



1

2

3

4

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

5

Document identifier: draft-sstc-core-30

6

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

7

Publication date: ~~April 3rd 2002~~ ~~March 29th 2002~~

8

Maturity Level: Committee Working Draft

9

Send comments to: security-requestors-comment@lists.oasis-open.org

10

Note: Before sending a message to this list you must first subscribe; send an email message to security-

11

requestors-comment-request@lists.oasis-open.org with the word "subscribe" as the body of the message.

12

Editors:

13

Phillip Hallam-Baker, VeriSign,

14

Eve Maler, Sun Microsystems

15

15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42

**ASSERTIONS AND PROTOCOL FOR THE OASIS SECURITY ASSERTION MARKUP
LANGUAGE (SAML) 1**

1. INTRODUCTION 10

1.1. NOTATION 10

1.2. SCHEMA ORGANIZATION AND NAMESPACES 10

1.2.1. String and URI Values 11

1.2.2. Time Values. 11

1.2.3. Comparing SAML values 11

1.3. SAML CONCEPTS (NON-NORMATIVE) 11

1.3.1. Overview 12

1.3.2. SAML and URI-Based Identifiers 13

1.3.3. SAML and Extensibility 13

2. SAML ASSERTIONS 14

2.1. SCHEMA HEADER AND NAMESPACE DECLARATIONS 14

2.2. SIMPLE TYPES 14

2.2.1. Simple Types IDType and IDReferenceType 14

2.2.2. Simple Type DecisionType 15

2.3. ASSERTIONS 15

2.3.1. Element <AssertionID> 15

2.3.2. Element <Assertion> 15

2.3.2.1. Element <Conditions> 17

2.3.2.1.1 Attributes NotBefore and NotOnOrAfter 18

2.3.2.1.2 Element <Condition> 18

2.3.2.1.3 Elements <AudienceRestrictionCondition> and <Audience> 18

2.3.2.2. Elements <Advice> and <AdviceElement> 19

2.4. STATEMENTS 19

2.4.1. Element <Statement> 19

43	<u>2.4.2. Element <SubjectStatement></u>	<u>19</u>
44	<u>2.4.2.1. Element <Subject></u>	<u>20</u>
45	<u>2.4.2.2. Element <NameIdentifier></u>	<u>20</u>
46	<u>2.4.2.3. Elements <SubjectConfirmation>, <ConfirmationMethod>, and <SubjectConfirmationData></u>	<u>21</u>
47	<u>2.4.3. Element <AuthenticationStatement></u>	<u>22</u>
48	<u>2.4.3.1. Element <SubjectLocality></u>	<u>22</u>
49	<u>2.4.3.2. Element <AuthorityBinding></u>	<u>23</u>
50	<u>2.4.4. Element <AuthorizationDecisionStatement></u>	<u>23</u>
51	<u>2.4.4.1. Element <Action></u>	<u>25</u>
52	<u>2.4.4.2. Element <Evidence></u>	<u>25</u>
53	<u>2.4.5. Element <AttributeStatement></u>	<u>25</u>
54	<u>2.4.5.1. Elements <AttributeDesignator> and <Attribute></u>	<u>26</u>
55	<u>2.4.5.1.1 Element <AttributeValue></u>	<u>26</u>
56	<u>3. SAML PROTOCOL</u>	<u>28</u>
57	<u>3.1. SCHEMA HEADER AND NAMESPACE DECLARATIONS</u>	<u>28</u>
58	<u>3.2. REQUESTS</u>	<u>28</u>
59	<u>3.2.1. Complex Type RequestAbstractType</u>	<u>28</u>
60	<u>3.2.1.1. Element <RespondWith></u>	<u>29</u>
61	<u>3.2.2. Element <Request></u>	<u>29</u>
62	<u>3.2.3. Element <AssertionArtifact></u>	<u>30</u>
63	<u>3.3. QUERIES</u>	<u>30</u>
64	<u>3.3.1. Element <Query></u>	<u>30</u>
65	<u>3.3.2. Element <SubjectQuery></u>	<u>31</u>
66	<u>3.3.3. Element <AuthenticationQuery></u>	<u>31</u>
67	<u>3.3.4. Element <AttributeQuery></u>	<u>32</u>
68	<u>3.3.5. Element <AuthorizationDecisionQuery></u>	<u>32</u>
69	<u>3.4. RESPONSES</u>	<u>33</u>
70	<u>3.4.1. Complex Type ResponseAbstractType</u>	<u>33</u>
71	<u>3.4.2. Element <Response></u>	<u>34</u>
72	<u>3.4.3. Element <Status></u>	<u>34</u>

73	3.4.3.1. Element <StatusCode>	35
74	3.4.3.2. Element <StatusMessage>	36
75	3.4.3.3. Element <StatusDetail>	36
76	3.4.4. Responses to <AuthenticationQuery> and <AttributeQuery>	36
77	4. SAML VERSIONING	38
78	4.1. ASSERTION VERSION	38
79	4.2. REQUEST VERSION	38
80	4.3. RESPONSE VERSION	39
81	5. SAML & XML-SIGNATURE SYNTAX AND PROCESSING	40
82	5.1. SIGNING ASSERTIONS	40
83	5.2. REQUEST /RESPONSE SIGNING	41
84	5.3. SIGNATURE INHERITANCE	41
85	5.3.1. <i>Rationale</i>	41
86	5.3.2. <i>Rules for SAML Signature Inheritance</i>	41
87	5.4. XML SIGNATURE PROFILE	41
88	5.4.1. <i>Signing formats</i>	41
89	5.4.2. <i>CanonicalizationMethod</i>	41
90	5.4.3. <i>Transforms</i>	42
91	5.4.4. <i>KeyInfo</i>	42
92	5.4.5. <i>Binding between statements in a multi-statement assertion</i>	42
93	6. SAML EXTENSIONS	43
94	6.1. ASSERTION SCHEMA EXTENSION	43
95	6.2. PROTOCOL SCHEMA EXTENSION	43
96	6.3. USE OF TYPE DERIVATION AND SUBSTITUTION GROUPS	44
97	7. SAML-DEFINED IDENTIFIERS	45

98	7.1. AUTHENTICATION METHOD IDENTIFIERS	45
99	7.1.1. Password (Pass-Through):	46
100	7.1.2. Kerberos	46
101	7.1.3. SSL/TLS Certificate Based Client Authentication:	47
102	7.1.4. X.509 Public Key	47
103	7.1.5. PGP Public Key	47
104	7.1.6. SPKI Public Key	47
105	7.1.7. XKMS Public Key	47
106	7.1.8. XML Digital Signature	48
107	7.2. ACTION NAMESPACE IDENTIFIERS	48
108	7.2.1. Read/Write/Execute/Delete/Control:	48
109	7.2.2. Read/Write/Execute/Delete/Control with Negation:	49
110	7.2.3. Get/Head/Put/Post:	49
111	7.2.4. UNIX File Permissions:	49
112	8. SAML SCHEMA LISTINGS	51
113	8.1. ASSERTION SCHEMA	51
114	8.2. PROTOCOL SCHEMA	54
115	9. REFERENCES	57
116	10. ACKNOWLEDGEMENTS	59
117	APPENDIX A. NOTICES	61
118	ASSERTIONS AND PROTOCOL FOR THE OASIS SECURITY ASSERTION MARKUP	
119	LANGUAGE (SAML)	1
120	1. INTRODUCTION	9
121	1.1. NOTATION	9
122	1.2. SCHEMA ORGANIZATION AND NAMESPACES	9

123	<i>1.2.1. String and URI Values</i>	10
124	<i>1.2.2. Time Values.</i>	10
125	<i>1.2.3. Comparing SAML values</i>	10
126	1.3. SAML CONCEPTS (NON-NORMATIVE)	10
127	<i>1.3.1. Overview</i>	11
128	<i>1.3.2. SAML and URI Based Identifiers</i>	12
129	<i>1.3.3. SAML and Extensibility</i>	12
130	2. SAML ASSERTIONS	13
131	2.1. SCHEMA HEADER AND NAMESPACE DECLARATIONS	13
132	2.2. SIMPLE TYPES	13
133	<i>2.2.1. Simple Types IDType and IDReferenceType</i>	13
134	<i>2.2.2. Simple Type DecisionType</i>	14
135	2.3. ASSERTIONS	14
136	<i>2.3.1. Element <AssertionID></i>	14
137	<i>2.3.2. Element <Assertion></i>	14
138	<i>2.3.2.1. Element <Conditions></i>	16
139	<i>2.3.2.1.1. Attributes NotBefore and NotOnOrAfter</i>	17
140	<i>2.3.2.1.2. Element <Condition></i>	17
141	<i>2.3.2.1.3. Elements <AudienceRestrictionCondition> and <Audience></i>	17
142	<i>2.3.2.2. Elements <Advice> and <AdviceElement></i>	18
143	2.4. STATEMENTS	18
144	<i>2.4.1. Element <Statement></i>	18
145	<i>2.4.2. Element <SubjectStatement></i>	18
146	<i>2.4.2.1. Element <Subject></i>	19
147	<i>2.4.2.2. Element <NameIdentifier></i>	19
148	<i>2.4.2.3. Elements <SubjectConfirmation>, <ConfirmationMethod>, and <SubjectConfirmationData></i>	20
149	<i>2.4.3. Element <AuthenticationStatement></i>	21
150	<i>2.4.3.1. Element <SubjectLocality></i>	21
151	<i>2.4.3.2. Element <AuthorityBinding></i>	22
152	<i>2.4.4. Element <AuthorizationDecisionStatement></i>	22

153	2.4.4.1. Element <Action>	23
154	2.4.4.2. Element <Evidence>	24
155	2.4.5. Element <AttributeStatement>	24
156	2.4.5.1. Elements <AttributeDesignator> and <Attribute>	25
157	2.4.5.1.1. Element <AttributeValue>	25
158	3. SAML PROTOCOL	26
159	3.1. SCHEMA HEADER AND NAMESPACE DECLARATIONS	26
160	3.2. REQUESTS	26
161	3.2.1. Complex Type RequestAbstractType	26
162	3.2.1.1. Element <RespondWith>	27
163	3.2.2. Element <Request>	27
164	3.2.3. Element <AssertionArtifact>	28
165	3.3. QUERIES	28
166	3.3.1. Element <Query>	28
167	3.3.2. Element <SubjectQuery>	29
168	3.3.3. Element <AuthenticationQuery>	29
169	3.3.4. Element <AttributeQuery>	30
170	3.3.5. Element <AuthorizationDecisionQuery>	30
171	3.4. RESPONSES	31
172	3.4.1. Complex Type ResponseAbstractType	31
173	3.4.2. Element <Response>	32
174	3.4.3. Element <Status>	32
175	3.4.3.1. Element <StatusCode>	33
176	3.4.3.2. Element <StatusMessage>	34
177	3.4.3.3. Element <StatusDetail>	34
178	3.4.4. Responses to <AuthenticationQuery> and <AttributeQuery>	34
179	4. SAML VERSIONING	35
180	4.1. ASSERTION VERSION	35

181	4.2. REQUEST VERSION	35
182	4.3. RESPONSE VERSION	36
183	5. SAML & XML SIGNATURE SYNTAX AND PROCESSING	37
184	5.1. SIGNING ASSERTIONS	37
185	5.2. REQUEST /RESPONSE SIGNING	38
186	5.3. SIGNATURE INHERITANCE	38
187	5.3.1. Rationale	38
188	5.3.2. Rules for SAML Signature Inheritance	38
189	5.4. XML SIGNATURE PROFILE	38
190	5.4.1. Signing formats	38
191	5.4.2. CanonicalizationMethod	38
192	5.4.3. Transforms	39
193	5.4.4. KeyInfo	39
194	5.4.5. Binding between statements in a multi-statement assertion	39
195	6. SAML EXTENSIONS	40
196	6.1. ASSERTION SCHEMA EXTENSION	40
197	6.2. PROTOCOL SCHEMA EXTENSION	40
198	6.3. USE OF TYPE DERIVATION AND SUBSTITUTION GROUPS	41
199	7. SAML-DEFINED IDENTIFIERS	42
200	7.1. AUTHENTICATION METHOD AND CONFIRMATION METHOD IDENTIFIERS	42
201	7.1.1. SAML Artifact (SHA-1):	42
202	7.1.2. Holder of Key:	43
203	7.1.3. Bearer Indication:	43
204	7.1.4. Sender Vouches:	43
205	7.1.5. Password (Pass-Through):	43

206	7.1.6. Password (One-Way Function SHA-1):	43
207	7.1.7. Kerberos	43
208	7.1.8. SSL/TLS Certificate Based Client Authentication:	43
209	7.1.9. Object Authenticator (SHA-1):	44
210	7.1.10. PKCS#7	44
211	7.1.11. Cryptographic Message Syntax	44
212	7.1.12. XML Digital Signature	44
213	7.2. ACTION NAMESPACE IDENTIFIERS	44
214	7.2.1. Read/Write/Execute/Delete/Control:	44
215	7.2.2. Read/Write/Execute/Delete/Control with Negation:	45
216	7.2.3. Get/Head/Put/Post:	45
217	7.2.4. UNIX File Permissions:	45
218	8. SAML SCHEMA LISTINGS	47
219	8.1. ASSERTION SCHEMA	47
220	8.2. PROTOCOL SCHEMA	50
221	9. REFERENCES	53
222	10. ACKNOWLEDGEMENTS	55
223	APPENDIX A. NOTICES	57
224		

225

1. Introduction

226
227
228
229
230
231

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 [SAMLBind] provides frameworks for this embedding and transport. Files containing just the SAML assertion schema [SAML-XSD] and protocol schema [SAMPL-XSD] are available.

232

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

233

1.1. Notation

234
235
236

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

237
238
239

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 [RFC 2119]:

240
241

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

242
243
244
245

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.

246
247

Listings of SAML schemas appear like this.

248

Example code listings appear like this.

249
250
251

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:

252

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

253

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

254

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

255
256

?? 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.

257
258

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

259

1.2. Schema Organization and Namespaces

260
261

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

262

`urn:oasis:names:tc:SAML:1.0:assertion`

263
264

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

265

`urn:oasis:names:tc:SAML:1.0:protocol`

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

268 The assertion schema is imported into the protocol schema. Also imported into both schemas is the
269 schema for XML Signature [**XMLSig-XSD**], which is associated with the following XML namespace:

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

271 **1.2.1. String and URI Values**

272 All SAML string and URI values have the types string and anyURI respectively, which are built in to
273 the W3C XML Schema Datatypes specification. All strings in SAML messages **MUST** consist of at
274 least one non-whitespace character (whitespace is defined in [XML 1.0 Sec. 2.3]). Empty and
275 whitespace-only values are disallowed. Also, unless otherwise indicated in this specification, all URI
276 values **MUST** consist of at least one non-whitespace character.

277 **1.2.2. Time Values.**

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

280 SAML Requestors and Responders **SHOULD NOT** rely on other applications supporting time
281 resolution finer than milliseconds. Implementations **MUST NOT** generate time instants that specify
282 leap seconds.

283 **1.2.3. Comparing SAML values**

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

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

296 Applications that compare data received in SAML documents to data from external sources **MUST**
297 take into account the normalization rules specified for XML. Text contained within elements is
298 normalized so that line endings are represented using linefeed characters (ASCII code 10_{Decimal}), as
299 described in section 2.11 of the XML Recommendation [**XML**]. Attribute values defined as strings
300 (or types derived from strings) are normalized as described in section 3.3.3 [**XML**]. All white space
301 characters are replaced with blanks (ASCII code 32_{Decimal}).

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

305 **1.3. SAML Concepts (Non-Normative)**

306 This section is informative only and is superseded by any contradicting information in the normative
307 text in Section 2 and following. A glossary of SAML terms and concepts [**SAMLGloss**] is available.

308

1.3.1. Overview

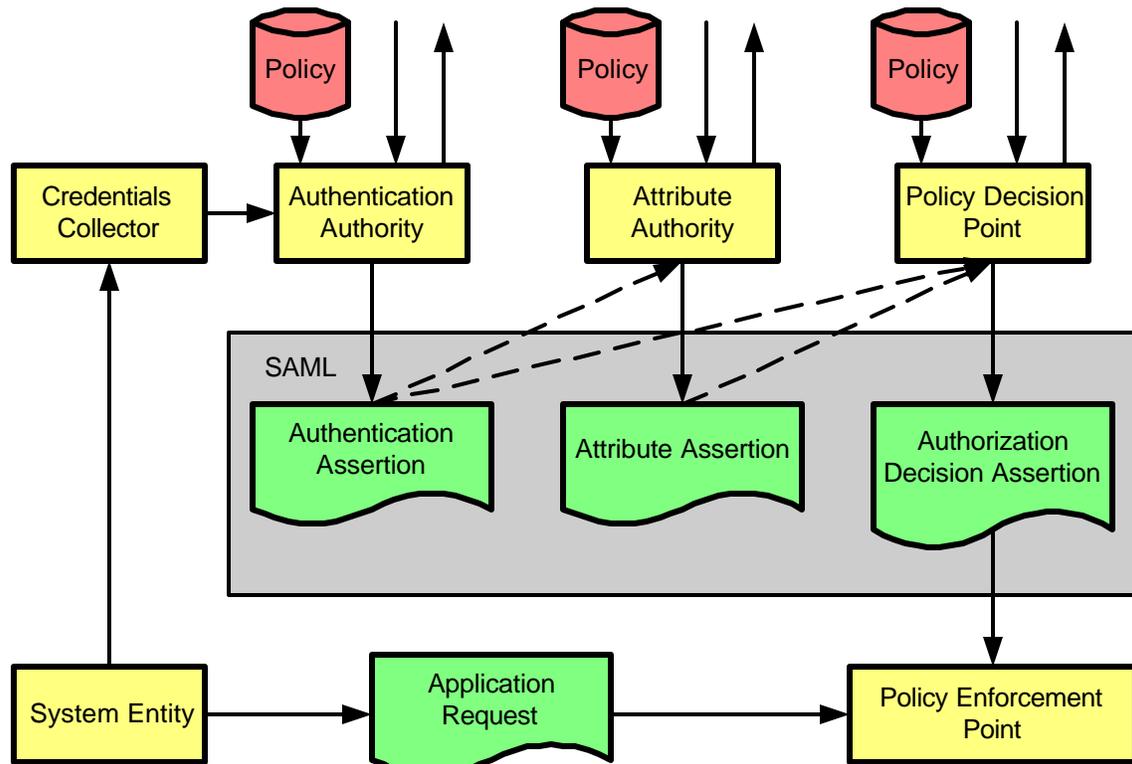
309 The Security Assertion Markup Language (SAML) is an XML-based framework for exchanging
310 security information. This security information is expressed in the form of assertions about subjects,
311 where a subject is an entity (either human or computer) that has an identity in some security
312 domain.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

314 Assertions can convey information about authentication acts performed by subjects, attributes of
315 subjects, and authorization decisions about whether subjects are allowed to access certain
316 resources. Assertions are represented as XML constructs and have a nested structure, whereby a
317 single assertion might contain several different internal statements about authentication,
318 authorization, and attributes. Note that assertions containing authentication statements merely
319 describe acts of authentication that happened previously.

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

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

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



330

331

Figure 1 The SAML Domain Model

332 One major design goal for SAML is Single Sign-On (SSO), the ability of a user to authenticate in
333 one domain and use resources in other domains without re-authenticating. However, SAML can be

334 used in various configurations to support additional scenarios as well. Several profiles of SAML are
335 currently being defined that support different styles of SSO and the securing of SOAP payloads.

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

340 **1.3.2. SAML and URI-Based Identifiers**

341 SAML defines some identifiers to manage references to well-known concepts and sets of values.
342 For example, the SAML-defined identifier for the [Kerberos-subject-confirmationpassword](#)
343 [authentication](#) method is as follows:

344 **[urn:oasis:names:tc:SAML:1.0:am:password](#)**
345 **[urn:iETF:rfc:1510](#)**

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

348 **[urn:oasis:names:tc:SAML:1.0:action:rwedc](#)**

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

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

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

360 **1.3.3. SAML and Extensibility**

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

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

365 2. SAML Assertions

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

368 ?? **Authentication:** The specified subject was authenticated by a particular means at a
369 particular time.

370 ?? **Authorization Decision:** A request to allow the specified subject to access the specified
371 resource has been granted or denied.

372 ?? **Attribute:** The specified subject is associated with the supplied attributes.

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

376 2.1. Schema Header and Namespace Declarations

377 The following schema fragment defines the XML namespaces and other header information for the
378 assertion schema:

```
379 <schema  
380   targetNamespace="urn:oasis:names:tc:SAML:1.0:assertion"  
381   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
382   xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"  
383   xmlns="http://www.w3.org/2001/XMLSchema"  
384   elementFormDefault="unqualified">  
385   <import namespace="http://www.w3.org/2000/09/xmldsig#"  
386     schemaLocation="xmldsig-core-schema.xsd"/>  
387   <annotation>  
388     <documentation>draft-sstc-schema-assertion-30.xsd</documentation>  
389   </annotation>  
390   ...  
391 </schema>
```

392 2.2. Simple Types

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

394 2.2.1. Simple Types IDType and IDReferenceType

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

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

398 ?? Any party that assigns an identifier MUST ensure that there is negligible probability that that
399 party or any other party will accidentally assign the same identifier to a different data object.

400 ?? Where a data object declares that it has a particular identifier, there MUST be exactly one
401 such declaration.

402 The mechanism by which the SAML Requestor or Responder ensures that the identifier is unique is
403 left to the implementation. In the case that a pseudorandom technique is employed, the probability
404 of two randomly chosen identifiers being identical MUST be less than 2^{-128} and SHOULD be less
405 than 2^{-160} . This requirement MAY be met by applying Base64 encoding to a randomly chosen value
406 128 or 160 bits in length.

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

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

```
411 <simpleType name="IDType">  
412   <restriction base="string"/>  
413 </simpleType>  
414 <simpleType name="IDReferenceType">  
415   <restriction base="string"/>  
416 </simpleType>
```

417 2.2.2. Simple Type DecisionType

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

420 Permit

421 The specified action is permitted.

422 Deny

423 The specified action is denied.

424 IndeterminateThe issuer cannot determine whether the specified action is permitted or denied.

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

429

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

```
431 <simpleType name="DecisionType">  
432   <restriction base="string">  
433     <enumeration value="Permit"/>  
434     <enumeration value="Deny"/>  
435     <enumeration value="Indeterminate"/>  
436   </restriction>  
437 </simpleType>
```

438 2.3. Assertions

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

440 2.3.1. Element <AssertionID>

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

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

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

445 2.3.2. Element <Assertion>

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

448 MajorVersion [Required]

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

451 **MinorVersion** [Required]
 452 The minor version of this assertion. The identifier for the version of SAML defined in this
 453 specification is 0. Processing of this attribute is specified in Section 3.4.4.

454 **AssertionID** [Required]
 455 The identifier for this assertion. It is of type **IDType**, and MUST follow the requirements
 456 specified by that type for identifier uniqueness.

457 **Issuer** [Required]
 458 The issuer of the assertion. The name of the issuer is provided as a string. The issuer
 459 name SHOULD be unambiguous to the intended relying parties. SAML authorities may use
 460 an identifier such as a URI reference that is designed to be unambiguous regardless of
 461 context.

462 **IssueInstant** [Required]
 463 The time instant of issue in UTC as described in section 1.2.1.

464 **<Conditions>** [Optional]
 465 Conditions that MUST be taken into account in assessing the validity of the assertion.

466 **<Advice>** [Optional]
 467 Additional information related to the assertion that assists processing in certain situations
 468 but which MAY be ignored by applications that do not support its use.

469 **<Signature>** [Optional]
 470 An XML Signature that authenticates the assertion, see section 5.

471 One or more of the following statement elements:

472 **<Statement>**
 473 A statement defined in an extension schema.

474 **<SubjectStatement>**
 475 A subject statement defined in an extension schema.

476 **<AuthenticationStatement>**
 477 An authentication statement.

478 **<AuthorizationDecisionStatement>**
 479 An authorization decision statement.

480 **<AttributeStatement>**
 481 An attribute statement.

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

```

484     <element name="Assertion" type="saml:AssertionType"/>
485     <complexType name="AssertionType">
486         <sequence>
487             <element ref="saml:Conditions" minOccurs="0"/>
488             <element ref="saml:Advice" minOccurs="0"/>
489             <choice maxOccurs="unbounded">
490                 <element ref="saml:Statement"/>
491                 <element ref="saml:SubjectStatement"/>
492                 <element ref="saml:AuthenticationStatement"/>
493                 <element ref="saml:AuthorizationDecisionStatement"/>
494                 <element ref="saml:AttributeStatement"/>
495             </choice>
496             <element ref="ds:Signature" minOccurs="0"/>
497         </sequence>
498         <attribute name="MajorVersion" type="integer" use="required"/>
499         <attribute name="MinorVersion" type="integer" use="required"/>
500         <attribute name="AssertionID" type="saml:IDType" use="required"/>
  
```

```
501     <attribute name="Issuer" type="string" use="required"/>
502     <attribute name="IssueInstant" type="dateTime" use="required"/>
503 </complexType>
```

504 2.3.2.1. Element <Conditions>

505 ~~If an assertion contains a <Conditions> element, the validity of the assertion is dependent on the~~
506 ~~conditions provided. Each condition evaluates to a status of Valid, Invalid, or Indeterminate.~~

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

508 NotBefore [Optional]

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

511 NotOnOrAfter [Optional]

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

514 <Condition> [Any Number]

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

516 <AudienceRestrictionCondition> [Any Number]

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

518 The following schema fragment defines the <Conditions> element and its **ConditionsType**
519 complex type:

```
520     <element name="Conditions" type="saml:ConditionsType"/>
521     <complexType name="ConditionsType">
522         <choice minOccurs="0" maxOccurs="unbounded">
523             <element ref="saml:AudienceRestrictionCondition"/>
524             <element ref="saml:Condition"/>
525         </choice>
526         <attribute name="NotBefore" type="dateTime" use="optional"/>
527         <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
528     </complexType>
```

529 If an assertion contains a <Conditions> element, the validity of the assertion is dependent on the
530 sub-elements and attributes provided. When processing the sub-elements and attributes of a
531 <Conditions> element, the following rules MUST be used in the order shown to determine the
532 overall validity of the assertion:

- 533 1. If no sub-elements or attributes are supplied in the <Conditions> element, then the
534 assertion is considered to be **Valid**.
- 535 2. If any sub-element or attribute of the <Conditions> element is determined to be invalid,
536 then the assertion is **Invalid**.
- 537 3. If any sub-element or attribute of the <Conditions> element cannot be evaluated, then
538 the validity of the assertion cannot be determined and is deemed to be **Indeterminate**.
- 539 4. If all sub-elements and attributes of the <Conditions> element are determined to be
540 **Valid**, then the assertion is considered to be **Valid**.

541 The <Conditions> element MAY be extended to contain additional conditions. If an element
542 contained within a <Conditions> element is encountered that is not understood, the status of the
543 condition cannot be evaluated and the validity status of the assertion MUST be deemed to be
544 **Indeterminate** in accordance with rule 3 above.

545 Note that an assertion that has validity status **Valid** may not be trustworthy by reasons such as not
546 being issued by a trustworthy issuer or not being authenticated by a trustworthy means.

547 **2.3.2.1.1 Attributes *NotBefore* and *NotOnOrAfter***

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

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

551 If the value for either `NotBefore` or `NotOnOrAfter` is omitted it is considered unspecified. If the
552 `NotBefore` attribute is unspecified (and if any other conditions that are supplied evaluate to
553 `Valid`), the assertion is valid at any time before the time instant specified by the `NotOnOrAfter`
554 attribute. If the `NotOnOrAfter` attribute is unspecified (and if any other conditions that are supplied
555 evaluate to `Valid`), the assertion is valid from the time instant specified by the `NotBefore`
556 attribute with no expiry. If neither attribute is specified (and if any other conditions that are supplied
557 evaluate to `Valid`), the assertion is valid at any time.

558 The `NotBefore` and `NotOnOrAfter` attributes are defined to have the **dateTime** simple type that
559 is built in to the W3C XML Schema Datatypes specification [**Schema2**]. All time instants are
560 specified in Universal Coordinated Time (UTC) as described in section 1.2.1. Implementations
561 MUST NOT generate time instants that specify leap seconds.

562 **2.3.2.1.2 Element `<Condition>`**

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

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

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

570 **2.3.2.1.3 Elements `<AudienceRestrictionCondition>` and `<Audience>`**

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

576 `<Audience>`

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

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

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

587 The following schema fragment defines the `<AudienceRestrictionCondition>` element and
588 its **AudienceRestrictionConditionType** complex type:

```
589 <element name="AudienceRestrictionCondition"  
590 type="saml:AudienceRestrictionConditionType"/>  
591 <complexType name="AudienceRestrictionConditionType">  
592 <complexContent>  
593 <extension base="saml:ConditionAbstractType">
```

```

594         <sequence>
595             <element ref="saml:Audience" maxOccurs="unbounded" />
596         </sequence>
597     </extension>
598 </complexContent>
599 </complexType>
600 <element name="Audience" type="anyURI" />

```

601 2.3.2.2. Elements <Advice> and <AdviceElement>

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

605 The <Advice> element contains a mixture of zero or more <Assertion> elements,
606 <AssertionIDReference> elements and elements in other namespaces, with lax schema
607 validation in effect for these other elements.

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

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

613 The following schema fragment defines the <Advice> element and its **AdviceType** complex type:

```

614 <element name="Advice" type="saml:AdviceType" />
615 <complexType name="AdviceType">
616     <choice minOccurs="0" maxOccurs="unbounded">
617         <element ref="saml:AssertionIDReference" />
618         <element ref="saml:Assertion" />
619         <any namespace="##other" processContents="lax" />
620     </choice>
621 </complexType>

```

622 2.4. Statements

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

624 2.4.1. Element <Statement>

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

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

```

630 <element name="Statement" type="saml:StatementAbstractType" />
631 <complexType name="StatementAbstractType" abstract="true" />

```

632 2.4.2. Element <SubjectStatement>

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

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

```
640 <element name="SubjectStatement" type="saml:SubjectStatementAbstractType"/>  
641 <complexType name="SubjectStatementAbstractType" abstract="true">  
642 <complexContent>  
643 <extension base="saml:StatementAbstractType">  
644 <sequence>  
645 <element ref="saml:Subject"/>  
646 </sequence>  
647 </extension>  
648 </complexContent>  
649 </complexType>
```

650 2.4.2.1. Element <Subject>

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

653 <NameIdentifier>

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

655 <SubjectConfirmation>

656 Information that allows the subject to be authenticated.

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

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

```
664 <element name="Subject" type="saml:SubjectType"/>  
665 <complexType name="SubjectType">  
666 <choice>  
667 <sequence>  
668 <element ref="saml:NameIdentifier"/>  
669 <element ref="saml:SubjectConfirmation" minOccurs="0"/>  
670 </sequence>  
671 <element ref="saml:SubjectConfirmation"/>  
672 </choice>  
673 </complexType>
```

674 2.4.2.2. Element <NameIdentifier>

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

677 NameQualifier [Optional]

678 The security or administrative domain that qualifies the name of the subject.

679 The NameQualifier attribute provides a means to federate names from disparate user
680 stores without collision.

681 Format [Optional]

682 The syntax used to describe the name of the subject

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

686 #emailAddress

687 Indicates that the content of the NameIdentifier element is in the form of an email address,

688 specifically "addr-spec" as defined in section 3.4.1 of RFC 2822 [RFC 2822]. An addr-spec
689 has the form local-part@domain. Note that an addr-spec has no phrase (such as a
690 common name) before it, has no comment (text surrounded in parentheses) after it, and is
691 not surrounded by "<" and ">".

692 #X509SubjectName

693 Indicates that the content of the NameIdentifier element is in the form specified for
694 the contents of <ds:X509SubjectName> element in [DSIG]. Implementors should note that
695 [DSIG] specifies encoding rules for X.509 subject names that differ from the rules given in
696 RFC2253 [RFC2253].

697 #WindowsDomainQualifiedName

698 Indicates that the content of the NameIdentifier element is a Windows domain qualified
699 name. A Windows domain qualified user name is a string of the form
700 "DomainName\UserName". The domain name and "\" separator may be omitted.

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

```
703 <element name="NameIdentifier" type="saml:NameIdentifierType"/>  
704 <complexType name="NameIdentifierType">  
705 <simpleContent>  
706 <extension base="string">  
707 <attribute name="NameQualifier" type="string" use="optional"/>  
708 <attribute name="Format" type="anyURI" use="optional"/>  
709 </extension>  
710 </simpleContent>  
711 </complexType>
```

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

714 Regardless of format, issues of anonymity, pseudonymity, and the persistence of
715 the identifier with respect to the asserting and relying parties, are also
716 implementation-specific.

717 **2.4.2.3. Elements <SubjectConfirmation>, <ConfirmationMethod>, and** 718 **<SubjectConfirmationData>**

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

721 <ConfirmationMethod> [One or more]

722 A URI reference that identifies a protocol to be used to authenticate the subject. URI
723 references identifying ~~common authentication protocols are listed in Section 7. SAML-~~
724 ~~defined confirmation methods are currently defined with the SAML profiles in [SAMLBind].~~
725 ~~Additional SAML confirmation methods may be defined in future OASIS-approved SAML~~
726 ~~profile specifications.~~

727 <SubjectConfirmationData> [Optional]

728 Additional authentication information to be used by a specific authentication protocol.

729 <ds:KeyInfo> [Optional]

730 An XML Signature [XMLSig] element that specifies a cryptographic key held by the
731 subject.

732 The following schema fragment defines the <SubjectConfirmation> element and its
733 **SubjectConfirmationType** complex type, along with the <SubjectConfirmationData>
734 element and the <ConfirmationMethod> element:

```
735 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>  
736 <complexType name="SubjectConfirmationType">  
737 <sequence>
```

```

738         <element ref="saml:ConfirmationMethod" maxOccurs="unbounded" />
739         <element ref="saml:SubjectConfirmationData" minOccurs="0" />
740         <element ref="ds:KeyInfo" minOccurs="0" />
741     </sequence>
742 </complexType>
743 <element name="SubjectConfirmationData" type="et:stringanyType" />
744 <element name="ConfirmationMethod" type="anyURI" />

```

745 2.4.3. Element <AuthenticationStatement>

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

750 AuthenticationMethod [Optional]

751 A URI reference that specifies the type of authentication that took place. URI references
752 identifying common authentication protocols are listed in Section 7.

753 AuthenticationInstant [Optional]

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

756 <SubjectLocality> [Optional]

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

759 <AuthorityBinding> [Any Number]

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

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

```

763 <element name="AuthenticationStatement"
764     type="saml:AuthenticationStatementType" />
765 <complexType name="AuthenticationStatementType">
766     <complexContent>
767         <extension base="saml:SubjectStatementAbstractType">
768             <sequence>
769                 <element ref="saml:SubjectLocality" minOccurs="0" />
770                 <element ref="saml:AuthorityBinding"
771                     minOccurs="0" maxOccurs="unbounded" />
772             </sequence>
773             <attribute name="AuthenticationMethod" type="anyURI" />
774             <attribute name="AuthenticationInstant" type="dateTime" />
775         </extension>
776     </complexContent>
777 </complexType>

```

778 2.4.3.1. Element <SubjectLocality>

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

781 IPAddress [Optional]

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

783 DNSAddress [Optional]

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

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

787 The following schema fragment defines the <SubjectLocality> element and its
788 **SubjectLocalityType** complex type:

```
789     <element name="SubjectLocality"  
790           type="saml: SubjectLocalityType" />  
791     <complexType name="SubjectLocalityType">  
792       <attribute name="IPAddress" type="string" use="optional" />  
793       <attribute name="DNSAddress" type="string" use="optional" />  
794     </complexType>
```

795 2.4.3.2. Element <AuthorityBinding>

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

801 *AuthorityKind* [Required]

802 The type of SAML Protocol queries to which the authority described by this element will
803 respond. The value is specified as an XML Schema QName. The acceptable values for
804 *AuthorityKind* are the namespace-qualified names of element types or elements
805 derived from the SAML Protocol Query element (see Section 3.3). For example, an
806 attribute authority would be identified by *AuthorityKind*="samlp:AttributeQuery".
807 For extension schemas, where the actual type of the *samlp:Query* would be identified by
808 an *xsi:type* attribute, the value of *AuthorityKind* MUST be the same as the value of
809 the *xsi:type* attribute for the corresponding query.

810 *Location* [Required]

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

814 *Binding* [Required]

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

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

```
819     <element name="AuthorityBinding" type="saml:AuthorityBindingType" />  
820     <complexType name="AuthorityBindingType">  
821       <attribute name="AuthorityKind" type="QName" use="required" />  
822       <attribute name="Location" type="anyURI" use="required" />  
823       <attribute name="Binding" type="anyURI" use="required" />  
824     </complexType>
```

825 2.4.4. Element <AuthorizationDecisionStatement>

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

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

834 *In general, the rules for equivalence and definition of a normal form, if any, are scheme*
835 *dependent. When a scheme uses elements of the common syntax, it will also use the common*
836 *syntax equivalence rules, namely that the scheme and hostname are case insensitive and a*

837 *URL with an explicit ":port", where the port is the default for the scheme, is equivalent to one*
838 *where the port is elided.*

839 To avoid ambiguity resulting from variations in URI encoding SAML requestors and responders
840 SHOULD employ the URI normalized form wherever possible as follows:

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

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

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

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

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

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

857 Resource [Required]

858 A URI reference identifying the resource to which access
859 authorization is sought. It is permitted for this attribute to have
860 the value of the empty URI reference (""), and the meaning is
861 defined to be "the start of the current document", as specified by
862 [RFC 2396]§ 4.2.

863 Decision [Required]

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

866 <Action> [One or more]

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

868 <Evidence> [Any Number]

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

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

```
872 <element name="AuthorizationDecisionStatement"  
873 type="saml:AuthorizationDecisionStatementType"/>  
874 <complexType name="AuthorizationDecisionStatementType">  
875 <complexContent>  
876 <extension base="saml:SubjectStatementAbstractType">  
877 <sequence>  
878 <element ref="saml:Action" maxOccurs="unbounded"/>  
879 <element ref="saml:Evidence" minOccurs="0"/>  
880 </sequence>  
881 <attribute name="Resource" type="anyURI" use="required"/>  
882 <attribute name="Decision" type="saml:DecisionType" use="required"/>  
883 </extension>  
884 </complexContent>  
885 </complexType>
```

886 2.4.4.1. Element <Action>

887 The <Action> element specifies an action on the specified resource for which permission is
888 sought. It has the following attribute:

889 Namespace [Optional]

890 A URI reference representing the namespace in which the name of the specified action is
891 to be interpreted. If this element is absent, the namespace
892 urn:oasis:names:tc:SAML:1.0:action:rwdc-negation specified in section 7.2.2 is in effect.

893 *string data* [Required]

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

895 The following schema fragment defines the <Action> element and its **ActionType** complex type:

```
896 <element name="Action" type="saml:ActionType"/>  
897 <complexType name="ActionType">  
898 <simpleContent>  
899 <extension base="string">  
900 <attribute name="Namespace" type="anyURI"/>  
901 </extension>  
902 </simpleContent>  
903 </complexType>
```

904 2.4.4.2. Element <Evidence>

905 The <Evidence> element contains an assertion that the issuer relied on in issuing the
906 authorization decision. It has the **EvidenceType** complex type. It contains one of the following
907 elements:

908 <AssertionIDReference>

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

910 <Assertion>

911 Specifies an assertion by value.

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

917 The following schema fragment defines the <Evidence> element and its **EvidenceType** complex
918 type:

```
919 <element name="Evidence" type="saml:EvidenceType"/>  
920 <complexType name="EvidenceType">  
921 <choice maxOccurs="unbounded">  
922 <element ref="saml:AssertionIDReference"/>  
923 <element ref="saml:Assertion"/>  
924 </choice>  
925 </complexType>
```

926 2.4.5. Element <AttributeStatement>

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

930 <Attribute> [One or More]

931 The <Attribute> element specifies an attribute of the subject.

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

```
934 <element name="AttributeStatement" type="saml:AttributeStatementType" />  
935 <complexType name="AttributeStatementType">  
936 <complexContent>  
937 <extension base="saml:SubjectStatementAbstractType">  
938 <sequence>  
939 <element ref="saml:Attribute" maxOccurs="unbounded" />  
940 </sequence>  
941 </extension>  
942 </complexContent>  
943 </complexType>
```

944 2.4.5.1. Elements <AttributeDesignator> and <Attribute>

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

949 AttributeNamespace [Optional]
950 The namespace in which the AttributeName elements are interpreted.

951 AttributeName [Optional]
952 The name of the attribute.

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

```
955 <element name="AttributeDesignator" type="saml:AttributeDesignatorType" />  
956 <complexType name="AttributeDesignatorType">  
957 <attribute name="AttributeName" type="string" use="required" />  
958 <attribute name="AttributeNamespace" type="anyURI" use="required" />  
959 </complexType>
```

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

963 <AttributeValue> [Any Number]
964 The value of the attribute.

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

```
967 <element name="Attribute" type="saml:AttributeType" />  
968 <complexType name="AttributeType">  
969 <complexContent>  
970 <extension base="saml:AttributeDesignatorType">  
971 <sequence>  
972 <element ref="saml:AttributeValue" maxOccurs="unbounded" />  
973 </sequence>  
974 </extension>  
975 </complexContent>  
976 </complexType>
```

977 2.4.5.1.1 Element <AttributeValue>

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

980 If the data content of an AttributeValue element is of a XML Schema simple type (e.g. interger,
981 string, etc) the data type MAY be declared explicitly by means of an `xsi:type` declaration in the

982 <AttributeValue> element. If the attribute value contains structured data the necessary data
983 elements may be defined in an extension schema introduced by means of the `xmlns=` mechanism.

984 The following schema fragment defines the <AttributeValue> element:

985 `<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 [SAMLBind] 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: 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="urn:oasis:names:tc:SAML:1.0:protocol "
  xmlns="http://www.w3.org/2001/XMLSchema "
  xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol "
  xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  elementFormDefault="unqualified ">
  <import namespace="urn:oasis:names:tc:SAML:1.0:assertion"
    schemaLocation="draft-sstc-schema-assertion-30.xsd"/>
  <import namespace="http://www.w3.org/2000/09/xmldsig#"
    schemaLocation="xmldsig-core-schema.xsd"/>
  <annotation>
    <documentation>draft-sstc-schema-protocol-30.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.

MajorVersion [Required]

The major version of this request. The identifier for the version of SAML defined in this specification is 1. Processing of this attribute is specified in Section 3.4.2.

1028 MinorVersion [Required]
 1029 The minor version of this request. The identifier for the version of SAML defined in this
 1030 specification is 0. Processing of this attribute is specified in Section 3.4.2.

1031 IssueInstant [Required]
 1032 The time instant of issue of the request. The time value is encoded in UTC as described in
 1033 section 1.2.1.

1034 <RespondWith> [Any Number]
 1035 Each <RespondWith> element specifies a type of response that is acceptable to the
 1036 requestor.

1037 <Signature> [Optional]
 1038 An XML Signature that authenticates the assertion, see section 5.

1039 The following schema fragment defines the **RequestAbstractType** complex type:

```

1040 <complexType name="RequestAbstractType" abstract="true">
1041   <sequence>
1042     <element ref="samlp:RespondWith"
1043       minOccurs="0" maxOccurs="unbounded"/>
1044     <element ref="ds:Signature" minOccurs="0"/>
1045   </sequence>
1046   <attribute name="RequestID" type="saml:IDType" use="required"/>
1047   <attribute name="MajorVersion" type="integer" use="required"/>
1048   <attribute name="MinorVersion" type="integer" use="required"/>
1049   <attribute name="IssueInstant" type="dateTime" use="required"/>
1050 </complexType>

```

1051 3.2.1.1. Element <RespondWith>

1052 The <RespondWith> element specifies the type of Statement the requestor wants from the
 1053 responder. Multiple <RespondWith> elements MAY be included to indicate that the requestor will
 1054 accept assertions containing any of the specified types. If no <RespondWith> element is given,
 1055 the responder may return assertions containing statements of any type.

1056 If the requestor sends one or more <RespondWith> elements, the responder MUST NOT respond
 1057 with assertions containing statements of any type not specified in one of the <RespondWith>
 1058 elements.

1059 NOTE: Inability to find assertions that meet <RespondWith> criteria should be treated identical to
 1060 any other query for which no assertions are available. In both cases a status of success would
 1061 normally be returned in the Response message, but no assertions to be found therein.

1062 <RespondWith> element values are XML QNames. The XML namespace and name specifically
 1063 refer to the namespace and element name of the Statement element, exactly as for the
 1064 saml:AuthorityKind attribute; see section 2.4.3.2. For example, a requestor that wishes to
 1065 receive assertions containing only attribute statements must specify
 1066 <RespondWith>saml:AttributeStatement</RespondWith>. To specify extension types,
 1067 the <RespondWith> element MUST contain exactly the extension element type as specified in the
 1068 xsi:type attribute on the corresponding element.

1069 The following schema fragment defines the <RespondWith> element:

```

1070 <element name="RespondWith" type=" QName "/>

```

1071 3.2.2. Element <Request>

1072 The <Request> element specifies a SAML request. It provides either a query or a request for a
 1073 specific assertion identified by <AssertionIDReference> or <AssertionArtifact>. It has

1074 the complex type **RequestType**, which extends **RequestAbstractType** by adding a choice of one
1075 of the following elements:

1076 <Query>

1077 An extension point that allows extension schemas to define new types of query.

1078 <SubjectQuery>

1079 An extension point that allows extension schemas to define new types of query that specify
1080 a single SAML subject.

1081 <AuthenticationQuery>

1082 Makes a query for authentication information.

1083 <AttributeQuery>

1084 Makes a query for attribute information.

1085 <AuthorizationDecisionQuery>

1086 Makes a query for an authorization decision.

1087 <AssertionIDReference> [One or more]

1088 Requests assertions by reference to its assertion identifier.

1089 <AssertionArtifact> [One or more]

1090 Requests assertions by supplying an assertion artifact that represents it.

1091 The following schema fragment defines the <Request> element and its **RequestType** complex
1092 type:

```
1093 <element name="Request" type="samlp:RequestType"/>  
1094 <complexType name="RequestType" >  
1095   <complexContent>  
1096     <extension base="samlp:RequestAbstractType">  
1097       <choice>  
1098         <element ref="samlp:Query"/>  
1099         <element ref="samlp:SubjectQuery"/>  
1100         <element ref="samlp:AuthenticationQuery"/>  
1101         <element ref="samlp:AttributeQuery"/>  
1102         <element ref="samlp:AuthorizationDecisionQuery"/>  
1103         <element ref="saml:AssertionIDReference" maxOccurs="unbounded"/>  
1104         <element ref="samlp:AssertionArtifact" maxOccurs="unbounded"/>  
1105       </choice>  
1106     </extension>  
1107   </complexContent>  
1108 </complexType>
```

1109 3.2.3. Element <AssertionArtifact>

1110 The <AssertionArtifact> element is used to specify the assertion artifact that represents an
1111 assertion.

1112 The following schema fragment defines the <AssertionArtifact> element:

```
1113 <element name="AssertionArtifact" type="string"/>
```

1114 3.3. Queries

1115 The following sections define the SAML constructs that contain query information.

1116 3.3.1. Element <Query>

1117 The <Query> element is an extension point that allows new SAML queries to be defined. Its
1118 **QueryAbstractType** is abstract; extension elements **MUST** use the `xsi:type` attribute to indicate

1119 the derived type. **QueryAbstractType** is the base type from which all SAML query elements are
1120 derived.

1121 The following schema fragment defines the <Query> element and its **QueryAbstractType**
1122 complex type:

```
1123 <element name="Query" type="samlp:QueryAbstractType" />  
1124 <complexType name="QueryAbstractType" abstract="true" />
```

1125 3.3.2. Element <SubjectQuery>

1126 The <SubjectQuery> element is an extension point that allows new SAML queries that specify a
1127 single SAML subject. Its **SubjectQueryAbstractType** complex type is abstract; extension elements
1128 MUST use the `xsi:type` attribute to indicate the derived type. **SubjectQueryAbstractType** adds
1129 the <Subject> element.

1130 The following schema fragment defines the <SubjectQuery> element and its
1131 **SubjectQueryAbstractType** complex type:

```
1132 <element name="SubjectQuery" type="samlp:SubjectQueryAbstractType" />  
1133 <complexType name="SubjectQueryAbstractType" abstract="true">  
1134 <complexContent>  
1135 <extension base="samlp:QueryAbstractType">  
1136 <sequence>  
1137 <element ref="saml:Subject" />  
1138 </sequence>  
1139 </extension>  
1140 </complexContent>  
1141 </complexType>
```

1142 3.3.3. Element <AuthenticationQuery>

1143 The <AuthenticationQuery> element is used to make the query “What assertions containing
1144 authentication statements are available for this subject?” A successful response will be in the form
1145 of assertions containing authentication statements.

1146 Note: The <AuthenticationQuery> MAY NOT be used as a request for a new authentication
1147 using credentials provided in the request. The <AuthenticationQuery> is a request for
1148 statements about authentication acts which have occurred in a previous interaction between the
1149 indicated principal and the Authentication Authority.

1150 This element is of type **AuthenticationQueryType**, which extends **SubjectQueryAbstractType**
1151 with the addition of the following element:

1152 <AuthenticationMethod> [Optional]

1153 A filter for possible responses. If it is present, the query made is “What assertions
1154 containing authentication statements do you have for this subject with the supplied
1155 authentication method?”

1156 In response to an authentication query, a responder returns assertions with authentication
1157 statements as follows:

1158 ?? First, rules given in section 3.4.4 for matching against the <Subject> element of the query
1159 identify the assertions that may be returned.

1160 ?? Further, if the <AuthenticationMethod> element is present in the query, at least one
1161 <AuthenticationMethod> element in the set of returned assertions MUST match. It is
1162 OPTIONAL for the complete set of all such matching assertions to be returned in the
1163 response.

1164 The <Subject> element in the returned assertions MUST be identical to the <Subject> element
1165 of the query. If the <ConfirmationMethod> element is present in the query, at least one

1166 <ConfirmationMethod> element in the response MUST match. It is OPTIONAL for the complete
1167 set of all such matching assertions to be returned in the response.

1168 The following schema fragment defines the <AuthenticationQuery> type and its
1169 **AuthenticationQueryType** complex type:

```
1170 <element name="AuthenticationQuery" type="samlp:AuthenticationQueryType"/>  
1171 <complexType name="AuthenticationQueryType">  
1172 <complexContent>  
1173 <extension base="samlp:SubjectQueryAbstractType">  
1174 <attribute name="AuthenticationMethod" type="anyURI"/>  
1175 </extension>  
1176 </complexContent>  
1177 </complexType>
```

1178 3.3.4. Element <AttributeQuery>

1179 The <AttributeQuery> element is used to make the query "Return the requested attributes for
1180 this subject." A successful response will be in the form of assertions containing attribute statements.
1181 This element is of type **AttributeQueryType**, which extends **SubjectQueryAbstractType** with the
1182 addition of the following element and attribute:

1183 Resource [Optional]

1184 The Resource attribute if present specifies that the attribute query is made in response to a
1185 specific authorization decision relating to the resource. The responder MAY use the
1186 resource attribute to establish the scope of the request. It is permitted for this attribute to
1187 have the value of the empty URI reference (""), and the meaning is defined to be "the start
1188 of the current document", as specified by [RFC 2396]§ 4.2.

1189 If the resource attribute is specified and the responder does not wish to support resource-
1190 specific attribute queries, or if the resource value provided is invalid or unrecognized, then it
1191 SHOULD respond with a top-level StatusCode value of Responder and a second-level
1192 code value of ResourceNotRecognized

1193 <AttributeDesignator> [Any Number] (see Section 2.4.5.1)

1194 Each <AttributeDesignator> element specifies an attribute whose value is to be
1195 returned. If no attributes are specified, it indicates that all attributes allowed by policy are
1196 requested.

1197 The following schema fragment defines the <AttributeQuery> element and its
1198 **AttributeQueryType** complex type:

```
1199 <element name="AttributeQuery" type="samlp:AttributeQueryType"/>  
1200 <complexType name="AttributeQueryType">  
1201 <complexContent>  
1202 <extension base="samlp:SubjectQueryAbstractType">  
1203 <sequence>  
1204 <element ref="saml:AttributeDesignator"  
1205 minOccurs="0" maxOccurs="unbounded"/>  
1206 </sequence>  
1207 <attribute name="Resource" type="anyURI reference" use="optional"/>  
1208 </extension>  
1209 </complexContent>  
1210 </complexType>
```

1211 3.3.5. Element <AuthorizationDecisionQuery>

1212 The <AuthorizationDecisionQuery> element is used to make the query "Should these
1213 actions on this resource be allowed for this subject, given this evidence?" A successful response
1214 will be in the form of assertions containing authorization decision statements. This element is of
1215 type **AuthorizationDecisionQueryType**, which extends **SubjectQueryAbstractType** with the
1216 addition of the following elements and attribute:

- 1217 Resource [Required]
 1218 A URI reference indicating the resource for which authorization is requested.
- 1219 <Action> [One or More]
 1220 The actions for which authorization is requested.
- 1221 <Evidence> [Any Number]
 1222 An assertion that the responder MAY rely on in making its response.

1223 The following schema fragment defines the <AuthorizationDecisionQuery> element and its
 1224 **AuthorizationDecisionQueryType** complex type:

```

1225 <element name="AuthorizationDecisionQuery"
1226 type="samlp:AuthorizationDecisionQueryType"/>
1227 <complexType name="AuthorizationDecisionQueryType">
1228 <complexContent>
1229 <extension base="samlp:SubjectQueryAbstractType">
1230 <sequence>
1231 <element ref="saml:Action" maxOccurs="unbounded"/>
1232 <element ref="saml:Evidence"
1233 minOccurs="0" maxOccurs="unbounded"/>
1234 </sequence>
1235 <attribute name="Resource" type="anyURI" use="required"/>
1236 </extension>
1237 </complexContent>
1238 </complexType>
  
```

1239 3.4. Responses

1240 The following sections define the SAML constructs that contain response information.

1241 3.4.1. Complex Type ResponseAbstractType

1242 All SAML responses are of types that are derived from the abstract **ResponseAbstractType**
 1243 complex type. This type defines common attributes and elements that are associated with all SAML
 1244 responses:

- 1245 ResponseID [Required]
 1246 An identifier for the response. It is of type **IDType**, and MUST follow the requirements
 1247 specified by that type for identifier uniqueness.
- 1248 InResponseTo [Optional]
 1249 A reference to the identifier of the request to which the response corresponds, if any. If the
 1250 response is not generated in response to a request, or if the RequestID of a request cannot
 1251 be determined (because the request is malformed), then this attribute MUST NOT be
 1252 present. Otherwise, it MUST be present and match the value of the corresponding
 1253 RequestID attribute.
- 1254 MajorVersion [Required]
 1255 The major version of this response. The identifier for the version of SAML defined in this
 1256 specification is 1. Processing of this attribute is specified in Section 3.4.4.
- 1257 MinorVersion [Required]
 1258 The minor version of this response. The identifier for the version of SAML defined in this
 1259 specification is 0. Processing of this attribute is specified in Section 3.4.4.
- 1260 IssueInstant [Optional]
 1261 The time instant of issue of the request. The time value is encoded in UTC as described in
 1262 section 1.2.1.

1263 Recipient [Optional]
1264 The intended recipient of this response. This is useful to prevent malicious forwarding of
1265 responses to unintended recipients, a protection that is required by some use profiles. It is
1266 set by the generator of the response to a URI reference that identifies the intended
1267 recipient. If present, the actual recipient MUST check that the URI reference identifies the
1268 recipient or a resource managed by the recipient. If it does not, the response MUST be
1269 discarded.

1270 <Signature> [Optional]
1271 An XML Signature that authenticates the assertion, see section 5.

1272 The following schema fragment defines the **ResponseAbstractType** complex type:

```
1273 <complexType name="ResponseAbstractType" abstract="true">  
1274 <sequence>  
1275 <element ref="ds:Signature" minOccurs="0"/>  
1276 </sequence>  
1277 <attribute name="ResponseID" type="saml:IDType" use="required"/>  
1278 <attribute name="InResponseTo" type="saml:IDReferenceType"  
1279 use="optional"/>  
1280 <attribute name="MajorVersion" type="integer" use="required"/>  
1281 <attribute name="MinorVersion" type="integer" use="required"/>  
1282 <attribute name="IssueInstant" type="dateTime" use="required"/>  
1283 <attribute name="Recipient" type="anyURI" use="optional"/>  
1284 </complexType>
```

1285 3.4.2. Element <Response>

1286 The <Response> element specifies the status of the corresponding SAML request and a list of
1287 zero or more assertions that answer the request. It has the complex type **ResponseType**, which
1288 extends **ResponseAbstractType** by adding the following elements (in an unbounded mixture):

1289 <Status> [Required] (see Section 3.4.3)
1290 A code representing the status of the corresponding request.

1291 <Assertion> [Any Number] (see Section 2.3.2)
1292 Specifies an assertion by value.

1293 The following schema fragment defines the <Response> element and its **ResponseType** complex
1294 type:

```
1295 <element name="Response" type="samlp:ResponseType"/>  
1296 <complexType name="ResponseType">  
1297 <complexContent>  
1298 <extension base="samlp:ResponseAbstractType">  
1299 <sequence>  
1300 <element ref="samlp:Status"/>  
1301 <element ref="saml:Assertion"  
1302 minOccurs="0" maxOccurs="unbounded"/>  
1303 </sequence>  
1304 </extension>  
1305 </complexContent>  
1306 </complexType>
```

1307 3.4.3. Element <Status>

1308 The <Status> element :

1309 <StatusCode> [Required]
1310 A code representing the status of the corresponding request.

1311 <StatusMessage> [Any Number]
1312 A message which MAY be returned to an operator.

1313 <StatusDetail> [Optional]
1314 Specifies additional information concerning an error condition.
1315 The following schema fragment defines the <Status> element and its **StatusType** complex type:

```
1316 <element name="Status" type="samlp:StatusType"/>  
1317 <complexType name="StatusType">  
1318 <sequence>  
1319 <element ref="samlp:StatusCode"/>  
1320 <element ref="samlp:StatusMessage"  
1321 minOccurs="0" maxOccurs="unbounded"/>  
1322 <element ref="samlp:StatusDetail" minOccurs="0"/>  
1323 </sequence>  
1324 </complexType>
```

1325 3.4.3.1. Element <StatusCode>

1326 The <StatusCode> element specifies one or more nested codes representing the status of the
1327 corresponding request. top-most code value MUST be one of the values defined below.
1328 Subsequent nested code values, if present, may provide more specific information concerning a
1329 particular error.

1330 Value [Required]
1331 The status code value as defined below.

1332 <StatusCode> [Optional]
1333 An optional subordinate status code value that provides more specific information on an
1334 error condition.

1335 The following top-level **StatusCode** Value QNames are defined. The responder MUST NOT
1336 include a code not listed below except by nesting it below one of the listed values.

1337 Success
1338 The request succeeded.

1339 VersionMismatch
1340 The receiver could not process the request because the version was incorrect.

1341 Receiver
1342 The request could not be performed due to an error at the receiving end.

1343 Sender
1344 The request could not be performed due to an error in the sender or in the request

1345 The following second-level status codes are referenced at various places in the specification.
1346 Additional subcodes MAY be defined in future versions of the SAML specification.

1347 RequestVersionTooHigh
1348 The protocol version specified in the request is a major upgrade from the highest protocol
1349 version supported by the responder.

1350 RequestVersionTooLow
1351 The responder cannot respond to the particular request using the SAML version specified
1352 in the request because it is too low.

1353 RequestVersionDeprecated
1354 The responder does not respond to any requests with the protocol version specified in the
1355 request.

1356 TooManyResponses
1357 The response would contain more elements than the responder will return.

1358 RequestDenied
1359 The responder is able to process the request but has chosen not to respond. MAY be used

1360 when the responder is concerned about the security context of the request or the sequence
1361 of requests received from a particular client.

1362 All status code values defined in this document are QNames associated with the SAML protocol
1363 namespace [SAML] and MUST be prefixed appropriately when they appear in SAML messages.
1364 SAML extensions and SAML Responders are free to define more specific status codes in other
1365 namespaces, but MAY NOT define additional codes in either the SAML assertion or protocol
1366 namespaces.

1367 The QNames defined as status codes SHOULD only be used in the StatusCode element's Value
1368 attribute and have the above semantics only in that context.

1369 The following schema fragment defines the <StatusCode> element and its **StatusCodeType**
1370 complex type:

```
1371 <element name="StatusCode" type="samlp:StatusCodeType" />  
1372 <complexType name="StatusCodeType">  
1373 <sequence>  
1374 <element ref="samlp:StatusCode" minOccurs="0" />  
1375 </sequence>  
1376 <attribute name="Value" type="QName" use="required" />  
1377 </complexType>
```

1378 3.4.3.2. Element <StatusMessage>

1379 The <StatusMessage> element specifies a message that MAY be returned to an operator:

1380 The following schema fragment defines the <StatusMessage> element and its
1381 **StatusMessageType** complex type:

```
1382 <element name="StatusMessage" type="string" />
```

1383 3.4.3.3. Element <StatusDetail>

1384 The <StatusDetail> element MAY be used to specify additional information concerning an error
1385 condition.

1386 The following schema fragment defines the <StatusDetail> element and its **StatusDetailType**
1387 complex type:

```
1388 <element name="StatusDetail" type="samlp:StatusDetailType" />  
1389 <complexType name="StatusDetailType">  
1390 <sequence>  
1391 <any namespace="##any"  
1392 processContents="lax" minOccurs="0" maxOccurs="unbounded" />  
1393 </sequence>  
1394 </complexType>
```

1395 3.4.4. Responses to <AuthenticationQuery> and <AttributeQuery>

1396 Responses to Authentication and Attribute queries are constructed by matching against the
1397 <saml:Subject> element found within the <AuthenticationQuery> or <AttributeQuery>
1398 elements. In response to these queries, every assertion returned by a SAML responder MUST
1399 contain at least one statement whose <saml:Subject> element **strongly matches** the
1400 <saml:Subject> element found in the query.

1401 A <saml:Subject> element S1 strongly matches S2 if and only if:

- 1402 1 If S2 includes a <saml:NameIdentifier> element, then S1 must include an identical
1403 <saml:NameIdentifier> element.
- 1404 2 If S2 includes a <saml:SubjectConfirmation> element, then S1 must include an
1405 identical <saml:SubjectConfirmation> element.

1406 If the responder cannot provide an assertion with any statement(s) satisfying the constraints
1407 expressed by a query, the <saml:Response> element MUST NOT contain an <assertion> element
1408 and MUST include a <saml:StatusCode> with value "Success". It MAY return a
1409 <saml:StatusMessage> with additional information.

4. SAML Versioning

1410

1411 SAML version information appears in the following elements:

1412 ?? <Assertion>

1413 ?? <Request>

1414 ?? <Response>

1415 The version numbering of the SAML assertion is independent of the version number of the SAML
1416 request-response protocol. The version information for each consists of a major version number
1417 and a minor version number, both of which are integers. In accordance with industry practice a
1418 version number SHOULD be presented to the user in the form *Major.Minor*. This document defines
1419 SAML Assertions 1.0 and SAML Protocol 1.0.

1420 The version number $Major_B.Minor_B$ is higher than the version number $Major_A.Minor_A$ if and only if:

1421 $Major_B > Major_A ? ((Major_B = Major_A) ? Minor_B > Minor_A)$

1422 Each revision of SAML SHALL assign version numbers to assertions, requests, and responses that
1423 are the same as or higher than the corresponding version number in the SAML version that
1424 immediately preceded it.

1425 New versions of SAML SHALL assign new version numbers as follows:

1426 ?? **Documentation change:** $(Major_B = Major_A) ? (Minor_B > Minor_A)$

1427 If the major and minor version numbers are unchanged, the new version *B* only introduces
1428 changes to the documentation that raise no compatibility issues with an implementation of
1429 version *A*.

1430 ?? **Minor upgrade:** $(Major_B = Major_A) ? (Minor_B > Minor_A)$

1431 If the major version number of versions *A* and *B* are the same and the minor version
1432 number of *B* is higher than that of *A*, the new SAML version MAY introduce changes to the
1433 SAML schema and semantics but any changes that are introduced in *B* SHALL be
1434 compatible with version *A*.

1435 ?? **Major upgrade:** $Major_B > Major_A$

1436 If the major version of *B* number is higher than the major version of *A*, Version *B* MAY
1437 introduce changes to the SAML schema and semantics that are incompatible with *A*.

4.1. Assertion Version

1438

1439 A SAML authority MUST NOT issue any assertion whose version number is not supported.

1440 | A SAML authority-relying party MUST reject any assertion whose major version number is not
1441 supported.

1442 | A SAML authority-relying party MAY reject any assertion whose version number is higher than the
1443 highest supported version.

4.2. Request Version

1444

1445 A SAML authority SHOULD issue requests that specify the highest SAML version supported by
1446 both the sender and recipient.

1447 If the SAML authority does not know the capabilities of the recipient it should assume that it
1448 supports the highest SAML version supported by the sender.

1449 **4.3. Response Version**

1450 A SAML authority **MUST NOT** issue responses that specify a higher SAML version number than the
1451 corresponding request.

1452 A SAML authority **MUST NOT** issue a response that has a major version number that is lower than
1453 the major version number of the corresponding request except to report the error
1454 `RequestVersionTooHigh`.

1455 An error response resulting from incompatible protocol versions **MUST** result in reporting a top-level
1456 `Statuscode` value of `VersionMismatch`, and **MAY** result in reporting one of the following second-
1457 level values:

1458 `RequestVersionTooHigh`

1459 The protocol version specified in the request is a major upgrade from the highest protocol
1460 version supported by the responder.

1461 `RequestVersionTooLow`

1462 The responder cannot respond to the particular request using the SAML version specified
1463 in the request because it is too low.

1464 `RequestVersionDeprecated`

1465 The responder does not respond to any requests with the protocol version specified in the
1466 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 asserting party (AP). This supports :

- (1) Message integrity
- (2) Authentication of the asserting party to a relying party (RP)
- (3) If the signature is based on the asserting party'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. **[XMLSig]** provides a framework for signing in XML and is the framework of choice. However, signing may also take place in the context of S/MIME or Java objects that contain SAML documents. One goal is to ensure compatibility with this type of "foreign" digital signing.

?? 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 asserting party.
- (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.

1509 **5.2. Request/Response Signing**

1510 All SAML requests and responses MAY be signed using the XML Signature. This is reflected in the
1511 schema – Section 3.2 & 3.4.

1512 **5.3. Signature Inheritance**

1513 **5.3.1. Rationale**

1514 SAML assertions may be embedded within request or response messages or other XML
1515 messages, which may be signed. Request or response messages may themselves be contained
1516 within other messages that are based on other XML messaging frameworks (e.g., SOAP) and the
1517 composite object may be the subject of a signature. Another possibility is that SAML assertions or
1518 request/response messages are embedded within a non-XML messaging object (e.g., MIME
1519 package) and signed.

1520 In such a case, the SAML sub-message (Assertion, request, response) may be viewed as inheriting
1521 a signature from the "super-signature" over the enclosing object, provided certain constraints are
1522 met.

1523 (1) An assertion may be viewed as inheriting a signature from a super signature, if the super
1524 signature applies all the elements within the assertion.

1525 A SAML request or response may be viewed as inheriting a signature from a super signature, if the
1526 super signature applies to all of the elements within the response.

1527 **5.3.2. Rules for SAML Signature Inheritance**

1528 Signature inheritance occurs when SAML message (assertion/request/response) is not signed but
1529 is enclosed within signed SAML such that the signature applies to all of the elements within the
1530 message. In such a case, the SAML message is said to inherit the signature and may be
1531 considered equivalent to the case where it is explicitly signed. The SAML message inherits the
1532 "closest enclosing signature".

1533 But if SAML messages need to be passed around by themselves, or embedded in other messages,
1534 they would need to be signed as per section 5.1

1535 **5.4. XML Signature Profile**

1536 The XML Signature [**XMLSig**] specification calls out a general XML syntax for signing data with
1537 many flexibilities and choices. This section details the constraints on these facilities so that SAML
1538 processors do not have to deal with the full generality of XML Signature processing.

1539 **5.4.1. Signing formats**

1540 XML Signature has three ways of representing signature in a document viz: enveloping, enveloped
1541 and detached.

1542 SAML assertions and protocols MUST use the enveloped signatures for signing assertions and
1543 protocols. SAML processors should support use of RSA signing and verification for public key
1544 operations.

1545 **5.4.2. CanonicalizationMethod**

1546 XML Signature REQUIRES the Canonical XML (omits comments)
1547 (<http://www.w3.org/TR/2001/REC-xml-c14n-20010315>). SAML implementations SHOULD use
1548 Canonical XML with no comments.

1549 **5.4.3. Transforms**

1550 [XMLSig] REQUIRES the enveloped signature transform
1551 <http://www.w3.org/2000/09/xmlsig#enveloped-signature>

1552 **5.4.4. KeyInfo**

1553 SAML does not restrict or impose any restrictions in this area. Therefore following [XMLSig]
1554 keyInfo may be absent.

1555 **5.4.5. Binding between statements in a multi-statement assertion**

1556 Use of signing does not affect semantics of statements within assertions in any way, as stated in
1557 this document Sections 1 through 4.

1558 **6. SAML Extensions**

1559 The SAML schemas support extensibility. An example of an application that extends SAML
1560 assertions is the XTAML system for management of embedded trust roots **[XTAML]**. The following
1561 sections explain how to use the extensibility features in SAML to create extension schemas.

1562 Note that elements in the SAML schemas are not blocked from substitution, so that all SAML
1563 elements MAY serve as the head element of a substitution group. Also, types are not defined as
1564 *final*, so that all SAML types MAY be extended and restricted. The following sections discuss
1565 only elements that have been specifically designed to support extensibility.

1566 **6.1. Assertion Schema Extension**

1567 The SAML assertion schema is designed to permit separate processing of the assertion package
1568 and the statements it contains, if the extension mechanism is used for either part.

1569 The following elements are intended specifically for use as extension points in an extension
1570 schema; their types are set to *abstract*, so that the use of an *xsi:type* attribute with these
1571 elements is REQUIRED:

1572 ?? <Assertion>

1573 ?? <Condition>

1574 ?? <Statement>

1575 ?? <SubjectStatement>

1576 ?? <AdviceElement>

1577 In addition, the following elements that are directly usable as part of SAML MAY be extended:

1578 ?? <AuthenticationStatement>

1579 ?? <AuthorizationDecisionStatement>

1580 ?? <AttributeStatement>

1581 ?? <AudienceRestrictionCondition>

1582 Finally, the following elements are defined to allow elements from arbitrary namespaces within
1583 them, which serves as a built-in extension point without requiring an extension schema:

1584 ?? <AttributeValue>

1585 ?? <Advice>

1586 **6.2. Protocol Schema Extension**

1587 The following elements are intended specifically for use as extension points in an extension
1588 schema; their types are set to *abstract*, so that the use of an *xsi:type* attribute with these
1589 elements is REQUIRED:

1590 ?? <Query>

1591 ?? <SubjectQuery>

1592 In addition, the following elements that are directly usable as part of SAML MAY be extended:

1593 ?? <Request>

1594 ?? <AuthenticationQuery>
1595 ?? <AuthorizationDecisionQuery>
1596 ?? <AttributeQuery>
1597 ?? <Response>

1598 6.3. Use of Type Derivation and Substitution Groups

1599 W3C XML Schema [Schema1] provides two principal mechanisms for specifying an element of an
1600 extended type: type derivation and substitution groups.

1601 For example, a <Statement> element can be assigned the type **NewStatementType** by means of
1602 the `xsi:type` attribute. For such an element to be schema-valid, **NewStatementType** needs to be
1603 derived from **StatementType**. The following example of a SAML assertion assumes that the
1604 extension schema (represented by the `new:` prefix) has defined this new type:

```
1605 <saml:Assertion ...>  
1606   <saml:Statement xsi:type="new:NewStatementType">  
1607     ...  
1608   </saml:Statement>  
1609 </saml:Assertion>
```

1610 Alternatively, the extension schema can define a <NewStatement> element that is a member of a
1611 substitution group that has <Statement> as a head element. For the substituted element to be
1612 schema-valid, it needs to have a type that matches or is derived from the head element's type. The
1613 following is an example of an extension schema fragment that defines this new element:

```
1614 <xsd:element "NewStatement" type="new:NewStatementType"  
1615   substitutionGroup="saml:Statement" />
```

1616 The substitution group declaration allows the <NewStatement> element to be used anywhere the
1617 SAML <Statement> element can be used. The following is an example of a SAML assertion that
1618 uses the extension element:

```
1619 <saml:Assertion ...>  
1620   <new:NewStatement>  
1621     ...  
1622   </new:NewStatement>  
1623 </saml:Assertion>
```

1624 The choice of extension method has no effect on the semantics of the XML document but does
1625 have implications for interoperability.

1626 The advantages of type derivation are as follows:

1627 ?? A document can be more fully interpreted by a parser that does not have access to the
1628 extension schema because a "native" SAML element is available.

1629 ?? At the time of writing, some W3C XML Schema validators do not support substitution
1630 groups, whereas the `xsi:type` attribute is widely supported.

1631 The advantage of substitution groups is that a document can be explained without the need to
1632 explain the functioning of the `xsi:type` attribute.

7. SAML-Defined Identifiers

1633

1634 The following sections define URI-based identifiers for common authentication protocols and
1635 actions.

1636 Where possible an existing URN is used to specify a protocol. In the case of IETF protocols the
1637 URN of the most current RFC that specifies the protocol is used. URI references created
1638 specifically for SAML have the initial stem:

1639 `urn:oasis:names:tc:SAML:1.0:`

7.1. Authentication Method ~~and Confirmation Method~~ Identifiers

1641

1642 The <AuthenticationMethod> and <SubjectConfirmationMethod> elements perform
1643 different functions within the SAML architecture, although both can ~~contain refer to the same~~
1644 ~~underlying mechanisms~~~~some of the same values~~. <AuthenticationMethod> is a part of an
1645 Authentication Statement, which describes an authentication act which occurred in the past. The
1646 <AuthenticationMethod> indicates how that authentication was done. Note that the
1647 authentication statement does not provide the means to perform that authentication, such as a
1648 password, key or certificate.

1649 In contrast, <SubjectConfirmationMethod> is a part of the <SubjectConfirmation>, which
1650 is used to allow the Relying Party to confirm that the request or message came from the System
1651 Entity that corresponds to the Subject in the statement. The <SubjectConfirmationMethod>
1652 indicates the method ~~which that~~ the Relying Party can use to do this in the future. This may or may
1653 not have any relationship to an authentication that was performed previously. Unlike the
1654 Authentication Method, the <SubjectConfirmationMethod> ~~will usually may~~ be accompanied
1655 with some piece of information, such as a certificate or key, which will allow the Relying Party to
1656 perform the necessary check.

1657 ~~Subject Confirmation Methods are defined in the SAML Profile or Profiles in which they are used~~
1658 ~~[SAMLBind]. Additional methods may be added by defining new profiles or by private~~
1659 ~~agreement. There are many <SubjectConfirmationMethod>, because there are many different~~
1660 ~~SAML usage scenarios. A few examples are:~~

1661 ~~The following identifiers refer to SAML specified Authentication methods.~~

1662 ~~1. A user logs in with a password, but a temporary passcode or cookie is issued for confirmation~~
1663 ~~purposes to avoid repeated exposure of the long term password.~~

1664 ~~2. There is no login, but an application request is digitally signed. The associated public key is used~~
1665 ~~for confirmation.~~

1666 ~~3. The user logs in using Kerberos and a Kerberos ticket is used subsequently for confirmation.~~
1667 ~~Notice that in this case although both the Authentication Method and the~~
1668 ~~<SubjectConfirmationMethod> are Kerberos, what happens at each step is actually different.~~
1669 ~~(See [RFC 1510])~~

1670 ~~The following identifiers are defined to refer to common authentication protocols.~~ Where Base64
1671 encoding is specified the data is encoded as specified by [RFC 2045].

7.1.1.SAML Artifact (SHA-1):

1673 **URI:** `urn:oasis:names:tc:SAML:1.0:cm:artifact-sha1`

1674 ~~<SubjectConfirmationData>: Base64 (SHA1 (Artifact))~~
1675 ~~The subject of the assertion is the party that can present a SAML Artifact such that the SHA1 digest~~
1676 ~~of the specified artifact matches the value specified in <SubjectConfirmationData>.~~

1677 **7.1.2. Holder of Key:**

1678 ~~URI: urn:oasis:names:tc:SAML:1.0:cm:Holder-Of-Key~~

1679 ~~<ds:KeyInfo>: Any cryptographic key~~

1680 ~~The subject of the assertion is the party that can demonstrate that it is the holder of the private~~
1681 ~~component of the key specified in <ds:KeyInfo>.~~

1682 **7.1.3. Bearer Indication:**

1683 ~~URI: urn:oasis:names:tc:SAML:1.0:cm:BearerIndication~~

1684 ~~The subject of the assertion is the bearer of the assertion.~~

1685 **7.1.4. Sender Vouches:**

1686 ~~URI: urn:oasis:names:tc:SAML:1.0:cm:sender-vouches~~

1687 ~~Indicates that no other information is available about the context of use of the assertion. The~~
1688 ~~Relying party SHOULD utilize other means to determine if it should process the assertion further.~~

1689 **7.1.5.7.1.1. Password (Pass-Through):**

1690 ~~URI: urn:oasis:names:tc:SAML:1.0:acm:password~~

1691 ~~<SubjectConfirmationData>: Base64 (Password)~~

1692 ~~The subject of the assertion is the party that can present the password value specified in~~
1693 ~~<SubjectConfirmationData>.~~

1694 ~~The username of the subject is specified by means of the <NameIdentifier> element. The~~
1695 ~~authentication was performed by means of a password.~~

1696 **7.1.6. Password (One-Way-Function SHA-1):**

1697 ~~URI: urn:oasis:names:tc:SAML:1.0:cm:password-sha1~~

1698 ~~<SubjectConfirmationData>: Base64 (SHA1 (Password))~~

1699 ~~The subject of the assertion is the party that can present the password such that the SHA1 digest of~~
1700 ~~the specified password matches the value specified in <SubjectConfirmationData>.~~

1701 ~~The username of the subject is specified by means of the <NameIdentifier> element.~~

1702 **7.1.7.7.1.2. Kerberos**

1703 ~~URI: urn:ietf:rfc:1510~~

1704 ~~<SubjectConfirmationData>: A Kerberos Ticket~~

1705 ~~The subject is authenticated~~ authentication was performed by means of the Kerberos protocol **[RFC**
1706 **1510][RFC 1510]**, an instantiation of the Needham-Schroeder symmetric key authentication
1707 mechanism **[Needham78] [Needham78]**.

1708 **7.1.3. SSL/TLS Certificate Based Client Authentication:**

1709 **URI:** <urn:ietf:rfc:2246>

1710 [The authentication was performed using either the SSL or TLS protocol with certificate based client](#)
1711 [authentication. TLS is described in \[RFC 2246\].](#)

1712 **7.1.4. X.509 Public Key**

1713 **URI:** <urn:oasis:names:tc:SAML:1.0:am:X509-PKI>

1714 [The authentication was performed by some \(unspecified\) mechanism on a key authenticated by](#)
1715 [means of an X.509 PKI \[X.509\]\[PKIX\]. It may have been one of the mechanisms for which a more](#)
1716 [specific identifier has been defined below.](#)

1717 **7.1.5. PGP Public Key**

1718 **URI:** <urn:oasis:names:tc:SAML:1.0:am:PGP>

1719 [The authentication was performed by some \(unspecified\) mechanism on a key authenticated by](#)
1720 [means of a PGP web of trust \[PGP\]. It may have been one of the mechanisms for which a more](#)
1721 [specific identifier has been defined below.](#)

1722 **7.1.6. SPKI Public Key**

1723 **URI:** <urn:oasis:names:tc:SAML:1.0:am:SPKI>

1724 [The authentication was performed by some \(unspecified\) mechanism on a key authenticated by](#)
1725 [means of a SPKI PKI \[SPKI\]. It may have been one of the mechanisms for which a more specific](#)
1726 [identifier has been defined below.](#)

1727 **7.1.7. XKMS Public Key**

1728 **URI:** <urn:oasis:names:tc:SAML:1.0:am:XKMS>

1729 [The authentication was performed by some \(unspecified\) mechanism on a key authenticated by](#)
1730 [means of a XKMS trust service \[XKMS\]. It may have been one of the mechanisms for which a more](#)
1731 [specific identifier has been defined below.](#)

1732 **7.1.8. SSL/TLS Certificate Based Client Authentication:**

1733 **URI:** <urn:ietf:rfc:2246>

1734 [<ds:KeyInfo>: Any cryptographic key](#)

1735 **7.1.9. Object Authenticator (SHA-1):**

1736 **URI:** <urn:oasis:names:tc:SAML:1.0:cm:object-sha1>

1737 [<SubjectConfirmationData>: Base64 \(SHA1 \(Object \)\)](#)

1738 [This authenticator element is the result of computing a digest, using the SHA-1 hash algorithm. It is](#)
1739 [used when the subject can be represented as a binary string, for example when it is an XML](#)
1740 [document or the disk image of executable code. Any preprocessing of the subject prior to](#)
1741 [computation of the digest is out of scope. The name of the subject should be conveyed in an](#)
1742 [accompanying NameIdentifier element.](#)

1743

7.1.10.PKCS#7

1744

URI: urn:ietf:rfc:2315

1745

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

1746

~~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 indicates whether the signer is authorized for the access request represented by the object element.~~

1747

1748

1749

1750

1751

1752

1753

1754

1755

7.1.11.Cryptographic Message Syntax

1756

URI: urn:ietf:rfc:2630

1757

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

1758

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

1759

7.1.12.7.1.8. XML Digital Signature

1760

URI: urn:ietf:rfc:3075

1761

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

1762

~~<ds:KeyInfo>: A cryptographic signing key~~

1763

~~The authentication was performed by means of an XML digital signature [RFC 3075]. This authenticator element is signed data in XML Signature format. See also 7.1.10~~

1764

1765

7.2. Action Namespace Identifiers

1766

The following identifiers MAY be used in the Namespace attribute of the <Action> element (see Section 2.4.4.1) to refer to common sets of actions to perform on resources.

1767

1768

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

1769

URI: urn:oasis:names:tc:SAML:1.0:action:rwdc

1770

Defined actions:

1771

Read Write Execute Delete Control

1772

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

1773

Read

1774

The subject may read the resource

1775

Write

1776

The subject may modify the resource

1777

Execute

1778

The subject may execute the resource

1779 Delete
1780 The subject may delete the resource
1781 Control
1782 The subject may specify the access control policy for the resource

1783 **7.2.2. Read/Write/Execute/Delete/Control with Negation:**

1784 **URI:** urn:oasis:names:tc:SAML:1.0:action:rwedc-negation

1785 Defined actions:

1786 Read Write Execute Delete Control ~Read ~Write ~Execute ~Delete ~Control

1787 The actions specified in section 7.2.1 are interpreted in the same manner described there. Actions
1788 prefixed with a tilde ~ are negated permissions and are used to affirmatively specify that the stated
1789 permission is denied. Thus a subject described as being authorized to perform the action ~Read is
1790 affirmatively denied read permission.

1791 A SAML authority MUST NOT authorize both an action and its negated form.

1792 **7.2.3. Get/Head/Put/Post:**

1793 **URI:** urn:oasis:names:tc:SAML:1.0:action:ghpp

1794 Defined actions:

1795 GET HEAD PUT POST

1796 These actions bind to the corresponding HTTP operations. For example a subject authorized to
1797 perform the GET action on a resource is authorized to retrieve it.

1798 The GET and HEAD actions loosely correspond to the conventional read permission and the PUT
1799 and POST actions to the write permission. The correspondence is not exact however since a HTTP
1800 GET operation may cause data to be modified and a POST operation may cause modification to a
1801 resource other than the one specified in the request. For this reason a separate Action URI
1802 reference specifier is provided.

1803 **7.2.4. UNIX File Permissions:**

1804 **URI:** urn:oasis:names:tc:SAML:1.0:action:unix

1805 The defined actions are the set of UNIX file access permissions expressed in the numeric (octal)
1806 notation.

1807 The action string is a four digit numeric code:

1808 *extended user group world*

1809 Where the *extended* access permission has the value

1810 +2 if sgid is set

1811 +4 if suid is set

1812 The *user group* and *world* access permissions have the value

1813 +1 if execute permission is granted

1814 +2 if write permission is granted

1815 +4 if read permission is granted

1816 For example 0754 denotes the UNIX file access permission: user read, write and execute, group
1817 read and execute and world read.

8. SAML Schema Listings

1818

1819 The following sections contain complete listings of the assertion and protocol schemas for SAML.

8.1. Assertion Schema

1820

1821 Following is a complete listing of the SAML assertion schema [SAML-XSD].

```
1822 <?xml version="1.0" encoding="UTF-8"?>
1823 <!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Phill Hallam-Baker
1824 (VeriSign Inc.) -->
1825 <schema
1826     targetNamespace="urn:oasis:names:tc:SAML:1.0:assertion"
1827     xmlns="http://www.w3.org/2001/XMLSchema"
1828     xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
1829     xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
1830     elementFormDefault="unqualified">
1831     <import namespace="http://www.w3.org/2000/09/xmldsig#"
1832           schemaLocation="xmldsig-core-schema.xsd"/>
1833     <annotation>
1834         <documentation>draft-sstc-schema-assertion-30.xsd</documentation>
1835     </annotation>
1836     <simpleType name="IDType">
1837         <restriction base="string"/>
1838     </simpleType>
1839     <simpleType name="IDReferenceType">
1840         <restriction base="string"/>
1841     </simpleType>
1842     <simpleType name="DecisionType">
1843         <restriction base="string">
1844             <enumeration value="Permit"/>
1845             <enumeration value="Deny"/>
1846             <enumeration value="Indeterminate"/>
1847         </restriction>
1848     </simpleType>
1849     <element name="AssertionIDReference" type="saml:IDReferenceType"/>
1850     <element name="Assertion" type="saml:AssertionType"/>
1851     <complexType name="AssertionType">
1852         <sequence>
1853             <element ref="saml:Conditions" minOccurs="0"/>
1854             <element ref="saml:Advice" minOccurs="0"/>
1855             <choice maxOccurs="unbounded">
1856                 <element ref="saml:Statement"/>
1857                 <element ref="saml:SubjectStatement"/>
1858                 <element ref="saml:AuthenticationStatement"/>
1859                 <element ref="saml:AuthorizationDecisionStatement"/>
1860                 <element ref="saml:AttributeStatement"/>
1861             </choice>
1862             <element ref="ds:Signature" minOccurs="0"/>
1863         </sequence>
1864         <attribute name="MajorVersion" type="integer" use="required"/>
1865         <attribute name="MinorVersion" type="integer" use="required"/>
1866         <attribute name="AssertionID" type="saml:IDType" use="required"/>
1867         <attribute name="Issuer" type="string" use="required"/>
1868         <attribute name="IssueInstant" type="dateTime" use="required"/>
1869     </complexType>
1870     <element name="Conditions" type="saml:ConditionsType"/>
1871     <complexType name="ConditionsType">
1872         <choice minOccurs="0" maxOccurs="unbounded">
1873             <element ref="saml:AudienceRestrictionCondition"/>
1874             <element ref="saml:Condition"/>
```

```

1875     </choice>
1876     <attribute name="NotBefore" type="dateTime" use="optional"/>
1877     <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
1878 </complexType>
1879 <element name="Condition" type="saml:ConditionAbstractType"/>
1880 <complexType name="ConditionAbstractType" abstract="true"/>
1881 <element name="AudienceRestrictionCondition"
1882     type="saml:AudienceRestrictionConditionType"/>
1883 <complexType name="AudienceRestrictionConditionType">
1884     <complexContent>
1885         <extension base="saml:ConditionAbstractType">
1886             <sequence>
1887                 <element ref="saml:Audience" maxOccurs="unbounded"/>
1888             </sequence>
1889         </extension>
1890     </complexContent>
1891 </complexType>
1892 <element name="Audience" type="anyURI"/>
1893 <element name="Advice" type="saml:AdviceType"/>
1894 <complexType name="AdviceType">
1895     <choice minOccurs="0" maxOccurs="unbounded">
1896         <element ref="saml:AssertionIDReference"/>
1897         <element ref="saml:Assertion"/>
1898         <any namespace="##other" processContents="lax"/>
1899     </choice>
1900 </complexType>
1901 <element name="Statement" type="saml:StatementAbstractType"/>
1902 <complexType name="StatementAbstractType" abstract="true"/>
1903 <element name="SubjectStatement" type="saml:SubjectStatementAbstractType"/>
1904 <complexType name="SubjectStatementAbstractType" abstract="true">
1905     <complexContent>
1906         <extension base="saml:StatementAbstractType">
1907             <sequence>
1908                 <element ref="saml:Subject"/>
1909             </sequence>
1910         </extension>
1911     </complexContent>
1912 </complexType>
1913 <element name="Subject" type="saml:SubjectType"/>
1914 <complexType name="SubjectType">
1915     <choice>
1916         <sequence>
1917             <element ref="saml:NameIdentifier"/>
1918             <element ref="saml:SubjectConfirmation" minOccurs="0"/>
1919         </sequence>
1920         <element ref="saml:SubjectConfirmation"/>
1921     </choice>
1922 </complexType>
1923 <element name="NameIdentifier" type="saml:NameIdentifierType"/>
1924 <complexType name="NameIdentifierType">
1925     <simpleContent>
1926         <extension base="string">
1927             <attribute name="NameQualifier" type="string" use="optional"/>
1928             <attribute name="Format" type="anyURI" use="optional"/>
1929         </extension>
1930     </simpleContent>
1931 </complexType>
1932 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>
1933 <complexType name="SubjectConfirmationType">
1934     <sequence>
1935         <element ref="saml:ConfirmationMethod" maxOccurs="unbounded"/>
1936         <element ref="saml:SubjectConfirmationData" minOccurs="0"/>
1937         <element ref="ds:KeyInfo" minOccurs="0"/>

```

```

1938     </sequence>
1939 </complexType>
1940 <element name="SubjectConfirmationData" type="stringanyType" />
1941 <element name="ConfirmationMethod" type="anyURI" />
1942 <element name="AuthenticationStatement"
1943     type="saml:AuthenticationStatementType" />
1944 <complexType name="AuthenticationStatementType" >
1945     <complexContent>
1946         <extension base="saml:SubjectStatementAbstractType" >
1947             <sequence>
1948                 <element ref="saml:SubjectLocality" minOccurs="0" />
1949                 <element ref="saml:AuthorityBinding"
1950                     minOccurs="0" maxOccurs="unbounded" />
1951             </sequence>
1952             <attribute name="AuthenticationMethod" type="anyURI" />
1953             <attribute name="AuthenticationInstant" type="dateTime" />
1954         </extension>
1955     </complexContent>
1956 </complexType>
1957 <element name="SubjectLocality"
1958     type="saml:SubjectLocalityType" />
1959 <complexType name="SubjectLocalityType" >
1960     <attribute name="IPAddress" type="string" use="optional" />
1961     <attribute name="DNSAddress" type="string" use="optional" />
1962 </complexType>
1963 <element name="AuthorityBinding" type="saml:AuthorityBindingType" />
1964 <complexType name="AuthorityBindingType" >
1965     <attribute name="AuthorityKind" type="QName" use="required" />
1966     <attribute name="Location" type="anyURI" use="required" />
1967     <attribute name="Binding" type="anyURI" use="required" />
1968 </complexType>
1969 <element name="AuthorizationDecisionStatement"
1970 type="saml:AuthorizationDecisionStatementType" />
1971 <complexType name="AuthorizationDecisionStatementType" >
1972     <complexContent>
1973         <extension base="saml:SubjectStatementAbstractType" >
1974             <sequence>
1975                 <element ref="saml:Action" maxOccurs="unbounded" />
1976                 <element ref="saml:Evidence" minOccurs="0" />
1977             </sequence>
1978             <attribute name="Resource" type="anyURI" use="required" />
1979             <attribute name="Decision" type="saml:DecisionType" use="required" />
1980         </extension>
1981     </complexContent>
1982 </complexType>
1983 <element name="Action" type="saml:ActionType" />
1984 <complexType name="ActionType" >
1985     <simpleContent>
1986         <extension base="string" >
1987             <attribute name="Namespace" type="anyURI" />
1988         </extension>
1989     </simpleContent>
1990 </complexType>
1991 <element name="Evidence" type="saml:EvidenceType" />
1992 <complexType name="EvidenceType" >
1993     <choice maxOccurs="unbounded" >
1994         <element ref="saml:AssertionIDReference" />
1995         <element ref="saml:Assertion" />
1996     </choice>
1997 </complexType>
1998 <element name="AttributeStatement" type="saml:AttributeStatementType" />
1999 <complexType name="AttributeStatementType" >
2000     <complexContent>

```

```

2001     <extension base="saml:SubjectStatementAbstractType">
2002         <sequence>
2003             <element ref="saml:Attribute" maxOccurs="unbounded"/>
2004         </sequence>
2005     </extension>
2006 </complexContent>
2007 </complexType>
2008 <element name="AttributeDesignator" type="saml:AttributeDesignatorType"/>
2009 <complexType name="AttributeDesignatorType">
2010     <attribute name="AttributeName" type="string" use="required"/>
2011     <attribute name="AttributeNamespace" type="anyURI" use="required"/>
2012 </complexType>
2013 <element name="Attribute" type="saml:AttributeType"/>
2014 <complexType name="AttributeType">
2015     <complexContent>
2016         <extension base="saml:AttributeDesignatorType">
2017             <sequence>
2018                 <element ref="saml:AttributeValue" maxOccurs="unbounded"/>
2019             </sequence>
2020         </extension>
2021     </complexContent>
2022 </complexType>
2023     <element name="AttributeValue" type="saml:anyType"/>
2024 </schema>

```

2025 8.2. Protocol Schema

2026 Following is a complete listing of the SAML protocol schema [SAML-XSD].

```

2027 <?xml version="1.0" encoding="UTF-8"?>
2028 <!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Phill Hallam-Baker
2029 (VeriSign Inc.) -->
2030 <schema
2031     targetNamespace="urn:oasis:names:tc:SAML:1.0:protocol"
2032     xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
2033     xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
2034     xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
2035     xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="unqualified">
2036     <import
2037         namespace="urn:oasis:names:tc:SAML:1.0:assertion"
2038         schemaLocation="draft-sstc-schema-assertion-30.xsd"/>
2039     <import namespace="http://www.w3.org/2000/09/xmldsig#"
2040         schemaLocation="xmldsig-core-schema.xsd"/>
2041     <annotation>
2042         <documentation>draft-sstc-schema-protocol-30.xsd</documentation>
2043     </annotation>
2044     <complexType name="RequestAbstractType" abstract="true">
2045         <sequence>
2046             <element ref="samlp:RespondWith"
2047                 minOccurs="0" maxOccurs="unbounded"/>
2048             <element ref="ds:Signature" minOccurs="0"/>
2049         </sequence>
2050         <attribute name="RequestID" type="saml:IDType" use="required"/>
2051         <attribute name="MajorVersion" type="integer" use="required"/>
2052         <attribute name="MinorVersion" type="integer" use="required"/>
2053         <attribute name="IssueInstant" type="dateTime" use="required"/>
2054     </complexType>
2055     <element name="RespondWith" type="QName"/>
2056     <element name="Request" type="samlp:RequestType"/>
2057     <complexType name="RequestType">
2058         <complexContent>
2059             <extension base="samlp:RequestAbstractType">
2060                 <choice>

```

```

2061         <element ref="sampl:Query" />
2062         <element ref="sampl:SubjectQuery" />
2063         <element ref="sampl:AuthenticationQuery" />
2064         <element ref="sampl:AttributeQuery" />
2065         <element ref="sampl:AuthorizationDecisionQuery" />
2066         <element ref="saml:AssertionID" maxOccurs="unbounded" />
2067         <element ref="sampl:AssertionArtifact" maxOccurs="unbounded" />
2068     </choice>
2069 </extension>
2070 </complexContent>
2071 </complexType>
2072 <element name="AssertionArtifact" type="string" />
2073 <element name="Query" type="sampl:QueryAbstractType" />
2074 <complexType name="QueryAbstractType" abstract="true" />
2075 <element name="SubjectQuery" type="sampl:SubjectQueryAbstractType" />
2076 <complexType name="SubjectQueryAbstractType" abstract="true">
2077     <complexContent>
2078         <extension base="sampl:QueryAbstractType">
2079             <sequence>
2080                 <element ref="saml:Subject" />
2081             </sequence>
2082         </extension>
2083     </complexContent>
2084 </complexType>
2085 <element name="AuthenticationQuery" type="sampl:AuthenticationQueryType" />
2086 <complexType name="AuthenticationQueryType">
2087     <complexContent>
2088         <extension base="sampl:SubjectQueryAbstractType">
2089             <attribute name="AuthenticationMethod" type="anyURI" />
2090         </extension>
2091     </complexContent>
2092 </complexType>
2093 <element name="AttributeQuery" type="sampl:AttributeQueryType" />
2094 <complexType name="AttributeQueryType">
2095     <complexContent>
2096         <extension base="sampl:SubjectQueryAbstractType">
2097             <sequence>
2098                 <element ref="saml:AttributeDesignator"
2099                     minOccurs="0" maxOccurs="unbounded" />
2100             </sequence>
2101             <attribute name="Resource" type="anyURI" use="optional" />
2102         </extension>
2103     </complexContent>
2104 </complexType>
2105 <element name="AuthorizationDecisionQuery"
2106     type="sampl:AuthorizationDecisionQueryType" />
2107 <complexType name="AuthorizationDecisionQueryType">
2108     <complexContent>
2109         <extension base="sampl:SubjectQueryAbstractType">
2110             <sequence>
2111                 <element ref="saml:Action" maxOccurs="unbounded" />
2112                 <element ref="saml:Evidence"
2113                     minOccurs="0" maxOccurs="unbounded" />
2114             </sequence>
2115             <attribute name="Resource" type="anyURI" use="required" />
2116         </extension>
2117     </complexContent>
2118 </complexType>
2119 <complexType name="ResponseAbstractType" abstract="true">
2120     <sequence>
2121         <element ref="ds:Signature" minOccurs="0" />
2122     </sequence>
2123     <attribute name="ResponseID" type="saml:IDType" use="required" />

```

```

2124     <attribute name="InResponseTo" type="saml:IDReferenceType"
2125         use="optional"/>
2126     <attribute name="MajorVersion" type="integer" use="required"/>
2127     <attribute name="MinorVersion" type="integer" use="required"/>
2128     <attribute name="IssueInstant" type="dateTime" use="required"/>
2129     <attribute name="Recipient" type="anyURI" use="optional"/>
2130 </complexType>
2131 <element name="Response" type="sampl:ResponseType"/>
2132 <complexType name="ResponseType">
2133     <complexContent>
2134         <extension base="sampl:ResponseAbstractType">
2135             <sequence>
2136                 <element ref="sampl:Status"/>
2137                 <element ref="saml:Assertion"
2138                     minOccurs="0" maxOccurs="unbounded"/>
2139             </sequence>
2140         </extension>
2141     </complexContent>
2142 </complexType>
2143 <element name="Status" type="sampl:StatusType"/>
2144 <complexType name="StatusType">
2145     <sequence>
2146         <element ref="sampl:StatusCode"/>
2147         <element ref="sampl:StatusMessage"
2148             minOccurs="0" maxOccurs="unbounded"/>
2149         <element ref="sampl:StatusDetail" minOccurs="0"/>
2150     </sequence>
2151 </complexType>
2152 <element name="StatusCode" type="sampl:StatusCodeType"/>
2153 <complexType name="StatusCodeType">
2154     <sequence>
2155         <element ref="sampl:StatusCode" minOccurs="0"/>
2156     </sequence>
2157     <attribute name="Value" type="QName" use="required"/>
2158 </complexType>
2159 <element name="StatusMessage" type="string"/>
2160 <element name="StatusDetail" type="sampl:StatusDetailType"/>
2161 <complexType name="StatusDetailType">
2162     <sequence>
2163         <any namespace="##any"
2164             processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
2165     </sequence>
2166 </complexType>
2167 </schema>
2168

```

9. References

2169

- 2170 [Kern-84] B. Kernighan, Rob Pike *The UNIX Programming Environment*, (March
2171 1984) Prentice Hall Computer Books;
- 2172 [Needham78] R. Needham et al., *Using Encryption for Authentication in Large Networks*
2173 *of Computers*, Communications of the ACM, Vol. 21 (12), pp. 993-999,
2174 December 1978.
- 2175 [PGP] ~~Atkins, D., Stallings, W. and P. Zimmermann, *PGP Message Exchange*
2176 *Formats, RFC 1991, August 1996.*~~
- 2177 [PKCS1] B. Kaliski, *PKCS #1: RSA Encryption Version 2.0*, RSA Laboratories, also
2178 IETF RFC 2437, October 1998. <http://www.ietf.org/rfc/rfc2437.txt>
- 2179 [PKCS7] B. Kaliski., "PKCS #7: Cryptographic Message Syntax, Version 1.5.", RFC
2180 2315, March 1998.
- 2181 [PKIX] ~~R. Housley, W. Ford, W. Polk, D. Solo. *Internet X.509 Public Key*
2182 *Infrastructure Certificate and CRL Profile. RFC 2459, January 1999.*~~
- 2183 [RFC 1510] J. Kohl, C. Neuman. *The Kerberos Network Authentication Requestor (V5)*.
2184 September 1993. <http://www.ietf.org/rfc/rfc1510.txt>
- 2185 [RFC 2045] N. Freed, N. Borenstein. *Multipurpose Internet Mail Extensions (MIME)*
2186 *Part One: Format of Internet Message Bodies*
2187 <http://www.ietf.org/rfc/rfc2045.txt> IETF RFC 2045, November 1996.
- 2188 [RFC 2104] H. Krawczyk et al., *HMAC: Keyed Hashing for Message Authentication*,
2189 <http://www.ietf.org/rfc/rfc2104.txt>, IETF RFC 2104, February 1997.
- 2190 [RFC 2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,
2191 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997
- 2192 [RFC 2246] T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. January 1999.
2193 <http://www.ietf.org/rfc/rfc2246.txt>
- 2194 [RFC 2396] T. Berners-Lee et. al., *Uniform Resource Identifiers (URI): Generic Syntax*
2195 <http://www.ietf.org/rfc/rfc2396.txt> IETF?
- 2196 [RFC 2630] R. Housley. *Cryptographic Message Syntax*. June 1999.
2197 <http://www.ietf.org/rfc/rfc630.txt>
- 2198 [RFC 2648] R. Moats. *A URN Namespace for IETF Documents*. August 1999.
2199 <http://www.ietf.org/rfc/rfc2648.txt>
- 2200 [RFC 3075] D. Eastlake, J. Reagle, D. Solo. *XML-Signature Syntax and Processing*.
2201 March 2001. <http://www.ietf.org/rfc/rfc3075.txt>
- 2202 [SAMLBind] P. Mishra et al., *Bindings and Profiles for the OASIS Security Assertion*
2203 *Markup Language (SAML)*, [http://www.oasis-](http://www.oasis-open.org/committees/security/docs/draft-sstc-bindings-model-07.pdf)
2204 [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),
2205 OASIS, December 2001.
- 2206 [SAMLConform] **TBS**
- 2207 [SAMLGloss] J. Hodges et al., *Glossary for the OASIS Security Assertion Markup*
2208 *Language (SAML)*, [http://www.oasis-](http://www.oasis-open.org/committees/security/docs/draft-sstc-glossary-02.pdf)
2209 [open.org/committees/security/docs/draft-sstc-glossary-02.pdf](http://www.oasis-open.org/committees/security/docs/draft-sstc-glossary-02.pdf), OASIS,
2210 December 2001.
- 2211 [SAMLXSD] 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)
2212 [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),
2213 OASIS, December 2001.
- 2214 [SAMLSecure] **TBS**

2215	[SAML-XSD]	P. Hallam-Baker et al., <i>SAML assertion schema</i> , http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-assertion-21.xsd , OASIS, December 2001.
2216		
2217		
2218	[Schema1]	H. S. Thompson et al., <i>XML Schema Part 1: Structures</i> , http://www.w3.org/TR/xmlschema-1/ , World Wide Web Consortium Recommendation, May 2001.
2219		
2220		
2221	[Schema2]	P. V. Biron et al., <i>XML Schema Part 2: Datatypes</i> , http://www.w3.org/TR/xmlschema-2 , World Wide Web Consortium Recommendation, May 2001.
2222		
2223		
2224	[SPKI]	<u>C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, T. Ylonen. <i>SPKI Certificate Theory</i>, RFC 2693, September 1999.</u>
2225		
2226	[UNICODE-C]	M. Davis, M. J. Dürst, http://www.unicode.org/unicode/reports/tr15/tr15-21.html , UNICODE Consortium
2227		
2228	[W3C-CHAR]	M. J. Dürst, <i>Requirements for String Identity Matching and String Indexing</i> http://www.w3.org/TR/WD-charreq , World Wide Web Consortium.
2229		
2230	[W3C-CharMod]	M. J. Dürst,, <i>Unicode Normalization Forms</i> http://www.w3.org/TR/charmod/ , World Wide Web Consortium.
2231		
2232	[X.500]	ITU-T Recommendation X.501: <i>Information Technology - Open Systems Interconnection - The Directory: Models</i> , 1993.
2233		
2234	[XKMS]	W. Ford, P. Hallam-Baker, B. Fox, B. Dillaway, B. LaMacchia, J. Epstein, J. Lapp, XML Key Management Specification (XKMS), W3C Note 30 March 2001, http://www.w3.org/TR/xkms/
2235		
2236		
2237	[XML]	T. Bray et. al. <i>Extensible Markup Language (XML) 1.0 (Second Edition)</i> , http://www.w3.org/TR/REC-xml , World Wide Web Consortium.
2238		
2239	[XMLEnc]	<i>XML Encryption Specification</i> , In development.
2240	[XMLSig]	D. Eastlake et al., <i>XML-Signature Syntax and Processing</i> , http://www.w3.org/TR/xmlsig-core/ , World Wide Web Consortium.
2241		
2242	[XMLSig-XSD]	XML Signature Schema available from http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd .
2243		
2244	[XTAML]	P. Hallam-Baker, <i>XML Trust Axiom Markup Language 1.0</i> , http://www.xmltrustcenter.org/ , VeriSign Inc. September 2001.
2245		
2246		

2247

10. Acknowledgements

2248 The editors would like to acknowledge the contributions of the OASIS SAML Technical Committee,
2249 whose voting members at the time of publication were:

2250 Allen Rogers, Authentica
2251 Irving Reid, Baltimore Technologies
2252 Krishna Sankar, Cisco Systems Inc
2253 Simon Godik, Crosslogix
2254 Gil Pilz, E2open
2255 Hal Lockhart, Entegriety Solutions
2256 Carlisle Adams, Entrust [Inc. Technologies](#)
2257 Robert Griffin, Entrust [Inc. Technologies](#)
2258 Don Flinn, Hitachi
2259 Joe Pato, Hewlett-Packard ([co-Chair](#))
2260 Jason Rouault, Hewlett-Packard
2261 Marc Chanliau, Netegrity
2262 Chris McLaren, Netegrity
2263 Prateek Mishra, Netegrity
2264 Charles Knouse, Oblix
2265 Steve Anderson, OpenNetwork
2266 Rob Philpott, RSA Security
2267 Jahan Moreh, Sigaba
2268 Bhavna Bhatnagar, Sun Microsystems
2269 Jeff Hodges, Sun Microsystems ([co-Chair](#))
2270 Eve Maler, Sun Microsystems ([Former Chair](#))
2271 Aravindan Ranganathan, Sun Microsystems
2272 Emily Xu, Sun Microsystems
2273 Bob Morgan, University of Washington
2274 Phillip Hallam-Baker, VeriSign Inc.

2275

2276 The editors would also like to thank the following people for their contributions:

2277 Stephen Farrell, Baltimore Technologies
2278 David Orchard, BEA [Systems](#)
2279 Tim Moses, Entrust [Inc.](#)
2280 Nigel Edwards, Hewlett-Packard
2281 Marc Chanliau, Netegrity
2282 Scott Cantor, The Ohio State University
2283 Darren Platt, Formerly with RSA Security
2284 Bob Blakeley [IBM Tivoli Software](#) ([Former Chair](#))
2285 Marlena Erdos, Tivoli

2286

2287

2288

2289

2290

2291

2292

2293

2294

2295

2296

2297

2298

2299
2300

2301

Appendix A. Notices

2302 OASIS takes no position regarding the validity or scope of any intellectual property or other rights
2303 that might be claimed to pertain to the implementation or use of the technology described in this
2304 document or the extent to which any license under such rights might or might not be available;
2305 neither does it represent that it has made any effort to identify any such rights. Information on
2306 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS
2307 website. Copies of claims of rights made available for publication and any assurances of licenses to
2308 be made available, or the result of an attempt made to obtain a general license or permission for
2309 the use of such proprietary rights by implementors or users of this specification, can be obtained
2310 from the OASIS Executive Director.

2311 OASIS invites any interested party to bring to its attention any copyrights, patents or patent
2312 applications, or other proprietary rights which may cover technology that may be required to
2313 implement this specification. Please address the information to the OASIS Executive Director.

2314 Copyright © The Organization for the Advancement of Structured Information Standards [OASIS]
2315 2001. All Rights Reserved.

2316 This document and translations of it may be copied and furnished to others, and derivative works
2317 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,
2318 published and distributed, in whole or in part, without restriction of any kind, provided that the above
2319 copyright notice and this paragraph are included on all such copies and derivative works. However,
2320 this document itself may not be modified in any way, such as by removing the copyright notice or
2321 references to OASIS, except as needed for the purpose of developing OASIS specifications, in
2322 which case the procedures for copyrights defined in the OASIS Intellectual Property Rights
2323 document must be followed, or as required to translate it into languages other than English.

2324 The limited permissions granted above are perpetual and will not be revoked by OASIS or its
2325 successors or assigns.

2326 This document and the information contained herein is provided on an "AS IS" basis and OASIS
2327 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO
2328 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY
2329 RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A
2330 PARTICULAR PURPOSE.