc-
400   schema-assertion-28.xsd"
401   xmlns="http://www.w3.org/2001/XMLSchema"
402   elementFormDefault="unqualified">
403   <import namespace="http://www.w3.org/2000/09/xmldsig#"
404     schemaLocation="xmldsig-core-schema.xsd" />
405   <annotation>
406     <documentation>draft-sstc-schema-assertion-28.xsd</documentation>
407   </annotation>
408 ...
409 </schema>
```

## 410 2.2. Simple Types

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

### 412 2.2.1. Simple Types **IDType** and **IDReferenceType**

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

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

- 416 ?? Any party that assigns an identifier MUST ensure that there is negligible probability that that  
417 party or any other party will accidentally assign the same identifier to a different data object.
- 418 ?? Where a data object declares that it has a particular identifier, there MUST be exactly one  
419 such declaration.

420 The mechanism by which the application ensures that the identifier is unique is left to the  
421 implementation. In the case that a pseudorandom technique is employed, the probability of two  
422 randomly chosen identifiers being identical MUST be less than  $2^{-128}$  and SHOULD be less than  
423  $2^{-160}$ . This requirement MAY be met by applying Base64 encoding to a randomly chosen value 128  
424 or 160 bits in length.

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

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

```
429 <simpleType name="IDType">  
430   <restriction base="string"/>  
431 </simpleType>  
432 <simpleType name="IDReferenceType">  
433   <restriction base="string"/>  
434 </simpleType>
```

## 435 2.2.2. Simple Type **DecisionType**

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

438 Permit

439 The specified action is permitted.

440 Deny

441 The specified action is denied.

442 Indeterminate

443 The issuer cannot determine whether the specified action is permitted or denied.

444 The Indeterminate Decision value is used in situations where the issuer requires the ability to  
445 provide an affirmative statement that it is not able to issue a decision. Additional information as to  
446 the reason for the refusal or inability to provide a decision MAY be returned as <StatusDetail>  
447 elements

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

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

```
450 <simpleType name="DecisionType">  
451   <restriction base="string">  
452     <enumeration value="Permit"/>  
453     <enumeration value="Deny"/>  
454     <enumeration value="Indeterminate"/>  
455   </restriction>  
456 </simpleType>
```

## 457 2.3. Assertions

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

### 459 2.3.1. Element **<AssertionSpecifier>**

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

462 <AssertionIDReference>

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

464 <Assertion>

465 Specifies an assertion by value.

466 The following schema fragment defines the <AssertionSpecifier> element and its  
467 AssertionSpecifierType complex type:

```
468 <element name="AssertionSpecifier" type="saml:AssertionSpecifierType"/>  
469 <complexType name="AssertionSpecifierType">  
470   <choice>
```

```
471 |     <element ref="saml:AssertionIDReference" />
472 |     <element ref="saml:Assertion" />
473 |   </choice>
474 | </complexType>
```

### 475 **2.3.2.3.1. Element <AssertionID>**

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

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

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

### 480 **2.3.3.2.3.2. Element <Assertion>**

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

483 MajorVersion [Required]

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

486 MinorVersion [Required]

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

489 AssertionID [Required]

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

492 Issuer [Required]

493 The issuer of the assertion. The name of the issuer is provided as a string. The issuer  
494 name SHOULD be unambiguous to the intended relying parties. SAML applications may  
495 use an identifier such as a URI [reference](#) that is designed to be unambiguous regardless of  
496 context.

497 IssueInstant [Required]

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

499 <Conditions> [Optional]

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

501 <Advice> [Optional]

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

504 <Signature> [Optional]

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

506 One or more of the following statement elements:

507 <Statement>

508 A statement defined in an extension schema.

509 <SubjectStatement>

510 A subject statement defined in an extension schema.

511 <AuthenticationStatement>

512 An authentication statement.

513 <AuthorizationDecisionStatement>

514 An authorization decision statement.

```

515 <AttributeStatement>
516     An attribute statement.
517 The following schema fragment defines the <Assertion> element and its AssertionType
518 complex type:
519 <element name="Assertion" type="saml:AssertionType" />
520 <complexType name="AssertionType">
521     <sequence>
522         <element ref="saml:Conditions" minOccurs="0" />
523         <element ref="saml:Advice" minOccurs="0" />
524         <choice maxOccurs="unbounded">
525             <element ref="saml:Statement" />
526             <element ref="saml:SubjectStatement" />
527             <element ref="saml:AuthenticationStatement" />
528             <element ref="saml:AuthorizationDecisionStatement" />
529             <element ref="saml:AttributeStatement" />
530         </choice>
531         <element ref="ds:Signature" minOccurs="0" />
532     </sequence>
533     <attribute name="MajorVersion" type="integer" use="required"/>
534     <attribute name="MinorVersion" type="integer" use="required"/>
535     <attribute name="AssertionID" type="saml:IDType" use="required"/>
536     <attribute name="Issuer" type="string" use="required"/>
537     <attribute name="IssueInstant" type="dateTime" use="required"/>
538 </complexType>
```

### **2.3.3.1.2.3.2.1. Element <Conditions>**

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

- 544 ?? If any condition evaluates to `Invalid`, the assertion status is `Invalid`.
- 545 ?? If no condition evaluates to `Invalid` and one or more conditions evaluate to `Indeterminate`, the assertion status is `Indeterminate`.
- 547 ?? If no conditions are supplied or all the specified conditions evaluate to `Valid`, the assertion status is `Valid`.

549 Note that an assertion that has validity status ‘`Valid`’ may not be trustworthy by reasons such as not  
550 being issued by a trustworthy issuer or not being authenticated by a trustworthy signature.

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

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

555 `NotBefore` [Optional]

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

558 `NotOnOrAfter` [Optional]

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

561 <Condition> [Any Number]

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

563    <AudienceRestrictionCondition> [Any Number]  
 564       Specifies that the assertion is addressed to a particular audience.  
 565    <~~TargetRestrictionCondition~~> [Any Number]  
 566       ~~The <TargetRestriction> condition is used to limit the use of the assertion to a particular relying party.~~  
 567  
 568    The following schema fragment defines the <Conditions> element and its **ConditionsType** complex type:  
 569  
 570    <element name="Conditions" type="saml:ConditionsType"/>  
 571    <complexType name="ConditionsType">  
 572       <choice minOccurs="0" maxOccurs="unbounded">  
 573         <element ref="saml:Condition"/>  
 574         <element ref="saml:AudienceRestrictionCondition"/>  
 575         <element ref="saml:TargetRestrictionCondition"/>  
 576       </choice>  
 577       <attribute name="NotBefore" type="dateTime" use="optional"/>  
 578       <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>  
 579    </complexType>

580    **2.3.3.1.12.3.2.1.1 Attributes NotBefore and NotOnOrAfter**  
 581    The `NotBefore` and `NotOnOrAfter` attributes specify time limits on the validity of the assertion.  
 582    The `NotBefore` attribute specifies the time instant at which the validity interval begins. The `NotOnOrAfter` attribute specifies the time instant at which the validity interval has ended.  
 583  
 584    If the value for either `NotBefore` or `NotOnOrAfter` is omitted it is considered unspecified. If the `NotBefore` attribute is unspecified (and if any other conditions that are supplied evaluate to `Valid`), the assertion is valid at any time before the time instant specified by the `NotOnOrAfter` attribute. If the `NotOnOrAfter` attribute is unspecified (and if any other conditions that are supplied evaluate to `Valid`), the assertion is valid from the time instant specified by the `NotBefore` attribute with no expiry. If neither attribute is specified (and if any other conditions that are supplied evaluate to `Valid`), the assertion is valid at any time.  
 585  
 586    The `NotBefore` and `NotOnOrAfter` attributes are defined to have the **dateTime** simple type that  
 587    is built in to the W3C XML Schema Datatypes specification . All time instants are specified in  
 588    Universal Coordinated Time (UTC) as described in section 1.2.1. Implementations MUST NOT  
 589    generate time instants that specify leap seconds.  
 590

595    **2.3.3.1.22.3.2.1.2 Element <Condition>**  
 596    The <Condition> element serves as an extension point for new conditions. Its  
 597    **ConditionAbstractType** complex type is abstract; extension elements MUST use the `xsi:type`  
 598    attribute to indicate the derived type.  
 599  
 600    The following schema fragment defines the <Condition> element and its  
 601    **ConditionAbstractType** complex type:  
 602    <element name="Condition" type="saml:ConditionAbstractType"/>  
 603    <complexType name="ConditionAbstractType" abstract="true"/>

603    **2.3.3.1.32.3.2.1.3 Elements <AudienceRestrictionCondition> and <Audience>**  
 604    The <AudienceRestrictionCondition> element specifies that the assertion is addressed to  
 605    one or more specific audiences identified by <Audience> elements. Although a party that is outside  
 606    the audiences specified is capable of drawing conclusions from an assertion, the issuer explicitly  
 607    makes no representation as to accuracy or trustworthiness to such a party. It contains the following  
 608    elements:

609 <Audience>  
610 A URI [reference](#) that identifies an intended audience. The URI [reference](#) MAY identify a  
611 document that describes the terms and conditions of audience membership.

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

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

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

```
622 <element name="AudienceRestrictionCondition"  
623   type="saml:AudienceRestrictionConditionType" />  
624 <complexType name="AudienceRestrictionConditionType">  
625   <complexContent>  
626     <extension base="saml:ConditionAbstractType">  
627       <sequence>  
628         <element ref="saml:Audience" maxOccurs="unbounded" />  
629       </sequence>  
630     </extension>  
631   </complexContent>  
632 </complexType>  
633 <element name="Audience" type="anyURI" />  
634 Element <TargetRestrictionCondition> and <Target>
```

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

638 <Target>  
639 A URI that identifies an intended relying party.  
640 The TargetRestrictionCondition evaluates to valid if and only if one or more URIs identify the  
641 recipient or a resource managed by the recipient.

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

```
644 <element name="TargetRestrictionCondition"  
645   type="saml:TargetRestrictionConditionType" />  
646 <complexType name="TargetRestrictionConditionType">  
647   <complexContent>  
648     <extension base="saml:ConditionAbstractType">  
649       <sequence>  
650         <element ref="saml:Target"  
651           minOccurs="1" maxOccurs="unbounded" />  
652       </sequence>  
653     </extension>  
654   </complexContent>  
655 </complexType>  
656 <element name="Target" type="anyURI" />
```

### 657 **2.3.3.2.3.2.2. Elements <Advice> and <AdviceElement>**

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

661     The <Advice> element contains a mixture of zero or more <Assertion**Specifier**> elements,  
662     <AssertionIDReference> elements, <AdviceElement> elements, and elements in other  
663     namespaces, with lax schema validation in effect for these other elements.

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

- 665       ?? Include evidence supporting the assertion claims to be cited, either directly (through  
666       incorporating the claims) or indirectly (by reference to the supporting assertions).
- 667       ?? State a proof of the assertion claims.
- 668       ?? Specify the timing and distribution points for updates to the assertion.

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

```
671 <element name="Advice" type="saml:AdviceType" />
672 <complexType name="AdviceType">
673   <choice minOccurs="0" maxOccurs="unbounded">
674     <element ref="saml:AssertionIDReference" />
675     <element ref="saml:Assertion" />-----element
676     ref="saml:AssertionSpecifier" />
677     <element ref="saml:AdviceElement" />
678     <any namespace="##other" processContents="lax" />
679   </choice>
680 </complexType>
681 <element name="AdviceElement" type="saml:AdviceAbstractType" />
682 <complexType name="AdviceAbstractType" />
```

## 683     2.4. Statements

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

### 685     2.4.1. Element <Statement>

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

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

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

### 693     2.4.2. Element <SubjectStatement>

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

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

```
701 <element name="SubjectStatement" type="saml:SubjectStatementAbstractType" />
702 <complexType name="SubjectStatementAbstractType" abstract="true" >
703   <complexContent>
704     <extension base="saml:StatementAbstractType" >
705       <sequence>
706         <element ref="saml:Subject" />
707       </sequence>
```

```
708     </extension>
709   </complexContent>
710 </complexType>
```

#### 711 2.4.2.1. Element <Subject>

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

```
714 <NameIdentifier>
715   An identification of a subject by its name and security domain.
716 <SubjectConfirmation>
717   Information that allows the subject to be authenticated.
```

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

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

```
725 <element name="Subject" type="saml:SubjectType" />
726 <complexType name="SubjectType">
727   <choice>
728     <sequence>
729       <element ref="saml:NameIdentifier" />
730       <element ref="saml:SubjectConfirmation" minOccurs="0" />
731     </sequence>
732     <element ref="saml:SubjectConfirmation" />
733   </choice>
734 </complexType>
```

#### 735 2.4.2.2. Element <NameIdentifier>

736 The <NameIdentifier> element specifies a subject by a combination of a name qualifier, a name  
737 and a format. It has the following attributes:

738 NameQualifier [Optional]

739 The security or administrative domain that qualifies the name of the subject.  
740 The NameQualifier attribute provides a means to federate names from disparate user  
741 stores without collision.

742 Format [Optional]

743 The syntax used to describe the name of the subject

744 The format value MUST be a URI reference. The following URI references are defined by this  
745 specification, where only the fragment identifier portion is shown, assuming a base URI of  
746 the SAML assertion namespace name.

747 #emailAddress

748 Indicates that the content of the NameIdentifier element is in the form of an email address,  
749 specifically "addr-spec" as defined in section 3.4.1 of RFC 2822 [RFC 2822]. An addr-spec  
750 has the form local-part@domain. Note that an addr-spec has no phrase (such as a  
751 common name) before it, has no comment (text surrounded in parentheses) after it, and is  
752 not surrounded by "<" and ">".

753 #X509SubjectName

754 Indicates that the content of the NameIdentifier element is in the form specified for  
755 the contents of <ds:X509SubjectName> element in [DSIG]. Implementors should note that

756 [DSIG] specifies encoding rules for X.509 subject names that differ from the rules given in  
757 RFC2253 [RFC2253].

758 #WindowsDomainQualifiedName  
759 Indicates that the content of the NameIdentifier element is a Windows domain qualified  
760 name. A Windows domain qualified user name is a string of the form  
761 "DomainName\UserName". The domain name and "\" separator may be omitted.

762 The ~~NameIdentifier~~ element specifies a subject by a combination of a name and a security  
763 domain. It has the following attributes:

764 SecurityDomain [Optional]  
765 The security domain governing the name of the subject.

766 Name [Required]  
767 The name of the subject.

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

771 The following schema fragment defines the `<NameIdentifier>` element and its  
772 `NameIdentifierType` complex type:

```
773 <element name="NameIdentifier" type="saml:NameIdentifierType" />
774 <complexType name="NameIdentifierType">
775   <simpleContent>
776     <extension base="string">
777       <attribute name="NameQualifier" type="string" use="optional" />
778       <attribute name="Format" type="anyURI" use="optional" />
779     </extension>
780   </simpleContent>
781 </complexType>
```

782 The interpretation of the NameQualifier, and NameIdentifier's content in the case of a Format not  
783 specified in this document, are left to individual implementations.

784 Regardless of format, issues of anonymity, pseudonymity, and the persistence of the identifier with  
785 respect to the asserting and relying parties, are also implementation-specific. —  
786 <element name="NameIdentifier" type="saml:NameIdentifierType"/>
787 <complexType name="NameIdentifierType">
788 <attribute name="SecurityDomain" type="string" />
789 <attribute name="Name" type="string" use="required" />
790 </complexType>

#### 791 2.4.2.3. Elements `<SubjectConfirmation>`, `<ConfirmationMethod>`, and 792 `<SubjectConfirmationData>`

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

795 <ConfirmationMethod> [One or more]  
796 A URI [reference](#) that identifies a protocol to be used to authenticate the subject. URI  
797 [references](#) identifying common authentication protocols are listed in Section 7.

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

800 <ds:KeyInfo> [Optional]  
801 An XML Signature element that specifies a cryptographic key held by the subject.

802 The following schema fragment defines the `<SubjectConfirmation>` element and its  
803 **SubjectConfirmationType** complex type, along with the `<SubjectConfirmationData>`  
804 element and the `<ConfirmationMethod>` element:

```
805 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType" />
806 <complexType name="SubjectConfirmationType">
807   <sequence>
808     <element ref="saml:ConfirmationMethod" maxOccurs="unbounded" />
809     <element ref="saml:SubjectConfirmationData" minOccurs="0" />
810     <element ref="ds:KeyInfo" minOccurs="0" />
811   </sequence>
812 </complexType>
813 <element name="SubjectConfirmationData" type="string" />
814 <element name="ConfirmationMethod" type="anyURI" />
```

### 815 2.4.3. Element `<AuthenticationStatement>`

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

820 `<AuthenticationMethod>` [Optional]

821 A URI [reference](#) that specifies the type of authentication that took place. URI [references](#) |  
822 identifying common authentication protocols are listed in Section 7.

823 `<AuthenticationInstant>` [Optional]

824 Specifies the time at which the authentication took place. The time value is encoded in UTC  
825 as described in section 1.2.1.

826 `<AuthenticationLocality>` [Optional]

827 Specifies the DNS domain name and IP address for the system entity from which the  
828 Subject was apparently authenticated.

829 `<AuthorityBinding>` [Any Number]

830 Indicates that additional information about the subject of the statement may be available.

831 The following schema fragment defines the `<AuthenticationStatement>` element and its  
832 **AuthenticationStatementType** complex type:

```
833 <element name="AuthenticationStatement"
834   type="saml:AuthenticationStatementType" />
835 <complexType name="AuthenticationStatementType">
836   <complexContent>
837     <extension base="saml:SubjectStatementAbstractType">
838       <sequence>
839         <element ref="saml:AuthenticationLocality" minOccurs="0" />
840         <element ref="saml:AuthorityBinding"
841           minOccurs="0" maxOccurs="unbounded" />
842       </sequence>
843       <attribute name="AuthenticationMethod" type="anyURI" />
844       <attribute name="AuthenticationInstant" type="dateTime" />
845     </extension>
846   </complexContent>
847 </complexType>
```

#### 848 2.4.3.1. Element `<AuthenticationLocality>`

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

851 `<IPAddress>` [Optional]

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

853 DNSAddress [Optional]  
854     The DNS address of the system entity that was authenticated.

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

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

```
<element name="AuthenticationLocality"
         type="saml:AuthenticationLocalityType"/>
<complexType name="AuthenticationLocalityType">
  <attribute name="IPAddress" type="string" use="optional"/>
  <attribute name="DNSAddress" type="string" use="optional"/>
</complexType>
```

#### 865 **2.4.3.2. Element <AuthorityBinding>**

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

871 AuthorityKind [Required]

872     The type of SAML authority (Authentication, Attribute, or Authorization Decision) advertised  
873 by the element. The kind of authority corresponds to the derived type of SubjectQuery that  
874 the authority expects to receive (and is likely to be able to successfully answer) at the  
875 location being advertised. For example, a value of "attribute" means that an  
876 <AttributeQuery> is expected. The type of SAML Protocol queries to which the authority  
877 described by this element will respond. The value is specified as an XML Schema QName.  
878 The acceptable values for AuthorityKind are the namespace-qualified names of  
879 element types or elements derived from the SAML Protocol Query element (see Section  
880 3.3). For example, an attribute authority would be identified by  
881 AuthorityKind="samlp:AttributeQuery". For extension schemas, where the actual  
882 type of the samlp:Query would be identified by an xsi:type attribute, the value of  
883 AuthorityKind MUST be the same as the value of the xsi:type attribute for the  
884 corresponding query.

885 Location [Required]

886     A URI [reference](#) describing how to locate and communicate with the authority, the exact  
887 syntax of which depends on the protocol binding in use. For example, a binding based on  
888 HTTP will be a web URL, while a binding based on SMTP might use the "mailto" scheme.

889 Binding [Required]

890     A URI [reference](#) identifying the SAML protocol binding to use in communicating with the  
891 authority. All SAML protocol bindings will have an assigned URI [reference](#).

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

```
<element name="AuthorityBinding" type="saml:AuthorityBindingType"/>
<complexType name="AuthorityBindingType">
  <attribute name="AuthorityKind" type="QName" use="required"/>—
  <attribute name="AuthorityKind" type="saml:AuthorityKindType"
            use="required"/>—
    <attribute name="Location" type="anyURI" use="required"/>
    <attribute name="Binding" type="anyURI" use="required"/>
  </complexType>
  <simpleType name="AuthorityKindType">
    <restriction base="string">
```

```
905     <enumeration value="authentication"/>
906     <enumeration value="attribute"/>
907     <enumeration value="authorization"/>
908   </restriction>
909 </simpleType>
```

#### 910 2.4.4. Element <AuthorizationDecisionStatement>

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

914 The resource is identified by means of a URI [reference](#). In order for the assertion to be interpreted  
915 correctly and securely the issuer and relying party MUST interpret each URI [reference](#) in a  
916 consistent manner. Failure to achieve a consistent URI [reference](#) interpretation can result in  
917 different authorization decisions depending on the encoding of the resource URI [reference](#). Rules  
918 for normalizing URI [references](#) are to be found in §6

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

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

926 ?? The assertion issuer SHOULD encode all resource URIs in normalized form.

927 ?? Relying parties SHOULD convert resource URIs to normalized form prior to processing.

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

932 ?? Parts of the URI syntax are case sensitive. If the underlying file system is case insensitive a  
933 requestor SHOULD NOT be able to gain access to a denied resource by changing the case  
934 of a part of the resource URI.

935 ?? Many file systems support mechanisms such as logical paths and symbolic links which  
936 allow users to establish logical equivalences between file system entries. A requestor  
937 SHOULD NOT be able to gain access to a denied resource by creating such an  
938 equivalence.

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

942 **Resource** [Required]  
943     A URI [reference](#) identifying the resource to which access authorization is sought.

944 **Decision** [Required]  
945     The decision rendered by the issuer with respect to the specified resource. The value is of  
946     the **DecisionType** simple type.

947 **<Actions>** [[One or more Required](#)]  
948     The set of actions authorized to be performed on the specified resource.

949 **<Evidence>** [Any Number]  
950     A set of assertions that the issuer relied on in making the decision.

951 The following schema fragment defines the `<AuthorizationDecisionStatement>` element  
952 and its **AuthorizationDecisionStatementType** complex type:

```
953 <element name="AuthorizationDecisionStatement"  
954 type="saml:AuthorizationDecisionStatementType"/>  
955 <complexType name="AuthorizationDecisionStatementType">  
956   <complexContent>  
957     <extension base="saml:SubjectStatementAbstractType">  
958       <sequence>  
959         <element ref="saml:Action" maxOccurs="unbounded"/>  
960         <element ref="saml:Evidence" minOccurs="0"/>  
961       </sequence>  
962       <attribute name="Resource" type="anyURI" use="required"/>  
963       <attribute name="Decision" type="saml:DecisionType" use="required"/>  
964     </extension>  
965   </complexContent>  
966 </complexType> <element name="AuthorizationDecisionStatement"  
967 type="saml:AuthorizationDecisionStatementType"/>  
968 <complexType name="AuthorizationDecisionStatementType">  
969   <complexContent>  
970     <extension base="saml:SubjectStatementAbstractType">  
971       <sequence>  
972         <element ref="saml:Actions"/>  
973         <element ref="saml:Evidence" minOccurs="0"  
974             maxOccurs="unbounded"/>  
975       </sequence>  
976       <attribute name="Resource" type="anyURI" use="required"/>  
977       <attribute name="Decision" type="saml:DecisionType"  
978           use="required"/>  
979     </extension>  
980   </complexContent>  
981 </complexType>
```

#### 982 2.4.4.1. Elements `<Actions>` and `<Action>`

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

985 Namespace [Optional]

986 A URI URI reference representing the namespace in which the names of the specified  
987 actions is are to be interpreted. If this element is absent, the namespace <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-28#rwedc-negation> specified in section  
988 7.2.2 is in effect.

990 `<Action> [One or more] string data [Required]`

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

992 The following schema fragment defines the `<Actions>` element and, its **ActionsType** complex  
993 type, and the `<Action>` element:

```
994 <element name="Action" type="saml:ActionType"/>  
995 <complexType name="ActionType">  
996   <simpleContent>  
997     <extension base="string">  
998       <attribute name="Namespace" type="anyURI"/>  
999     </extension>  
1000   </simpleContent>  
1001 </complexType> <element name="Actions" type="saml:ActionsType"/>  
1002 <complexType name="ActionsType">  
1003   <sequence>  
1004     <element ref="saml:Action" maxOccurs="unbounded"/>  
1005   </sequence>  
1006   <attribute name="Namespace" type="anyURI" use="optional"/>
```

```
1007 -----</complexType>
1008 -----<element name="Action" type="string"/>
```

#### 1009 **2.4.4.2. Element <Evidence>**

1010 The <Evidence> element contains an assertion that the issuer relied on in issuing the  
1011 authorization decision. It has the **EvidenceType AssertionSpecifierType** complex type.

1012 It contains one of the following elements:

1013 <AssertionIDReference>  
1014     Specifies an assertion by reference to the value of the assertion's AssertionID attribute.
1015 <Assertion>  
1016     Specifies an assertion by value.

1017 The provision of an assertion as evidence MAY affect the reliance agreement between the  
1018 requestor and the Authorization Authority. For example, in the case that the requestor presented an  
1019 assertion to the Authorization Authority in a request, the Authorization Authority MAY use that  
1020 assertion as evidence in making its response without endorsing the assertion as valid either to the  
1021 requestor or any third party.

1022 The following schema fragment defines the <Evidence> element and its EvidenceType complex  
1023 type: The following schema fragment defines the <Evidence> element:

```
1024 -----<element name="Evidence" type="saml:EvidenceType"/>
1025 -----<complexType name="EvidenceType">
1026 -----<choice maxOccurs="unbounded">
1027 -----<element ref="saml:AssertionIDReference" />
1028 -----<element ref="saml:Assertion" />
1029 -----</choice>
1030 -----</complexType>-----<element name="Evidence" type="saml:AssertionSpecifierType"/>
```

#### 1031 **2.4.5. Element <AttributeStatement>**

1032 The <AttributeStatement> element supplies a statement by the issuer that the specified  
1033 subject is associated with the specified attributes. It is of type **AttributeStatementType**, which  
1034 extends **SubjectStatementAbstractType** with the addition of the following element:

1035 <Attribute> [One or More]  
1036     The <Attribute> element specifies an attribute of the subject.

1037 The following schema fragment defines the <AttributeStatement> element and its  
1038 **AttributeStatementType** complex type:

```
1039 -----<element name="AttributeStatement" type="saml:AttributeStatementType"/>
1040 -----<complexType name="AttributeStatementType">
1041 -----<complexContent>
1042 -----<extension base="saml:SubjectStatementAbstractType">
1043 -----<sequence>
1044 -----<element ref="saml:Attribute" maxOccurs="unbounded" />
1045 -----</sequence>
1046 -----</extension>
1047 -----</complexContent>
1048 -----</complexType>
```

#### 1049 **2.4.5.1. Elements <AttributeDesignator> and <Attribute>**

1050 The <AttributeDesignator> element identifies an attribute name within an attribute  
1051 namespace. It has the **AttributeDesignatorType** complex type. It is used in an attribute **assertion**  
1052 query to request that attribute values within a specific namespace be returned (see 3.3.4 for more  
1053 information). The <AttributeDesignator> element contains the following XML attributes:

1054     AttributeNamespace [Optional]  
 1055         The namespace in which the AttributeName elements are interpreted.  
 1056     AttributeName [Optional]  
 1057         The name of the attribute.  
 1058     The following schema fragment defines the `<AttributeDesignator>` element and its  
 1059         **AttributeDesignatorType** complex type:  
 1060         `<element name="AttributeDesignator" type="saml:AttributeDesignatorType" />`  
 1061         `<complexType name="AttributeDesignatorType">`  
 1062             `<attribute name="AttributeName" type="string" use="required" />`  
 1063             `<attribute name="AttributeNamespace" type="anyURI" use="required" />`  
 1064         `</complexType>`  
 1065     The `<Attribute>` element supplies the value for an attribute of an assertion subject. It has the  
 1066         **AttributeType** complex type, which extends **AttributeDesignatorType** with the addition of the  
 1067         following element:  
 1068         `<AttributeValue> [Any Number]`  
 1069             The value of the attribute.  
 1070     The following schema fragment defines the `<Attribute>` element and its **AttributeType** complex  
 1071         type:  
 1072         `<element name="Attribute" type="saml:AttributeType" />`  
 1073         `<complexType name="AttributeType">`  
 1074             `<complexContent>`  
 1075                 `<extension base="saml:AttributeDesignatorType" >`  
 1076                 `<sequence>`  
 1077                     `<element ref="saml:AttributeValue" maxOccurs="unbounded" />`  
 1078                 `</sequence>`  
 1079                 `</extension>`  
 1080             `</complexContent>`  
 1081         `</complexType>`  
 1082     **2.4.5.1.1 Element `<AttributeValue>`**  
 1083     The `<AttributeValue>` element supplies the value of a specified attribute. It is of the **anyType**  
 1084         simple type, which allows any well-formed XML to appear as the content of the element.  
 1085     If the data content of an `<AttributeValue>` element is of a XML Schema simple type (e.g. integer,  
 1086         string, etc) the data type MAY be declared explicitly by means of an `xsi:type` declaration in the  
 1087         `<AttributeValue>` element. If the attribute value contains structured data the necessary data  
 1088         elements may be defined in an extension schema introduced by means of the `xmlns=` mechanism.  
 1089     The following schema fragment defines the `<AttributeValue>` element:  
 1090         `<element name="AttributeValue" type="anyType" />`

### 3. SAML Protocol

SAML assertions MAY be generated and exchanged using a variety of protocols. The bindings and profiles specification for SAML describes specific means of transporting assertions using existing widely deployed protocols.

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



Figure 2: SAML Request-Response Protocol

#### 3.1. Schema Header and Namespace Declarations

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

```
<schema
  targetNamespace="http://www.oasis-open.org/committees/security/docs/draft-
sstc-schema-protocol-28.xsd"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:samlp="http://www.oasis-open.org/committees/security/docs/draft-sstc-
schema-protocol-28.xsd"
  xmlns:saml="http://www.oasis-open.org/committees/security/docs/draft-sstc-
schema-assertion-28.xsd"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  elementFormDefault="unqualified">
  <import namespace="http://www.oasis-open.org/committees/security/docs/draft-
sstc-schema-assertion-28.xsd"
    schemaLocation="draft-sstc-schema-assertion-28.xsd"/>
  <import namespace="http://www.w3.org/2000/09/xmldsig#"
    schemaLocation="xmldsig-core-schema.xsd"/>
  <annotation>
    <documentation>draft-sstc-schema-protocol-28.xsd</documentation>
  </annotation>
  ...
</schema>
```

#### 3.2. Requests

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

##### 3.2.1. Complex Type RequestAbstractType

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

RequestID [Required]

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.

1134 MajorVersion [Required]  
 1135     The major version of this request. The identifier for the version of SAML defined in this  
 1136     specification is 1. Processing of this attribute is specified in Section 3.4.2.  
 1137 MinorVersion [Required]  
 1138     The minor version of this request. The identifier for the version of SAML defined in this  
 1139     specification is 0. Processing of this attribute is specified in Section 3.4.2.  
 1140 IssueInstant [Required]  
 1141     The time instant of issue of the request. The time value is encoded in UTC as described in  
 1142     section 1.2.1.  
 1143 <RespondWith> [Any Number]  
 1144     Each <RespondWith> element specifies a type of response that is acceptable to the  
 1145     requestor.  
 1146 <Signature> [Optional]  
 1147     An XML Signature that authenticates the assertion, see section 5.

1148 The following schema fragment defines the **RequestAbstractType** complex type:

```

<complexType name="RequestAbstractType" abstract="true">
  <sequence>
    <element ref="samlp:RespondWith"
             minOccurs="0" maxOccurs="unbounded"/>
    <element ref = "ds:Signature" minOccurs="0"/>
  </sequence>
  <attribute name="RequestID" type="saml:IDType" use="required"/>
  <attribute name="MajorVersion" type="integer" use="required"/>
  <attribute name="MinorVersion" type="integer" use="required"/>
  <attribute name="IssueInstant" type="dateTime" use="required"/>
</complexType>

```

### 3.2.1.1. Element <RespondWith>

The **<RespondWith>** element specifies the type of Statement the requestor wants from the responder. Multiple **<RespondWith>** elements MAY be included to indicate that the requestor will accept assertions containing any of the specified types. If no **<RespondWith>** element is given, the responder may return assertions containing statements of any type.

If the requestor sends one or more **<RespondWith>** elements, the responder MUST NOT respond with assertions containing statements of any type not specified in one of the **<RespondWith>** elements. The **<RespondWith>** element specifies a type of response that is acceptable to the requestor. If no **<RespondWith>** element is specified the default is SingleStatement.

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

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

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

**<RespondWith>** element values are XML QNames. The XML namespace and name specifically refer to the namespace and element name of the Statement element, exactly as for the saml:AuthorityKind attribute; see section 2.4.3.2. For example, a requestor that wishes to

1182 receive assertions containing only attribute statements must specify  
1183 <RespondWith>saml:AttributeStatement</RespondWith>. To specify extension types,  
1184 the <RespondWith> element MUST contain exactly the extension element type as specified in the  
1185 xsi:type attribute on the corresponding element. <RespondWith> element values are URIs. A  
1186 requestor MAY use an XML schema identifier as a <RespondWith> element value to inform the  
1187 responder that the specified SAML extension schema is supported. <RespondWith> values  
1188 defined in this document are specified as URI fragment identifiers, the nominal base for these  
1189 identifier values being the SAML protocol schema identifier URI.

1190 Acceptable values for the <Respondwith> element are:

1191 #SingleStatement  
1192     An assertion carrying exactly one statement element.

1193 #MultipleStatement  
1194     An assertion carrying at least one statement element.

1195 #AuthenticationStatement  
1196     An assertion carrying an Authentication statement.

1197 #AuthorizationDecisionStatement  
1198     An assertion carrying an Authorization Decision statement.

1199 #AttributeStatement  
1200     An assertion carrying an Attribute statement.

1201 Schema URI  
1202     An assertion containing additional elements from the specified schema.

1203 The following schema fragment defines the <RespondWith> element:

```
1204 <element name="RespondWith" type="anyURIOrName" />
```

### 1205 3.2.2. Element <Request>

1206 The <Request> element specifies a SAML request. It provides either a query or a request for a  
1207 specific assertion identified by <AssertionIDReference> or <AssertionArtifact>. It has  
1208 the complex type **RequestType**, which extends **RequestAbstractType** by adding a choice of one  
1209 of the following elements:

1210 <Query>  
1211     An extension point that allows extension schemas to define new types of query.

1212 <SubjectQuery>  
1213     An extension point that allows extension schemas to define new types of query that specify  
1214     a single SAML subject.

1215 <AuthenticationQuery>  
1216     Makes a query for authentication information.

1217 <AttributeQuery>  
1218     Makes a query for attribute information.

1219 <AuthorizationDecisionQuery>  
1220     Makes a query for an authorization decision.

1221 <AssertionIDReference> [One or more]  
1222     Requests assertions by reference to its assertion identifier.

1223 <AssertionArtifact> [One or more]  
1224     Requests assertions by supplying an assertion artifact that represents it.

1225 The following schema fragment defines the `<Request>` element and its **RequestType** complex  
1226 type:  
1227

```
<element name="Request" type="samlp:RequestType"/>
<complexType name="RequestType">
  <complexContent>
    <extension base="samlp:RequestAbstractType">
      <choice>
        <element ref="samlp:Query"/>
        <element ref="samlp:SubjectQuery"/>
        <element ref="samlp:AuthenticationQuery"/>
        <element ref="samlp:AttributeQuery"/>
        <element ref="samlp:AuthorizationDecisionQuery"/>
        <element ref="saml:AssertionIDReference" maxOccurs="unbounded"/>
        <element ref="samlp:AssertionArtifact" maxOccurs="unbounded"/>
      </choice>
    </extension>
  </complexContent>
</complexType>
```

### 3.2.3. Element `<AssertionArtifact>`

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

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

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

## 3.3. Queries

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

### 3.3.1. Element `<Query>`

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

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

```
<element name="Query" type="samlp:QueryAbstractType"/>
<complexType name="QueryAbstractType" abstract="true"/>
```

### 3.3.2. Element `<SubjectQuery>`

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

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

```
<element name="SubjectQuery" type="samlp:SubjectQueryAbstractType"/>
<complexType name="SubjectQueryAbstractType" abstract="true">
  <complexContent>
    <extension base="samlp:QueryAbstractType">
      <sequence>
        <element ref="saml:Subject"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

```
1272         </sequence>
1273     </extension>
1274   </complexContent>
1275 </complexType>
```

### 1276 3.3.3. Element <AuthenticationQuery>

1277 The <AuthenticationQuery> element is used to make the query “What assertions  
1278 authenticationcontaining authentication statementsassertions are available for this subject?” A  
1279 successful response will be in the form of assertions containing authentication statements. This  
1280 element is of type **AuthenticationQueryType**, which extends **SubjectQueryAbstractType** with  
1281 the addition of the following element:

1282 <ConfirmationMethod> [Optional]

1283 A filter for possible responses. If it is present, the query made is “What assertions  
1284 containing authentication statementsauthentication assertions do you have for this subject  
1285 with the supplied confirmation method?”

1286 In response to an authentication query, a responder returns assertions with authentication  
1287 statements as follows: The <Subject> element in the returned assertions MUST be identical to  
1288 the <Subject> element of the query. If the <ConfirmationMethod> element is present in the  
1289 query, at least one <ConfirmationMethod> element in the response MUST match. It is  
1290 OPTIONAL for the complete set of all such matching assertions to be returned in the response.

1291 The following schema fragment defines the <AuthenticationQuery> type and its  
1292 **AuthenticationQueryType** complex type:

```
1293 <element name="AuthenticationQuery" type="samlp:AuthenticationQueryType" />
1294 <complexType name="AuthenticationQueryType">
1295   <complexContent>
1296     <extension base="samlp:SubjectQueryAbstractType">
1297       <sequence>
1298         <element ref="saml:ConfirmationMethod" minOccurs="0" />
1299       </sequence>
1300     </extension>
1301   </complexContent>
1302 </complexType>
```

### 1303 3.3.4. Element <AttributeQuery>

1304 The <AttributeQuery> element is used to make the query “Return the requested attributes for  
1305 this subject.” A successful response will be in the form of assertions containing attribute statements.  
1306 This element is of type **AttributeQueryType**, which extends **SubjectQueryAbstractType** with the  
1307 addition of the following element and attribute:

1308 Resource [Optional]

1309 The Resource attribute if present specifies that the attribute query is made in response to a  
1310 specific authorization decision relating to the resource. The responder MAY use the  
1311 resource attribute to establish the scope of the request.

1312 If the resource attribute is specified and the responder does not wish to support resource-  
1313 specific attribute queries, or if the resource value provided is invalid or unrecognized, then it  
1314 SHOULD respond with a SAML status of "Error.Receiver.ResourceNotRecognized".

1315 <AttributeDesignator> [Any Number] (see Section 2.4.5.1)

1316 Each <AttributeDesignator> element specifies an attribute whose value is to be  
1317 returned. If no attributes are specified, the list of desired attributes is implicit and  
1318 application-specific.

1319 The following schema fragment defines the <AttributeQuery> element and its  
1320 **AttributeQueryType** complex type:

```

1321 <element name="AttributeQuery" type="samlp:AttributeQueryType" />
1322 <complexType name="AttributeQueryType">
1323   <complexContent>
1324     <extension base="samlp:SubjectQueryAbstractType">
1325       <sequence>
1326         <element ref="saml:AttributeDesignator"
1327             minOccurs="0" maxOccurs="unbounded" />
1328       </sequence>
1329       <attribute name="Resource" type="anyURI" reference="
1330 use="optional" />
1331     </extension>
1332   </complexContent>
1333 </complexType>

```

### 1334 3.3.5. Element <AuthorizationDecisionQuery>

1335 The <AuthorizationDecisionQuery> element is used to make the query “Should these  
 1336 actions on this resource be allowed for this subject, given this evidence?” A successful response  
 1337 will be in the form of assertions containing authorization decision statements. This element is of  
 1338 type **AuthorizationDecisionQueryType**, which extends **SubjectQueryAbstractType** with the  
 1339 addition of the following elements and attribute:

1340 Resource [Required]  
 1341 A **URI** indicating the resource for which authorization is requested.  
 1342 <Actions> [**One or More Required**]  
 1343 The actions for which authorization is requested.  
 1344 <Evidence> [Any Number]  
 1345 An assertion that the responder MAY rely on in making its response.

1346 The following schema fragment defines the <AuthorizationDecisionQuery> element and its  
 1347 **AuthorizationDecisionQueryType** complex type:

```

1348 <element name="AuthorizationDecisionQuery"
1349 type="samlp:AuthorizationDecisionQueryType" />
1350 <complexType name="AuthorizationDecisionQueryType">
1351   <complexContent>
1352     <extension base="samlp:SubjectQueryAbstractType">
1353       <sequence>
1354         <element ref="saml:Actions" maxOccurs="unbounded" />
1355         <element ref="saml:Evidence"
1356             minOccurs="0" maxOccurs="unbounded" />
1357       </sequence>
1358       <attribute name="Resource" type="anyURI" use="required" />
1359     </extension>
1360   </complexContent>
1361 </complexType>

```

## 1362 3.4. Responses

1363 The following sections define the SAML constructs that contain response information.

### 1364 3.4.1. Complex Type ResponseAbstractType

1365 All SAML responses are of types that are derived from the abstract **ResponseAbstractType**  
 1366 complex type. This type defines common attributes and elements that are associated with all SAML  
 1367 responses:

1368 ResponseID [Required]  
 1369 An identifier for the response. It is of type **IDType**, and MUST follow the requirements  
 1370 specified by that type for identifier uniqueness.

1371 InResponseTo [Required]  
 1372 A reference to the identifier of the request to which the response corresponds. The value of  
 1373 this attribute MUST match the value of the corresponding RequestID attribute.  
 1374 MajorVersion [Required]  
 1375 The major version of this response. The identifier for the version of SAML defined in this  
 1376 specification is 1. Processing of this attribute is specified in Section 3.4.4.  
 1377 MinorVersion [Required]  
 1378 The minor version of this response. The identifier for the version of SAML defined in this  
 1379 specification is 0. Processing of this attribute is specified in Section 3.4.4.  
 1380 IssueInstant [Optional]  
 1381 The time instant of issue of the request. The time value is encoded in UTC as described in  
 1382 section 1.2.1.  
 1383 Recipient [Optional]  
 1384 The intended recipient of this response. This is useful to prevent malicious forwarding of  
 1385 responses to unintended recipients, a protection that is required by some use profiles. It is  
 1386 set by the generator of the response to a URI reference that identifies the intended  
 1387 recipient. If present, the actual recipient MUST check that the URI reference identifies the  
 1388 recipient or a resource managed by the recipient. If it does not, the response MUST be  
 1389 discarded.

1390  
 1391 <Signature> [Optional]  
 1392 An XML Signature that authenticates the assertion, see section 5.  
 1393 The following schema fragment defines the **ResponseAbstractType** complex type:  
 1394   <complexType name="ResponseAbstractType" abstract="true">  
 1395     <sequence>  
 1396       <element ref = "ds:Signature" minOccurs="0"/>  
 1397     </sequence>  
 1398     <attribute name="ResponseID" type="saml:IDType" use="required"/>  
 1399     <attribute name="InResponseTo" type="saml:IDReferenceType"  
 1400       use="required"/>  
 1401     <attribute name="MajorVersion" type="integer" use="required"/>  
 1402     <attribute name="MinorVersion" type="integer" use="required"/>  
 1403     <attribute name="IssueInstant" type="dateTime" use="required"/>  
 1404     <attribute name="Recipient" type="dateTime" use="optional"/>  
 1405   </complexType>

### 1406 3.4.2. Element <Response>

1407 The <Response> element specifies the status of the corresponding SAML request and a list of  
 1408 zero or more assertions that answer the request. It has the complex type **ResponseType**, which  
 1409 extends **ResponseAbstractType** by adding the following elements (in an unbounded mixture):

1410 <Status> [Required] (see Section 3.4.3)  
 1411 A code representing the status of the corresponding request.  
 1412 <Assertion> [Any Number] (see Section [2.3.22.3.3](#))  
 1413 Specifies an assertion by value.

1414 The following schema fragment defines the <Response> element and its **ResponseType** complex  
 1415 type:  
 1416   <element name="Response" type="samlp:ResponseType"/>  
 1417   <complexType name="ResponseType">  
 1418     <complexContent>  
 1419       <extension base="samlp:ResponseAbstractType">

```

1420         <sequence>
1421             <element ref="samlp:Status" />
1422             <element ref="saml:Assertion"
1423                     minOccurs="0" maxOccurs="unbounded" />
1424         </sequence>
1425     </extension>
1426 </complexContent>
1427 </complexType>
```

### 3.4.3. Element <Status>

The <Status> element :

<StatusCode> [Required]  
A code representing the status of the corresponding request.

<StatusMessage> [Any Number]  
A message which MAY be returned to an operator.

<StatusDetail> [Optional]  
Specifies additional information concerning an error condition.

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

```

<element name="Status" type="samlp:StatusType" />
<complexType name="StatusType">
    <sequence>
        <element ref="samlp:StatusCode" />
        <element ref="samlp:StatusMessage"
                minOccurs="0" maxOccurs="unbounded" />
        <element ref="samlp:StatusDetail" minOccurs="0" />
    </sequence>
</complexType>
```

#### 3.4.3.1. Element <StatusCode>

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

Value [Required]  
The status code value as defined below.

<SubStatusCode> [Optional]  
An optional subordinate status code value that provides more specific information on an error condition.

The following **StatusCode** values are defined:

Success  
The request succeeded.

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

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

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

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

```

<element name="StatusCode" type="samlp:StatusCodeType" />
<complexType name="StatusCodeType">
```

```

1467     <sequence>
1468         <element ref="samlp:SubStatusCode" minOccurs="0" />
1469     </sequence>
1470     <attribute name="Value" type="samlp:StatusCodeEnumType" use="required" />
1471 </complexType>
1472 <simpleType name="StatusCodeEnumType">
1473     <restriction base="QName">
1474         <enumeration value="samlp:Success"/>
1475         <enumeration value="samlp:VersionMismatch"/>
1476         <enumeration value="samlp:Receiver"/>
1477         <enumeration value="samlp:Sender"/>
1478     </restriction>
1479 </simpleType>

```

### 1480 3.4.3.2. Element <SubStatusCode>

1481 The <SubStatusCode> element specifies an additional code representing the status of the  
 1482 corresponding request:

1483 Value [Required]

1484     The status code value as defined below.

1485 <SubStatusCode> [Optional]

1486     An optional subordinate status code value that provides an additional level of specific  
 1487 information on an error condition.

1488 The following **SubStatusCode** values are defined, additional codes MAY be defined in future  
 1489 versions of the SAML specification:

1490 RequestVersionTooHigh

1491     The protocol version specified in the request is a major upgrade from the highest protocol  
 1492 version supported by the responder.

1493 RequestVersionTooLow

1494     The responder cannot respond to the particular request using the SAML version specified  
 1495 in the request because it is too low.

1496 RequestVersionDeprecated

1497     The responder does not respond to any requests with the protocol version specified in the  
 1498 request.

1499 TooManyResponses

1500     The response would contain more elements than the responder will return.

1501 The following schema fragment defines the <SubStatusCode> element and its

1502 **SubStatusCodeType** complex type:

```

1503 <element name="SubStatusCode" type="samlp:SubStatusCodeType" />
1504 <complexType name="SubStatusCodeType">
1505     <sequence>
1506         <element ref="samlp:SubStatusCode" minOccurs="0" />
1507     </sequence>
1508     <attribute name="Value" type="QName" use="required" />
1509 </complexType>

```

### 1510 3.4.3.3. Element <StatusMessage>

1511 The <StatusMessage> element specifies a message that MAY be returned to an operator:

1512 The following schema fragment defines the <StatusMessage> element and its  
 1513 **StatusMessageType** complex type:

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

1515    **3.4.3.4. Element <StatusDetail>**

1516    The <StatusDetail> element MAY be used to specify additional information concerning an error  
1517    condition.

1518    The following schema fragment defines the <StatusDetail> element and its **StatusDetailType**  
1519    complex type:

```
1520 <element name="StatusDetail" type="samlp:StatusDetailType" />
1521 <complexType name="StatusDetailType">
1522   <sequence>
1523     <any namespace="#any"
1524       processContents="lax" minOccurs="0" maxOccurs="unbounded" />
1525   </sequence>
1526 </complexType>
```

1527    **3.4.4. Responses to <AuthenticationQuery> and <AttributeQuery>**

1528    Responses to Authentication and Attribute queries are constructed by matching against the  
1529    <saml:Subject> element found within the <AuthenticationQuery> or <AttributeQuery>  
1530    elements. In response to these queries, every assertion returned by a SAML responder MUST  
1531    contain at least one statement whose <saml:Subject> element **strongly matches** the  
1532    <saml:Subject> element found in the query.

1533    A <saml:Subject> element S1 strongly matches S2 if and only if:

- 1    If S2 includes a <saml:NameIdentifier> element, then S1 must include an identical  
1535    <saml:NameIdentifier> element.
- 2    If S2 includes a <saml:SubjectConfirmation> element, then S1 must include an  
1537    identical <saml:SubjectConfirmation> element.

1538    If the responder cannot provide an assertion with any statement(s) satisfying the constraints  
1539    expressed by a query, the <saml:Response> element MUST NOT contain an <assertion> element  
1540    and MUST include a <saml:StatusCode> with value "Success". It MAY return a  
1541    <saml:StatusMessage> with additional information.

## 1542 4. SAML Versioning

1543 SAML version information appears in the following elements:

1544 ?? <Assertion>

1545 ?? <Request>

1546 ?? <Response>

1547 The version numbering of the SAML assertion is independent of the version number of the SAML  
1548 request-response protocol. The version information for each consists of a major version number  
1549 and a minor version number, both of which are integers. In accordance with industry practice a  
1550 version number SHOULD be presented to the user in the form *Major.Minor*. This document defines  
1551 SAML Assertions 1.0 and SAML Protocol 1.0.

1552 The version number  $\text{Major}_B.\text{Minor}_B$  is higher than the version number  $\text{Major}_A.\text{Minor}_A$  if and only if:

1553  $\text{Major}_B > \text{Major}_A \ ? \ ((\text{Major}_B = \text{Major}_A) \ ? \ \text{Minor}_B > \text{Minor}_A)$

1554 Each revision of SAML SHALL assign version numbers to assertions, requests, and responses that  
1555 are the same as or higher than the corresponding version number in the SAML version that  
1556 immediately preceded it.

1557 New versions of SAML SHALL assign new version numbers as follows:

1558 ?? **Documentation change:** ( $\text{Major}_B = \text{Major}_A$ ) ? ( $\text{Minor}_B > \text{Minor}_A$ )

1559 If the major and minor version numbers are unchanged, the new version *B* only introduces  
1560 changes to the documentation that raise no compatibility issues with an implementation of  
1561 version *A*.

1562 ?? **Minor upgrade:** ( $\text{Major}_B = \text{Major}_A$ ) ? ( $\text{Minor}_B > \text{Minor}_A$ )

1563 If the major version number of versions *A* and *B* are the same and the minor version  
1564 number of *B* is higher than that of *A*, the new SAML version MAY introduce changes to the  
1565 SAML schema and semantics but any changes that are introduced in *B* SHALL be  
1566 compatible with version *A*.

1567 ?? **Major upgrade:**  $\text{Major}_B > \text{Major}_A$

1568 If the major version of *B* number is higher than the major version of *A*, Version *B* MAY  
1569 introduce changes to the SAML schema and semantics that are incompatible with *A*.

### 1570 4.1. Assertion Version

1571 A SAML application MUST NOT issue any assertion whose version number is not supported.

1572 A SAML application MUST reject any assertion whose major version number is not supported.

1573 A SAML application MAY reject any assertion whose version number is higher than the highest  
1574 supported version.

### 1575 4.2. Request Version

1576 A SAML application SHOULD issue requests that specify the highest SAML version supported by  
1577 both the sender and recipient.

1578 If the SAML application does not know the capabilities of the recipient it should assume that it  
1579 supports the highest SAML version supported by the sender.

1580    **4.3. Response Version**

1581    A SAML application MUST NOT issue responses that specify a higher SAML version number than  
1582    the corresponding request.

1583    A SAML application MUST NOT issue a response that has a major version number that is lower  
1584    than the major version number of the corresponding request except to report the error  
1585    RequestVersionTooHigh.

1586    Incompatible protocol versions MAY cause the following errors to be reported:

1587    RequestVersionTooHigh

1588         The protocol version specified in the request is a major upgrade from the highest protocol  
1589         version supported by the responder.

1590    RequestVersionTooLow

1591         The responder cannot respond to the particular request using the SAML version specified  
1592         in the request because it is too low.

1593    RequestVersionDeprecated

1594         The responder does not respond to any requests with the protocol version specified in the  
1595         request.

## 5. SAML & XML-Signature Syntax and Processing

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

?? An Assertion signed by the issuer (AP). This supports :

(1) Message integrity

(2) Authentication of the issuer to a relying party

(3) If the signature is based on the issuer's public-private key pair, then it also provides for non-repudiation of origin.

?? A SAML request or a SAML response message signed by the message originator. This supports :

(1) Message integrity

(2) Authentication of message origin to a destination

(3) If the signature is based on the originator's public-private key pair, then it also provides for non-repudiation of origin.

Note :

?? SAML documents may be the subject of signatures from different packaging contexts. 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.

?? It is useful to characterize situations when a digital signature is NOT required in SAML.

Assertions:

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

Request/Response messages:

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

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

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

(1) An assertion obtained by a relying party from an entity other than the asserting party MUST be signed by the issuer.

(2) A SAML message arriving at a destination from an entity other than the originating site MUST be signed by the origin site.

### 5.1. Signing Assertions

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

1637 **5.2. Request/Response Signing**  
1638 All SAML requests and responses MAY be signed using the XML Signature. This is reflected in the  
1639 schema – Section 3.2 & 3.4.

## 1640 **5.3. Signature Inheritance**

1641 **5.3.1. Rationale**  
1642 SAML assertions may be embedded within request or response messages or other XML  
1643 messages, which may be signed. Request or response messages may themselves be contained  
1644 within other messages that are based on other XML messaging frameworks (e.g., SOAP) and the  
1645 composite object may be the subject of a signature. Another possibility is that SAML assertions or  
1646 request/response messages are embedded within a non-XML messaging object (e.g., MIME  
1647 package) and signed.  
1648 In such a case, the SAML sub-message (Assertion, request, response) may be viewed as inheriting  
1649 a signature from the "super-signature" over the enclosing object, provided certain constraints are  
1650 met.  
1651 (1) An assertion may be viewed as inheriting a signature from a super signature, if the super  
1652 signature applies all the elements within the assertion.  
1653 A SAML request or response may be viewed as inheriting a signature from a super signature, if the  
1654 super signature applies to all of the elements within the response.

1655 **5.3.2. Rules for SAML Signature Inheritance**  
1656 Signature inheritance occurs when SAML message (assertion/request/response) is not signed but  
1657 is enclosed within signed SAML such that the signature applies to all of the elements within the  
1658 message. In such a case, the SAML message is said to inherit the signature and may be  
1659 considered equivalent to the case where it is explicitly signed. The SAML message inherits the  
1660 "closest enclosing signature".  
1661 But if SAML messages need to be passed around by themselves, or embedded in other messages,  
1662 they would need to be signed as per section 5.1

## 1663 **5.4. XML Signature Profile**

1664 The XML Signature specification calls out a general XML syntax for signing data with many  
1665 flexibilities and choices. This section details the constraints on these facilities so that SAML  
1666 processors do not have to deal with the full generality of XML Signature processing.

1667 **5.4.1. Signing formats**  
1668 XML Signature has three ways of representing signature in a document viz: enveloping, enveloped  
1669 and detached.  
1670 SAML assertions and protocols MUST use the enveloped signatures for signing assertions and  
1671 protocols. SAML processors should support use of RSA signing and verification for public key  
1672 operations.

1673 **5.4.2. CanonicalizationMethod**  
1674 XML Signature REQUIRES the Canonical XML (omits comments)  
1675 (<http://www.w3.org/TR/2001/REC-xml-c14n-20010315>). SAML implementations SHOULD use  
1676 Canonical XML with no comments.

1677    **5.4.3. Transforms**

1678    REQUIRES the enveloped signature transform <http://www.w3.org/2000/09/xmldsig#enveloped-signature>

1680    **5.4.4. KeyInfo**

1681    SAML does not restrict or impose any restrictions in this area. Therefore following keyInfo may be  
1682    absent.

1683    **5.4.5. Binding between statements in a multi-statement assertion**

1684    Use of signing does not affect semantics of statements within assertions in any way, as stated in  
1685    this document Sections 1 through 4.

## 1686 6. SAML Extensions

1687 The SAML schemas support extensibility. An example of an application that extends SAML  
1688 assertions is the XTAML system for management of embedded trust roots . The following sections  
1689 explain how to use the extensibility features in SAML to create extension schemas.

1690 Note that elements in the SAML schemas are not blocked from substitution, so that all SAML  
1691 elements MAY serve as the head element of a substitution group. Also, types are not defined as  
1692 final, so that all SAML types MAY be extended and restricted. The following sections discuss  
1693 only elements that have been specifically designed to support extensibility.

### 1694 6.1. Assertion Schema Extension

1695 The SAML assertion schema is designed to permit separate processing of the assertion package  
1696 and the statements it contains, if the extension mechanism is used for either part.

1697 The following elements are intended specifically for use as extension points in an extension  
1698 schema; their types are set to abstract, so that the use of an xsi:type attribute with these  
1699 elements is REQUIRED:

```
1700    ?? <Assertion>
1701    ?? <Condition>
1702    ?? <Statement>
1703    ?? <SubjectStatement>
1704    ?? <AdviceElement>
```

1705 In addition, the following elements that are directly usable as part of SAML MAY be extended:

```
1706    ?? <AuthenticationStatement>
1707    ?? <AuthorizationDecisionStatement>
1708    ?? <AttributeStatement>
1709    ?? <AudienceRestrictionCondition>
```

1710 Finally, the following elements are defined to allow elements from arbitrary namespaces within  
1711 them, which serves as a built-in extension point without requiring an extension schema:

```
1712    ?? <AttributeValue>
1713    ?? <Advice>
```

### 1714 6.2. Protocol Schema Extension

1715 The following elements are intended specifically for use as extension points in an extension  
1716 schema; their types are set to abstract, so that the use of an xsi:type attribute with these  
1717 elements is REQUIRED:

```
1718    ?? <Query>
1719    ?? <SubjectQuery>
```

1720 In addition, the following elements that are directly usable as part of SAML MAY be extended:

```
1721    ?? <Request>
```

```
1722    ?? <AuthenticationQuery>
1723    ?? <AuthorizationDecisionQuery>
1724    ?? <AttributeQuery>
1725    ?? <Response>
```

## 1726 6.3. Use of Type Derivation and Substitution Groups

1727 W3C XML Schema provides two principal mechanisms for specifying an element of an extended  
1728 type: type derivation and substitution groups.

1729 For example, a `<Statement>` element can be assigned the type **NewStatementType** by means of  
1730 the `xsi:type` attribute. For such an element to be schema-valid, **NewStatementType** needs to be  
1731 derived from **StatementType**. The following example of a SAML assertion assumes that the  
1732 extension schema (represented by the `new:` prefix) has defined this new type:

```
1733 <saml:Assertion ...>
1734     <saml:Statement xsi:type="new:NewStatementType">
1735     ...
1736     </saml:Statement>
1737 </saml:Assertion>
```

1738 Alternatively, the extension schema can define a `<NewStatement>` element that is a member of a  
1739 substitution group that has `<Statement>` as a head element. For the substituted element to be  
1740 schema-valid, it needs to have a type that matches or is derived from the head element's type. The  
1741 following is an example of an extension schema fragment that defines this new element:

```
1742 <xsd:element "NewStatement" type="new:NewStatementType"
1743     substitutionGroup="saml:Statement" />
```

1744 The substitution group declaration allows the `<NewStatement>` element to be used anywhere the  
1745 SAML `<Statement>` element can be used. The following is an example of a SAML assertion that  
1746 uses the extension element:

```
1747 <saml:Assertion ...>
1748     <new:NewStatement>
1749     ...
1750     </new:NewStatement>
1751 </saml:Assertion>
```

1752 The choice of extension method has no effect on the semantics of the XML document but does  
1753 have implications for interoperability.

1754 The advantages of type derivation are as follows:

- 1755 ?? A document can be more fully interpreted by a parser that does not have access to the  
1756 extension schema because a “native” SAML element is available.
- 1757 ?? At the time of writing, some W3C XML Schema validators do not support substitution  
1758 groups, whereas the `xsi:type` attribute is widely supported.

1759 The advantage of substitution groups is that a document can be explained without the need to  
1760 explain the functioning of the `xsi:type` attribute.

## 7. SAML-Defined Identifiers

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

Where possible an existing URN is used to specify a protocol. In the case of IETF protocols the URN of the most current RFC that specifies the protocol is used. [URI](#)s created specifically for SAML have the initial stem:

<http://www.oasis-open.org/committees/security/docs/draft-sstc-core-28>

### 7.1. Authentication Method and Confirmation Method Identifiers

The `<AuthenticationMethod>` and `<SubjectConfirmationMethod>` elements perform different functions within the SAML architecture although both can contain some of the same values. `<AuthenticationMethod>` is a part of an Authentication Statement, which describes an authentication act which occurred in the past. The `<AuthenticationMethod>` indicates how that authentication was done. Note that the authentication statement does not provide the means to perform that authentication, such as a password, key or certificate.

In contrast, `<SubjectConfirmationMethod>` is a part of the `<SubjectConfirmation>`, which is used to allow the Relying Party to confirm that the request or message came from the System Entity that corresponds to the Subject in the statement. The `<SubjectConfirmationMethod>` indicates the method which the Relying Party can use to do this in the future. This may or may not have any relationship to an authentication that was performed previously. Unlike the Authentication Method, the `<SubjectConfirmationMethod>` will usually be accompanied with some piece of information, such as a certificate or key, which will allow the Relying Party to perform the necessary check.

There are many `<SubjectConfirmationMethod>`, because there are many different SAML usage scenarios. A few examples are:

1. A user logs in with a password, but a temporary passcode or cookie is issued for confirmation purposes to avoid repeated exposure of the long term password.
2. There is no login, but an application request is digitally signed. The associated public key is used for confirmation.
3. The user logs in using Kerberos and a Kerberos ticket is used subsequently for confirmation. Notice that in this case although both the Authentication Method and the `<SubjectConfirmationMethod>` are Kerberos, what happens at each step is actually different. (See RFC 1510)

The following identifiers ~~MAY be used in the `<ConfirmationMethod>` element~~ are defined (see [Section 2.4.2.3](#)) to refer to common authentication protocols.

#### 7.1.1. SAML Artifact:

URI: <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-28#artifact>

`<SubjectConfirmationData>`: Base64 (*Artifact*)

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

1801 **7.1.2. SAML Artifact (SHA-1):**  
1802 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-28#artifact-sha1>  
1803 <SubjectConfirmationData>: *Base64 ( SHA1 ( Artifact ) )*  
1804 The subject of the assertion is the party that can present a SAML Artifact such that the SHA1 digest  
1805 of the specified artifact matches the value specified in <SubjectConfirmationData>.

1806 **7.1.3. Holder of Key:**  
1807 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-28#Holder-Of-Key>  
1808 <ds:KeyInfo>: Any cryptographic key  
1809 The subject of the assertion is the party that can demonstrate that it is the holder of the private  
1810 component of the key specified in <ds:KeyInfo>.

1811 **7.1.4. Bearer Indication:**  
1812 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-28#BearerIndication>  
1813 The subject of the assertion is the bearer of the assertion.

1814 **7.1.4.7.1.5. Sender Vouches:**  
1815 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-28#sender-vouches>  
1816 Indicates that no other information is available about the context of use of the assertion. The  
1817 Relying party SHOULD utilize other means to determine if it should process the assertion further.

1818 **7.1.5.7.1.6. Password (Pass-Through):**  
1819 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-28#password>  
1820 <SubjectConfirmationData>: *Base64 ( Password )*  
1821 The subject of the assertion is the party that can present the password value specified in  
1822 <SubjectConfirmationData>.  
1823 The username of the subject is specified by means of the <NameIdentifier> element.

1824 **7.1.6.7.1.7. Password (One-Way-Function SHA-1):**  
1825 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-28#password-sha1>  
1826 <SubjectConfirmationData>: *Base64 ( SHA1 ( Password ) )*  
1827 The subject of the assertion is the party that can present the password such that the SHA1 digest of  
1828 the specified password matches the value specified in <SubjectConfirmationData>.  
1829 The username of the subject is specified by means of the <NameIdentifier> element.

1830 **7.1.7.7.1.8. Kerberos**  
1831 **URI:** <urn:ietf:rfc:1510>  
1832 <SubjectConfirmationData>: A Kerberos Ticket

1833 The subject is authenticated by means of the Kerberos protocol , an instantiation of the Needham-Schroeder symmetric key authentication mechanism [Needham78].

1835 **7.1.8.7.1.9. SSL/TLS Certificate Based Client Authentication:**

1836 **URI:** urn:ietf:rfc:2246

1837 <ds:KeyInfo>: Any cryptographic key

1838 **7.1.9.7.1.10. Object Authenticator (SHA-1):**

1839 **URI:** http://www.oasis-open.org/committees/security/docs/draft-sstc-core-28#object-sha1

1840 <SubjectConfirmationData>: Base64( SHA1( Object ) )

1841 This authenticator element is the result of computing a digest, using the SHA-1 hash algorithm. It is used when the subject can be represented as a binary string, for example when it is an XML document or the disk image of executable code. Any preprocessing of the subject prior to computation of the digest is out of scope. The name of the subject should be conveyed in an accompanying NameIdentifier element.

1846 **7.1.10.7.1.11. PKCS#7**

1847 **URI:** urn:ietf:rfc:2315

1848 <SubjectConfirmationData>: Base64( PKCS#7( Object ) )

1849 This authenticator element is signed data in PKCS#7 format [PKCS#7]. The posited identity of the signer must be conveyed in an accompanying NameIdentifier element. This subject type may be included in the subject field of an authentication query, in which case the corresponding response indicates whether the posited signer is, indeed, the signer. It may be included in an attribute query, in which case, the requested attribute values for the subject authenticated by the signed data are returned. It may be included in an authorization query, in which case, the access request represented by the signed data shall be identified by the accompanying object element, and the corresponding assertion containing an authorization decision statement assertion indicates whether the signer is authorized for the access request represented by the object element.

1858 **7.1.11.7.1.12. Cryptographic Message Syntax**

1859 **URI:** urn:ietf:rfc:2630

1860 <SubjectConfirmationData>: Base64( CMS( Object ) )

1861 This authenticator element is signed data in CMS format [CMS]. See also [7.1.11.7.1.10](#)

1862 **7.1.12.7.1.13. XML Digital Signature**

1863 **URI:** urn:ietf:rfc:3075

1864 <SubjectConfirmationData>: Base64( XML-SIG( Object ) )

1865 <ds:KeyInfo>: A cryptographic signing key

1866 This authenticator element is signed data in XML Signature format. See also [7.1.11.7.1.10](#)

1867 **7.2. Action Namespace Identifiers**

1868 The following identifiers MAY be used in the `ActionNamespace` attribute (see Section 2.4.4.1) to  
1869 refer to common sets of actions to perform on resources.

1870 **7.2.1. Read/Write/Execute/Delete/Control:**

1871 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-28#rwedc>

1872 Defined actions:

1873     Read Write Execute Delete Control

1874 These actions are interpreted in the normal manner, i.e.

1875     Read

1876         The subject may read the resource

1877     Write

1878         The subject may modify the resource

1879     Execute

1880         The subject may execute the resource

1881     Delete

1882         The subject may delete the resource

1883     Control

1884         The subject may specify the access control policy for the resource

1885 **7.2.2. Read/Write/Execute/Delete/Control with Negation:**

1886 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-28#rwedc-negation>

1887 Defined actions:

1888     Read Write Execute Delete Control ~Read ~Write ~Execute ~Delete ~Control

1889 The actions specified in section 7.2.1 are interpreted in the same manner described there. Actions  
1890 prefixed with a tilde ~ are negated permissions and are used to affirmatively specify that the stated  
1891 permission is denied. Thus a subject described as being authorized to perform the action ~Read is  
1892 affirmatively denied read permission.

1893 An application MUST NOT authorize both an action and its negated form.

1894 **7.2.3. Get/Head/Put/Post:**

1895 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-28#ghpp>

1896 Defined actions:

1897 GET HEAD PUT POST

1898 These actions bind to the corresponding HTTP operations. For example a subject authorized to  
1899 perform the GET action on a resource is authorized to retrieve it.

1900 The GET and HEAD actions loosely correspond to the conventional read permission and the PUT  
1901 and POST actions to the write permission. The correspondence is not exact however since a HTTP  
1902 GET operation may cause data to be modified and a POST operation may cause modification to a  
1903 resource other than the one specified in the request. For this reason a separate Action **URI**  
1904 **reference** specifier is provided.

1905 **7.2.4. UNIX File Permissions:**

1906 **URI:** <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-28#unix>

1907 The defined actions are the set of UNIX file access permissions expressed in the numeric (octal)  
1908 notation.

1909 The action string is a four digit numeric code:

1910       *extended user group world*

1911 Where the *extended* access permission has the value

1912       +2 if sgid is set

1913       +4 if suid is set

1914 The *user group* and *world* access permissions have the value

1915       +1 if execute permission is granted

1916       +2 if write permission is granted

1917       +4 if read permission is granted

1918 For example 0754 denotes the UNIX file access permission: user read, write and execute, group  
1919 read and execute and world read.

## 1920 8. SAML Schema Listings

1921 The following sections contain complete listings of the assertion and protocol schemas for SAML.

### 1922 8.1. Assertion Schema

1923 Following is a complete listing of the SAML assertion schema .

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Phill Hallam-Baker
(VeriSign Inc.) -->
<schema
    targetNamespace="http://www.oasis-open.org/committees/security/docs/draft-
sstc-schema-assertion-28.xsd"
    xmlns="http://www.w3.org/2001/XMLSchema" xmlns:saml="http://www.oasis-
open.org/committees/security/docs/draft-sstc-schema-assertion-28.xsd"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    elementFormDefault="unqualified">
    <import namespace="http://www.w3.org/2000/09/xmldsig#"
        schemaLocation="xmldsig-core-schema.xsd"/>
    <annotation>
        <documentation>draft-sstc-schema-assertion-28.xsd</documentation>
    </annotation>
    <simpleType name="IDType">
        <restriction base="string"/>
    </simpleType>
    <simpleType name="IDReferenceType">
        <restriction base="string"/>
    </simpleType>
    <simpleType name="DecisionType">
        <restriction base="string">
            <enumeration value="Permit"/>
            <enumeration value="Deny"/>
            <enumeration value="Indeterminate"/>
        </restriction>
    </simpleType>
    <element name="AssertionSpecifier" type="saml:AssertionSpecifierType"/>
    <complexType name="AssertionSpecifierType">
        <choice>
            <element ref="saml:AssertionIDReference"/>
            <element ref="saml:Assertion"/>
        </choice>
    </complexType>
    <element name="AssertionIDReference" type="saml:IDReferenceType"/>
    <element name="Assertion" type="saml:AssertionType"/>
    <complexType name="AssertionType">
        <sequence>
            <element ref="saml:Conditions" minOccurs="0"/>
            <element ref="saml:Advice" minOccurs="0"/>
            <choice maxOccurs="unbounded">
                <element ref="saml:Statement"/>
                <element ref="saml:SubjectStatement"/>
                <element ref="saml:AuthenticationStatement"/>
                <element ref="saml:AuthorizationDecisionStatement"/>
                <element ref="saml:AttributeStatement"/>
            </choice>
            <element ref="ds:Signature" minOccurs="0"/>
        </sequence>
        <attribute name="MajorVersion" type="integer" use="required"/>
        <attribute name="MinorVersion" type="integer" use="required"/>
        <attribute name="AssertionID" type="saml:IDType" use="required"/>
```

```

1977      <attribute name="Issuer" type="string" use="required"/>
1978          <attribute name="IssueInstant" type="dateTime" use="required"/>
1979      </complexType>
1980      <element name="Conditions" type="saml:ConditionsType" />
1981      <complexType name="ConditionsType">
1982          <choice minOccurs="0" maxOccurs="unbounded">
1983              <element ref="saml:Condition" />
1984              <element ref="saml:AudienceRestrictionCondition" />
1985          </choice>
1986          <attribute name="NotBefore" type="dateTime" use="optional" />
1987          <attribute name="NotOnOrAfter" type="dateTime" use="optional" />
1988      </complexType>
1989      <element name="Condition" type="saml:ConditionAbstractType" />
1990      <complexType name="ConditionAbstractType" abstract="true" />
1991      <element name="AudienceRestrictionCondition" type="saml:AudienceRestrictionConditionType" />
1992      <complexType name="AudienceRestrictionConditionType" >
1993          <complexContent>
1994              <extension base="saml:ConditionAbstractType" >
1995                  <sequence>
1996                      <element ref="saml:Audience" maxOccurs="unbounded" />
1997                  </sequence>
1998              </extension>
1999          </complexContent>
2000      </complexType>
2001      <element name="Audience" type="anyURI" />
2002      <element name="TargetRestrictionCondition" type="saml:TargetRestrictionConditionType" />
2003      <complexType name="TargetRestrictionConditionType" >
2004          <complexContent>
2005              <extension base="saml:ConditionAbstractType" >
2006                  <sequence>
2007                      <element ref="saml:Target" minOccurs="1" maxOccurs="unbounded" />
2008                  </sequence>
2009              </extension>
2010          </complexContent>
2011      </complexType>
2012      <element name="Advice" type="saml:AdviceType" />
2013      <complexType name="AdviceType" >
2014          <choice minOccurs="0" maxOccurs="unbounded" >
2015              <element ref="saml:AssertionIDReference" />
2016              <element ref="saml:Assertion" />
2017                  <complexType name="AssertionSpecifier" >
2018                      <element ref="saml:AdviceElement" />
2019                      <any namespace="##other" processContents="lax" />
2020                  </choice>
2021          </complexType>
2022          <element name="AdviceElement" type="saml:AdviceAbstractType" />
2023          <complexType name="AdviceAbstractType" />
2024          <element name="Statement" type="saml:StatementAbstractType" />
2025          <complexType name="StatementAbstractType" abstract="true" />
2026          <element name="SubjectStatement" type="saml:SubjectStatementAbstractType" />
2027          <complexType name="SubjectStatementAbstractType" abstract="true" >
2028              <complexContent>
2029                  <extension base="saml:StatementAbstractType" >
2030                      <sequence>
2031                          <element ref="saml:Subject" />
2032                      </sequence>
2033                  </extension>
2034              </complexContent>
2035          </complexType>
2036      </complexType>
2037      <element name="Statement" type="saml:StatementAbstractType" />
2038      <complexType name="StatementAbstractType" abstract="true" >
2039          <complexContent>

```

```

2040      </complexContent>
2041  </complexType>
2042  <element name="Subject" type="saml:SubjectType" />
2043  <complexType name="SubjectType">
2044    <choice>
2045      <sequence>
2046        <element ref="saml:NameIdentifier"/>
2047        <element ref="saml:SubjectConfirmation" minOccurs="0" />
2048      </sequence>
2049      <element ref="saml:SubjectConfirmation" />
2050    </choice>
2051  </complexType>
2052  <element name="NameIdentifier" type="saml:NameIdentifierType" />
2053  <complexType name="NameIdentifierType">
2054    <simpleContent>
2055      <extension base="string">
2056        <attribute name="NameQualifier" type="string" use="optional" />
2057        <attribute name="Format" type="anyURI" use="optional" />
2058      </extension>
2059    </simpleContent>
2060  </complexType>—<element name="NameIdentifier"
2061 type="saml:NameIdentifierType" />
2062  <complexType name="NameIdentifierType">
2063    <attribute name="SecurityDomain" type="string" />
2064    <attribute name="Name" type="string" use="required" />
2065  </complexType>
2066  <element name="SubjectConfirmation" type="saml:SubjectConfirmationType" />
2067  <complexType name="SubjectConfirmationType">
2068    <sequence>
2069      <element ref="saml:ConfirmationMethod" maxOccurs="unbounded" />
2070      <element ref="saml:SubjectConfirmationData" minOccurs="0" />
2071      <element ref="ds:KeyInfo" minOccurs="0" />
2072    </sequence>
2073  </complexType>
2074  <element name="SubjectConfirmationData" type="string" />
2075  <element name="ConfirmationMethod" type="anyURI" />
2076  <element name="AuthenticationStatement"
2077    type="saml:AuthenticationStatementType" />
2078  <complexType name="AuthenticationStatementType">
2079    <complexContent>
2080      <extension base="saml:SubjectStatementAbstractType" >
2081        <sequence>
2082          <element ref="saml:AuthenticationLocality" minOccurs="0" />
2083          <element ref="saml:AuthorityBinding"
2084            minOccurs="0" maxOccurs="unbounded" />
2085        </sequence>
2086        <attribute name="AuthenticationMethod" type="anyURI" />
2087        <attribute name="AuthenticationInstant" type="dateTime" />
2088      </extension>
2089    </complexContent>
2090  </complexType>
2091  <element name="AuthenticationLocality"
2092    type="saml:AuthenticationLocalityType" />
2093  <complexType name="AuthenticationLocalityType">
2094    <attribute name="IPAddress" type="string" use="optional" />
2095    <attribute name="DNSAddress" type="string" use="optional" />
2096  </complexType>
2097  <element name="AuthorityBinding" type="saml:AuthorityBindingType" />
2098  <complexType name="AuthorityBindingType">
2099    <attribute name="AuthorityKind" type="saml:AuthorityKindTypeQName"
2100      use="required" />
2101    <attribute name="Location" type="anyURI" use="required" />
2102    <attribute name="Binding" type="anyURI" use="required" />

```

```

2103    </complexType>
2104    <simpleType name="AuthorityKindType">
2105        <restriction base="string">
2106            <enumeration value="authentication"/>
2107            <enumeration value="attribute"/>
2108            <enumeration value="authorization"/>
2109        </restriction>
2110    </simpleType>
2111    <element name="AuthorizationDecisionStatement"
2112 type="saml:AuthorizationDecisionStatementType" />
2113    <complexType name="AuthorizationDecisionStatementType">
2114        <complexContent>
2115            <extension base="saml:SubjectStatementAbstractType" >
2116                <sequence>
2117                    <element ref="saml:Action" maxOccurs="unbounded"/>
2118                    <element ref="saml:Evidence" minOccurs="0" />
2119                </sequence>
2120                <attribute name="Resource" type="anyURI" use="required" />
2121                <attribute name="Decision" type="saml:DecisionType" use="required" />
2122            </extension>
2123        </complexContent>
2124    </complexType>
2125    <element name="Action" type="saml:ActionType" />
2126    <complexType name="ActionType">
2127        <simpleContent>
2128            <extension base="string">
2129                <attribute name="Namespace" type="anyURI" />
2130            </extension>
2131        </simpleContent>
2132    </complexType>
2133    <element name="Evidence" type="saml:EvidenceType" />—<element
2134 name="AuthorizationDecisionStatement"
2135 type="saml:AuthorizationDecisionStatementType" />
2136    <complexType name="AuthorizationDecisionStatementType">
2137        <complexContent>
2138            <extension base="saml:SubjectStatementAbstractType" >
2139                <sequence>
2140                    <element ref="saml:Actions" />
2141                    <element ref="saml:Evidence"
2142                         minOccurs="0" maxOccurs="unbounded" />
2143                </sequence>
2144                <attribute name="Resource" type="anyURI" use="required" />
2145                <attribute name="Decision"
2146                     type="saml:DecisionType" use="required" />
2147            </extension>
2148        </complexContent>
2149    </complexType>
2150    <element name="Actions" type="saml:ActionsType" />
2151    <complexType name="ActionsType">
2152        <sequence>
2153            <element ref="saml:Action" maxOccurs="unbounded" />
2154        </sequence>
2155            <attribute name="Namespace" type="anyURI" use="optional" />
2156        </complexType>
2157        <element name="Action" type="string" />
2158        <complexType name="EvidenceType" >
2159            <choice maxOccurs="unbounded" >
2160                <element ref="saml:AssertionIDReference" />
2161                <element ref="saml:Assertion" />
2162            </choice>
2163        </complexType>—<element name="Evidence" type="saml:AssertionSpecifierType" />
2164        <element name="AttributeStatement" type="saml:AttributeStatementType" />
2165        <complexType name="AttributeStatementType" >

```

```

2166     <complexContent>
2167         <extension base="saml:SubjectStatementAbstractType">
2168             <sequence>
2169                 <element ref="saml:Attribute" maxOccurs="unbounded" />
2170             </sequence>
2171         </extension>
2172     </complexContent>
2173 </complexType>
2174 <element name="AttributeDesignator" type="saml:AttributeDesignatorType" />
2175 <complexType name="AttributeDesignatorType">
2176     <attribute name="AttributeName" type="string" use="required" />
2177     <attribute name="AttributeNamespace" type="anyURI" use="required" />
2178 </complexType>
2179 <element name="Attribute" type="saml:AttributeType" />
2180 <complexType name="AttributeType">
2181     <complexContent>
2182         <extension base="saml:AttributeDesignatorType" >
2183             <sequence>
2184                 <element ref="saml:AttributeValue" maxOccurs="unbounded" />
2185             </sequence>
2186         </extension>
2187     </complexContent>
2188 </complexType>
2189 <element name="AttributeValue" type="saml:anyType" />
2190 </schema>

```

## 8.2. Protocol Schema

Following is a complete listing of the SAML protocol schema .

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Phill Hallam-Baker
(VeriSign Inc.) -->
<schema
    targetNamespace="http://www.oasis-open.org/committees/security/docs/draft-
sstc-schema-protocol-28.xsd"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:saml="http://www.oasis-open.org/committees/security/docs/draft-sstc-
schema-assertion-28.xsd"
    xmlns:samlp="http://www.oasis-open.org/committees/security/docs/draft-sstc-
schema-protocol-28.xsd"
    xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="unqualified">
    <import
        namespace="http://www.oasis-open.org/committees/security/docs/draft-sstc-
schema-assertion-28.xsd"
        schemaLocation="draft-sstc-schema-assertion-28.xsd" />
    <import
        namespace="http://www.w3.org/2000/09/xmldsig#"
        schemaLocation="xmldsig-core-schema.xsd" />
    <annotation>
        <documentation>draft-sstc-schema-protocol-28.xsd</documentation>
    </annotation>
    <complexType name="RequestAbstractType" abstract="true">
        <sequence>
            <element ref="samlp:RespondWith"
                    minOccurs="0" maxOccurs="unbounded" />
            <element ref = "ds:Signature" minOccurs="0" />
        </sequence>
        <attribute name="RequestID" type="saml:IDType" use="required" />
        <attribute name="MajorVersion" type="integer" use="required" />
        <attribute name="MinorVersion" type="integer" use="required" />
        <attribute name="IssueInstant" type="dateTime" use="required" />
        <attribute name="Recipient" type="dateTime" use="optional" />
    </complexType>

```

```

2226 |     <element name="RespondWith" type="anyURIQName" />
2227 |     <element name="Request" type="samlp:RequestType"/>
2228 |     <complexType name="RequestType">
2229 |         <complexContent>
2230 |             <extension base="samlp:RequestAbstractType">
2231 |                 <choice>
2232 |                     <element ref="samlp:Query" />
2233 |                     <element ref="samlp:SubjectQuery" />
2234 |                     <element ref="samlp:AuthenticationQuery" />
2235 |                     <element ref="samlp:AttributeQuery" />
2236 |                     <element ref="samlp:AuthorizationDecisionQuery" />
2237 |                     <element ref="saml:AssertionID" maxOccurs="unbounded" />
2238 |                     <element ref="samlp:AssertionArtifact" maxOccurs="unbounded" />
2239 |                 </choice>
2240 |             </extension>
2241 |         </complexContent>
2242 |     </complexType>
2243 |     <element name="AssertionArtifact" type="string" />
2244 |     <element name="Query" type="samlp:QueryAbstractType" />
2245 |     <complexType name="QueryAbstractType" abstract="true" />
2246 |     <element name="SubjectQuery" type="samlp:SubjectQueryAbstractType" />
2247 |     <complexType name="SubjectQueryAbstractType" abstract="true" >
2248 |         <complexContent>
2249 |             <extension base="samlp:QueryAbstractType" >
2250 |                 <sequence>
2251 |                     <element ref="saml:Subject" />
2252 |                 </sequence>
2253 |             </extension>
2254 |         </complexContent>
2255 |     </complexType>
2256 |     <element name="AuthenticationQuery" type="samlp:AuthenticationQueryType" />
2257 |     <complexType name="AuthenticationQueryType">
2258 |         <complexContent>
2259 |             <extension base="samlp:SubjectQueryAbstractType" >
2260 |                 <sequence>
2261 |                     <element ref="saml:ConfirmationMethod" minOccurs="0" />
2262 |                 </sequence>
2263 |             </extension>
2264 |         </complexContent>
2265 |     </complexType>
2266 |     <element name="AttributeQuery" type="samlp:AttributeQueryType" />
2267 |     <complexType name="AttributeQueryType">
2268 |         <complexContent>
2269 |             <extension base="samlp:SubjectQueryAbstractType" >
2270 |                 <sequence>
2271 |                     <element ref="saml:AttributeDesignator"
2272 |                           minOccurs="0" maxOccurs="unbounded" />
2273 |                 </sequence>
2274 |                 <attribute name="Resource" type="anyURI" use="optional" />
2275 |             </extension>
2276 |         </complexContent>
2277 |     </complexType>
2278 |     <element name="AuthorizationDecisionQuery"
2279 |           type="samlp:AuthorizationDecisionQueryType" />
2280 |     <complexType name="AuthorizationDecisionQueryType">
2281 |         <complexContent>
2282 |             <extension base="samlp:SubjectQueryAbstractType" >
2283 |                 <sequence>
2284 |                     <element ref="saml:Action" maxOccurs="unbounded" />
2285 |                     <element ref="saml:Evidence"
2286 |                           minOccurs="0" maxOccurs="unbounded" />
2287 |                 </sequence>
2288 |                 <attribute name="Resource" type="anyURI" use="required" />

```

```

2289         </extension>
2290     </complexContent>
2291   </complexType>
2292   <complexType name="ResponseAbstractType" abstract="true" >
2293     <sequence>
2294       <element ref = "ds:Signature" minOccurs="0"/>
2295     </sequence>
2296     <attribute name="ResponseID" type="saml:IDType" use="required"/>
2297     <attribute name="InResponseTo" type="saml:IDReferenceType"
2298       use="required"/>
2299     <attribute name="MajorVersion" type="integer" use="required"/>
2300     <attribute name="MinorVersion" type="integer" use="required"/>
2301     <attribute name="IssueInstant" type="dateTime" use="required"/>
2302   </complexType>
2303
2304   <element name="Response" type="samlp:ResponseType" />
2305   <complexType name="ResponseType">
2306     <complexContent>
2307       <extension base="samlp:ResponseAbstractType">
2308         <sequence>
2309           <element ref="samlp:Status" />
2310           <element ref="saml:Assertion"
2311             minOccurs="0" maxOccurs="unbounded" />
2312         </sequence>
2313       </extension>
2314     </complexContent>
2315   </complexType>
2316   <element name="Status" type="samlp:StatusType" />
2317   <complexType name="StatusType">
2318     <sequence>
2319       <element ref="samlp:StatusCode" />
2320       <element ref="samlp:StatusMessage"
2321         minOccurs="0" maxOccurs="unbounded" />
2322       <element ref="samlp:StatusDetail" minOccurs="0" />
2323     </sequence>
2324   </complexType>
2325   <element name="StatusCode" type="samlp:StatusCodeType" />
2326   <complexType name="StatusCodeType">
2327     <sequence>
2328       <element ref="samlp:SubStatusCode" minOccurs="0" />
2329     </sequence>
2330     <attribute name="Value" type="samlp:StatusCodeEnumType" use="required"/>
2331   </complexType>
2332   <simpleType name="StatusCodeEnumType">
2333     <restriction base="QName">
2334       <enumeration value="samlp:Success" />
2335       <enumeration value="samlp:VersionMismatch" />
2336       <enumeration value="samlp:Receiver" />
2337       <enumeration value="samlp:Sender" />
2338     </restriction>
2339   </simpleType>
2340   <element name="SubStatusCode" type="samlp:SubStatusCodeType" />
2341   <complexType name="SubStatusCodeType">
2342     <sequence>
2343       <element ref="samlp:SubStatusCode" minOccurs="0" />
2344     </sequence>
2345     <attribute name="Value" type="QName" use="required"/>
2346   </complexType>
2347   <element name="StatusMessage" type="string" />
2348   <element name="StatusDetail" type="samlp:StatusDetailType" />
2349   <complexType name="StatusDetailType">
2350     <sequence>
2351       <any namespace="##any" />

```

```
2352           processContents="lax" minOccurs="0" maxOccurs="unbounded" />
2353       </sequence>
2354     </complexType>
2355   </schema>
2356
```

## 2357 9. References

- 2358 [Kern-84] B. Kernighan, Rob Pike *The UNIX Programming Environment*, (March  
2359 1984) Prentice Hall Computer Books;
- 2360 [Needham78] R. Needham et al., *Using Encryption for Authentication in Large Networks*  
2361 *of Computers*, Communications of the ACM, Vol. 21 (12), pp. 993-999,  
2362 December 1978.
- 2363 [PKCS1] B. Kaliski, *PKCS #1: RSA Encryption Version 2.0*, RSA Laboratories, also  
2364 IETF RFC 2437, October 1998. <http://www.ietf.org/rfc/rfc2437.txt>
- 2365 [PKCS7] B. Kaliski.. "PKCS #7: Cryptographic Message Syntax, Version 1.5.", RFC  
2366 2315, March 1998.
- 2367 [RFC 1510] J. Kohl, C. Neuman. *The Kerberos Network Authentication Requestor (V5)*,  
2368 September 1993. <http://www.ietf.org/rfc/rfc1510.txt>
- 2369 [RFC 2246] T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. January 1999.  
2370 <http://www.ietf.org/rfc/rfc2246.txt>
- 2371 [RFC 2396] T. Berners-Lee et. al., *Uniform Resource Identifiers (URI): Generic Syntax*  
2372 <http://www.ietf.org/rfc/rfc2396.txt> IETF?
- 2373 [RFC 2630] R. Housley. *Cryptographic Message Syntax*. June 1999.  
2374 <http://www.ietf.org/rfc/rfc630.txt>
- 2375 [RFC 2648] R. Moats. *A URN Namespace for IETF Documents*. August 1999.  
2376 <http://www.ietf.org/rfc/rfc2648.txt>
- 2377 [RFC 3075] D. Eastlake, J. Reagle, D. Solo. *XML-Signature Syntax and Processing*,  
2378 March 2001. <http://www.ietf.org/rfc/rfc3075.txt>
- 2379 [RFC2104] H. Krawczyk et al., *HMAC: Keyed Hashing for Message Authentication*,  
2380 <http://www.ietf.org/rfc/rfc2104.txt>, IETF RFC 2104, February 1997.
- 2381 [RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,  
2382 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997
- 2383 [SAMLBind] P. Mishra et al., *Bindings and Profiles for the OASIS Security Assertion  
2384 Markup Language (SAML)*, [http://www.oasis-](http://www.oasis-open.org/committees/security/docs/draft-sstc-bindings-model-07.pdf)  
2385 [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),  
2386 OASIS, December 2001.
- 2387 [SAMLConform] **TBS**
- 2388 [SAMLGloss] J. Hodges et al., *Glossary for the OASIS Security Assertion Markup  
2389 Language (SAML)*, [http://www.oasis-](http://www.oasis-open.org/committees/security/docs/draft-sstc-glossary-02.pdf)  
2390 [open.org/committees/security/docs/draft-sstc-glossary-02.pdf](http://www.oasis-open.org/committees/security/docs/draft-sstc-glossary-02.pdf), OASIS,  
2391 December 2001.
- 2392 [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)  
2393 [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),  
2394 OASIS, December 2001.
- 2395 [SAMLSecure] **TBS**
- 2396 [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)  
2397 [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),  
2398 OASIS, December 2001.
- 2399 [Schema1] H. S. Thompson et al., *XML Schema Part 1: Structures*,  
2400 <http://www.w3.org/TR/xmlschema-1/>, World Wide Web Consortium  
2401 Recommendation, May 2001.

- 2402 [Schema2] P. V. Biron et al., *XML Schema Part 2: Datatypes*,  
 2403 <http://www.w3.org/TR/xmlschema-2>, World Wide Web Consortium  
 2404 Recommendation, May 2001.
- 2405 [UNICODE-CI] M. Davis, M. J. Dürst, <http://www.unicode.org/unicode/reports/tr15/tr15-21.html>, UNICODE Consortium
- 2406 [W3C-CHAR] M. J. Dürst, *Requirements for String Identity Matching and String Indexing*  
 2407 <http://www.w3.org/TR/WD-charreq>, World Wide Web Consortium.
- 2408 [W3C-CharMod] M. J. Dürst., *Unicode Normalization Forms*  
 2409 <http://www.w3.org/TR/charmod/>, World Wide Web Consortium.
- 2410 [XML] T. Bray et. al. *Extensible Markup Language (XML) 1.0 (Second Edition)*,  
 2411 <http://www.w3.org/TR/REC-xml>, World Wide Web Consortium.
- 2412 [XMLEnc] *XML Encryption Specification*, In development.
- 2413 [XMLSig] D. Eastlake et al., *XML-Signature Syntax and Processing*,  
 2414 <http://www.w3.org/TR/xmldsig-core/>, World Wide Web Consortium.
- 2415 [XMLSig-XSD] XML Signature Schema available from <http://www.w3.org/TR/2000/CR-xmldsig-core-20001031/xmldsig-core-schema.xsd>.
- 2416 [XTAML] P. Hallam-Baker, *XML Trust Axiom Markup Language 1.0*,  
 2417 <http://www.xmltrustcenter.org/>, VeriSign Inc. September 2001.
- 2418 [Needham78] R. Needham et al., *Using Encryption for Authentication in Large Networks of Computers*, Communications of the ACM, Vol. 21 (12), pp. 993-999, December 1978.
- 2419 [Kern-84] B. Kernighan, Rob Pike *The UNIX Programming Environment*, (March 1984) Prentice Hall Computer Books;
- 2420 [PKCS1] B. Kaliski, *PKCS #1: RSA Encryption Version 2.0*, RSA Laboratories, also IETF RFC 2437, October 1998. <http://www.ietf.org/rfc/rfc2437.txt>
- 2421 [PKCS7] B. Kaliski., "PKCS #7: Cryptographic Message Syntax, Version 1.5.", RFC 2315, March 1998.
- 2422 [RFC 1510] J. Kohl, C. Neuman. *The Kerberos Network Authentication Requestor (V5)*, September 1993. <http://www.ietf.org/rfc/rfc1510.txt>
- 2423 [RFC 2246] T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. January 1999. <http://www.ietf.org/rfc/rfc2246.txt>
- 2424 [RFC 2630] R. Housley. *Cryptographic Message Syntax*. June 1999. <http://www.ietf.org/rfc/rfc630.txt>
- 2425 [RFC 2648] R. Moats. *A URN Namespace for IETF Documents*. August 1999. <http://www.ietf.org/rfc/rfc2648.txt>
- 2426 [RFC 3075] D. Eastlake, J. Reagle, D. Solo. *XML-Signature Syntax and Processing*. March 2001. <http://www.ietf.org/rfc/rfc3075.txt>
- 2427 [RFC2104] H. Krawczyk et al., *HMAC: Keyed Hashing for Message Authentication*, <http://www.ietf.org/rfc/rfc2104.txt>, IETF RFC 2104, February 1997.
- 2428 [RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997
- 2429 [SAMLBind] P. Mishra et al., *Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML)*, <http://www.oasis-open.org/committees/security/docs/draft-sstc-bindings-model-07.pdf>, OASIS, December 2001.
- 2430 [SAMLConform] *TBS*
- 2431 [SAMLGloss] J. Hedges et al., *Glossary for the OASIS Security Assertion Markup Language (SAML)*, <http://www.oasis-open.org/committees/security/docs/draft-sstc-glossary-01.pdf>

2450	<a href="http://www.oasis-open.org/committees/security/docs/draft-sstc-glossary-02.pdf">open.org/committees/security/docs/draft-sstc-glossary-02.pdf</a> , OASIS, December 2001.
2451	
2452	<b>[SAML-P-XSD]</b> P. Hallam-Baker et al., <i>SAML protocol schema</i> , <a href="http://www.oasis-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</a> , OASIS, December 2001.
2453	
2454	
2455	<b>[SAML-Secure]</b> TBS
2456	<b>[SAML-XSD]</b> P. Hallam-Baker et al., <i>SAML assertion schema</i> , <a href="http://www.oasis-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</a> , OASIS, December 2001.
2457	
2458	
2459	<b>[Schema1]</b> H. S. Thompson et al., <i>XML Schema Part 1: Structures</i> , <a href="http://www.w3.org/TR/xmlschema-1/">http://www.w3.org/TR/xmlschema-1/</a> , World Wide Web Consortium Recommendation, May 2001.
2460	
2461	
2462	<b>[Schema2]</b> P. V. Biron et al., <i>XML Schema Part 2: Datatypes</i> , <a href="http://www.w3.org/TR/xmlschema-2/">http://www.w3.org/TR/xmlschema-2/</a> , World Wide Web Consortium Recommendation, May 2001.
2463	
2464	
2465	<b>[XMLEnc]</b> <i>XML Encryption Specification</i> , In development.
2466	<b>[XMLSig]</b> D. Eastlake et al., <i>XML-Signature Syntax and Processing</i> , <a href="http://www.w3.org/TR/xmldsig-core/">http://www.w3.org/TR/xmldsig-core/</a> , World Wide Web Consortium.
2467	
2468	<b>[XMLSig-XSD]</b> XML Signature Schema available from <a href="http://www.w3.org/TR/2000/CR-xmldsig-core-20001031/xmldsig-core-schema.xsd">http://www.w3.org/TR/2000/CR-xmldsig-core-20001031/xmldsig-core-schema.xsd</a> .
2469	
2470	<b>[XTAML]</b> P. Hallam-Baker, <i>XML Trust Axiom Markup Language 1.0</i> , <a href="http://www.xmltrustcenter.org/">http://www.xmltrustcenter.org/</a> , VeriSign Inc. September 2001.
2471	
2472	<b>[W3C-CHAR]</b> <a href="http://www.w3.org/TR/WD-charreq">http://www.w3.org/TR/WD-charreq</a>
2473	<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>
2474	<b>[W3C-CharMod]</b> <a href="http://www.w3.org/TR/charmod/">http://www.w3.org/TR/charmod/</a>
2475	<b>[XML]</b> <a href="http://www.w3.org/TR/REC-xml">http://www.w3.org/TR/REC-xml</a>

2476 **10. [RFC 2396]**

2477 **[http://www.ietf.org/rfc/rfc2396.txt?](http://www.ietf.org/rfc/rfc2396.txt)**

2478 **Acknowledgements**

2479 The editors would like to acknowledge the contributions of the OASIS SAML Technical Committee,  
2480 whose voting members at the time of publication were:

2481 [sort on last names; list to be supplied by Steve Anderson]

2482 Paul Apple, Foo Co.

2483 Ann Bingham, Bar Inc.

2484 Evan Cinch, Baz Company

2485 The editors would also like to thank the following people for their contributions:

2486 Mary Hadalittlelamb, former editor

2487 Peter Pan, who wrote the first draft of the section on XYZ

2488 John Doe, former chair of the Foo subcommittee

2489 **Contributors [who should appeareth in one list or t' other]:**

2490 Carlisle Adams, Entrust

2491 Scott Cantor, The Ohio State University

2492 Marc Chanliau, Netegrity

2493 Nigel Edwards, Hewlett-Packard

2494 Marlena Erdos, Tivoli

2495 Stephen Farrell, Baltimore Technologies

2496 Simon Godik, Crosslogic

2497 Jeff Hodges, Oblix

2498 Charles Knouse, Oblix

2499 Hal Lockhart, Entegrity Solutions

2500 Chris McLaren, Netegrity

2501 Prateek Mishra, Netegrity

2502 RL "Bob" Morgan, University of Washington

2503 Tim Moses, Entrust

2504 David Orchard, BEA

2505 Joe Pato, Hewlett Packard

2506 Darren Platt, RSA Security

2507 Irving Reid, Baltimore Technologies

2508 Krishna Sankar, Cisco Systems Inc

2509

## Appendix A. Notices

2510 OASIS takes no position regarding the validity or scope of any intellectual property or other rights  
2511 that might be claimed to pertain to the implementation or use of the technology described in this  
2512 document or the extent to which any license under such rights might or might not be available;  
2513 neither does it represent that it has made any effort to identify any such rights. Information on  
2514 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS  
2515 website. Copies of claims of rights made available for publication and any assurances of licenses to  
2516 be made available, or the result of an attempt made to obtain a general license or permission for  
2517 the use of such proprietary rights by implementors or users of this specification, can be obtained  
2518 from the OASIS Executive Director.

2519 OASIS invites any interested party to bring to its attention any copyrights, patents or patent  
2520 applications, or other proprietary rights which may cover technology that may be required to  
2521 implement this specification. Please address the information to the OASIS Executive Director.

2522 Copyright © The Organization for the Advancement of Structured Information Standards [OASIS]  
2523 2001. All Rights Reserved.

2524 This document and translations of it may be copied and furnished to others, and derivative works  
2525 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,  
2526 published and distributed, in whole or in part, without restriction of any kind, provided that the above  
2527 copyright notice and this paragraph are included on all such copies and derivative works. However,  
2528 this document itself may not be modified in any way, such as by removing the copyright notice or  
2529 references to OASIS, except as needed for the purpose of developing OASIS specifications, in  
2530 which case the procedures for copyrights defined in the OASIS Intellectual Property Rights  
2531 document must be followed, or as required to translate it into languages other than English.

2532 The limited permissions granted above are perpetual and will not be revoked by OASIS or its  
2533 successors or assigns.

2534 This document and the information contained herein is provided on an "AS IS" basis and OASIS  
2535 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO  
2536 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY  
2537 RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A  
2538 PARTICULAR PURPOSE.