



1
2
3
4
5
6
7
8
9
10
11
12
13
14

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

Document identifier: draft-sstc-core-29

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

Publication date: March 29th 2002

Maturity Level: Committee Working Draft

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

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

Editors:

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

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
43
44
45
46
47
48
49
50
51

ASSERTIONS AND PROTOCOL FOR THE OASIS SECURITY ASSERTION MARKUP LANGUAGE (SAML)	1
1. INTRODUCTION	5
1.1. NOTATION	5
1.2. SCHEMA ORGANIZATION AND NAMESPACES	5
1.2.1. <i>String and URI Values</i>	6
1.2.2. <i>Time Values.</i>	6
1.2.3. <i>Comparing SAML values</i>	6
1.3. SAML CONCEPTS (NON-NORMATIVE)	6
1.3.1. <i>Overview</i>	7
1.3.2. <i>SAML and URI-Based Identifiers</i>	8
1.3.3. <i>SAML and Extensibility</i>	8
2. SAML ASSERTIONS	9
2.1. SCHEMA HEADER AND NAMESPACE DECLARATIONS	9
2.2. SIMPLE TYPES	9
2.2.1. <i>Simple Types IDType and IDReferenceType</i>	9
2.2.2. <i>Simple Type DecisionType</i>	10
2.3. ASSERTIONS	10
2.3.1. <i>Element <AssertionID></i>	10
2.3.2. <i>Element <Assertion></i>	10
2.3.2.1. <i>Element <Conditions></i>	12
2.3.2.1.1. <i>Attributes NotBefore and NotOnOrAfter</i>	13
2.3.2.1.2. <i>Element <Condition></i>	13
2.3.2.1.3. <i>Elements <AudienceRestrictionCondition> and <Audience></i>	13
2.3.2.2. <i>Elements <Advice> and <AdviceElement></i>	14
2.4. STATEMENTS	14
2.4.1. <i>Element <Statement></i>	14
2.4.2. <i>Element <SubjectStatement></i>	14
2.4.2.1. <i>Element <Subject></i>	15
2.4.2.2. <i>Element <NameIdentifier></i>	15
2.4.2.3. <i>Elements <SubjectConfirmation>, <ConfirmationMethod>, and <SubjectConfirmationData></i>	16
2.4.3. <i>Element <AuthenticationStatement></i>	17
2.4.3.1. <i>Element <SubjectLocality></i>	17
2.4.3.2. <i>Element <AuthorityBinding></i>	18
2.4.4. <i>Element <AuthorizationDecisionStatement></i>	18
2.4.4.1. <i>Element <Action></i>	20

Deleted: March 29th 2002

52	2.4.4.2. Element <Evidence>	20
53	2.4.5. Element <AttributeStatement>	20
54	2.4.5.1. Elements <AttributeDesignator> and <Attribute>	21
55	2.4.5.1.1 Element <AttributeValue>	21
56	3. SAML PROTOCOL	23
57	3.1. SCHEMA HEADER AND NAMESPACE DECLARATIONS	23
58	3.2. REQUESTS	23
59	3.2.1. Complex Type RequestAbstractType	23
60	3.2.1.1. Element <RespondWith>	24
61	3.2.2. Element <Request>	24
62	3.2.3. Element <AssertionArtifact>	25
63	3.3. QUERIES	25
64	3.3.1. Element <Query>	25
65	3.3.2. Element <SubjectQuery>	26
66	3.3.3. Element <AuthenticationQuery>	26
67	3.3.4. Element <AttributeQuery>	27
68	3.3.5. Element <AuthorizationDecisionQuery>	27
69	3.4. RESPONSES	28
70	3.4.1. Complex Type ResponseAbstractType	28
71	3.4.2. Element <Response>	29
72	3.4.3. Element <Status>	29
73	3.4.3.1. Element <StatusCode>	30
74	3.4.3.2. Element <StatusMessage>	31
75	3.4.3.3. Element <StatusDetail>	31
76	3.4.4. Responses to <AuthenticationQuery> and <AttributeQuery>	31
77	4. SAML VERSIONING	33
78	4.1. ASSERTION VERSION	33
79	4.2. REQUEST VERSION	33
80	4.3. RESPONSE VERSION	34
81	5. SAML & XML-SIGNATURE SYNTAX AND PROCESSING	35
82	5.1. SIGNING ASSERTIONS	35
83	5.2. REQUEST/RESPONSE SIGNING	36
84	5.3. SIGNATURE INHERITANCE	36
85	5.3.1. Rationale	36
86	5.3.2. Rules for SAML Signature Inheritance	36
87	5.4. XML SIGNATURE PROFILE	36

Deleted: March 29th 2002

88	5.4.1. Signing formats	36
89	5.4.2. CanonicalizationMethod	36
90	5.4.3. Transforms	37
91	5.4.4. KeyInfo	37
92	5.4.5. Binding between statements in a multi-statement assertion	37
93	6. SAML EXTENSIONS	38
94	6.1. ASSERTION SCHEMA EXTENSION	38
95	6.2. PROTOCOL SCHEMA EXTENSION	38
96	6.3. USE OF TYPE DERIVATION AND SUBSTITUTION GROUPS	39
97	7. SAML-DEFINED IDENTIFIERS	40
98	7.1. AUTHENTICATION METHOD AND CONFIRMATION METHOD IDENTIFIERS	40
99	7.1.1. SAML Artifact (SHA-1):	40
100	7.1.2. Holder of Key:	41
101	7.1.3. Bearer Indication:	41
102	7.1.4. Sender Vouches:	41
103	7.1.5. Password (Pass-Through):	41
104	7.1.6. Password (One-Way-Function SHA-1):	41
105	7.1.7. Kerberos	41
106	7.1.8. SSL/TLS Certificate Based Client Authentication:	41
107	7.1.9. Object Authenticator (SHA-1):	42
108	7.1.10. PKCS#7	42
109	7.1.11. Cryptographic Message Syntax	42
110	7.1.12. XML Digital Signature	42
111	7.2. ACTION NAMESPACE IDENTIFIERS	42
112	7.2.1. Read/Write/Execute/Delete/Control:	42
113	7.2.2. Read/Write/Execute/Delete/Control with Negation:	43
114	7.2.3. Get/Head/Put/Post:	43
115	7.2.4. UNIX File Permissions:	43
116	8. SAML SCHEMA LISTINGS	45
117	8.1. ASSERTION SCHEMA	45
118	8.2. PROTOCOL SCHEMA	48
119	9. REFERENCES	51
120	10. ACKNOWLEDGEMENTS	53
121	APPENDIX A. NOTICES	55
122		

Deleted: ASSERTIONS AND PROTOCOL FOR THE OASIS SECURITY ASSERTION MARKUP LANGUAGE (SAML) 1¶

1. INTRODUCTION 6¶

1.1. NOTATION 6¶

1.2. SCHEMA ORGANIZATION AND NAMESPACES 6¶

1.2.1. Time Values. 7¶

1.2.2. Comparing SAML values 7¶

1.3. SAML CONCEPTS (NON-NORMATIVE) 7¶

1.3.1. Overview 7¶

1.3.2. SAML and URI-Based Identifiers 9¶

1.3.3. SAML and Extensibility 9¶

2. SAML ASSERTIONS 10¶

2.1. SCHEMA HEADER AND NAMESPACE DECLARATIONS 10¶

2.2. SIMPLE TYPES 10¶

2.2.1. Simple Types IDType and IDReferenceType 10¶

2.2.2. Simple Type DecisionType 11¶

2.3. ASSERTIONS 11¶

2.3.1. Element <AssertionID> 11¶

2.3.2. Element <Assertion> 11¶

2.3.2.1. Element <Conditions> 13¶

2.3.2.1.1. Attributes NotBefore and NotOnOrAfter 13¶

2.3.2.1.2. Element <Condition> 14¶

2.3.2.1.3. Elements <AudienceRestrictionCondition> and <Audience> 14¶

2.3.2.2. Elements <Advice> and <AdviceElement> 15¶

2.4. STATEMENTS 15¶

2.4.1. Element <Statement> 15¶

2.4.2. Element <SubjectStatement> 15¶

2.4.2.1. Element <Subject> 16¶

2.4.2.2. Element <NameIdentifier> 16¶

2.4.2.3. Elements <SubjectConfirmation>, <ConfirmationMethod>, and <SubjectConfirmationData> 17¶

2.4.3. Element <AuthenticationStatement> 18¶

2.4.3.1. Element <AuthenticationLocality> 18¶

2.4.3.2. Element <AuthorityBinding> 19¶

2.4.4. Element <AuthorizationDecisionStatement> 19¶

2.4.4.1. Element <Action> 20¶

2.4.4.2. Element <Evidence> 21¶

2.4.5. Element <AttributeStatement> 21¶

2.4.5.1. Elements <AttributeDesignator> and <Attribute> 22¶

2.4.5.1.1. Element <AttributeValue> 22¶

3. SAML PROTOCOL 23¶

3.1. SCHEMA HEADER AND NAMESPACE DECLARATIONS 23¶

3.2. REQUESTS 23¶

Deleted: 55
Deleted: ¶
Inserted: 56
Deleted: March 29th 2002

1. Introduction

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

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

1.1. Notation

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

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

"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)"

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

Listings of SAML schemas appear like this.

Example code listings appear like this.

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

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

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

1.2. Schema Organization and Namespaces

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

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

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

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

Deleted: <http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-assertion-29.xsd>

Deleted: <http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-protocol-29.xsd>

Deleted: March 29th 2002

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

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

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

169 **1.2.1. String and URI Values**

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

175 **1.2.2. Time Values.**

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

178 SAML Requestors and Responders SHOULD NOT rely on other applications supporting time
179 resolution finer than milliseconds. Implementations MUST NOT generate time instants that specify
180 leap seconds.

Deleted: application

181 **1.2.3. Comparing SAML values**

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

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

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

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

203 **1.3. SAML Concepts (Non-Normative)**

204 This section is informative only and is superseded by any contradicting information in the normative
205 text in Section [2](#), and following. A glossary of SAML terms and concepts [**SAMLGloss**] is available.

Deleted: s 1.2

Deleted: March 29th 2002

206 **1.3.1. Overview**

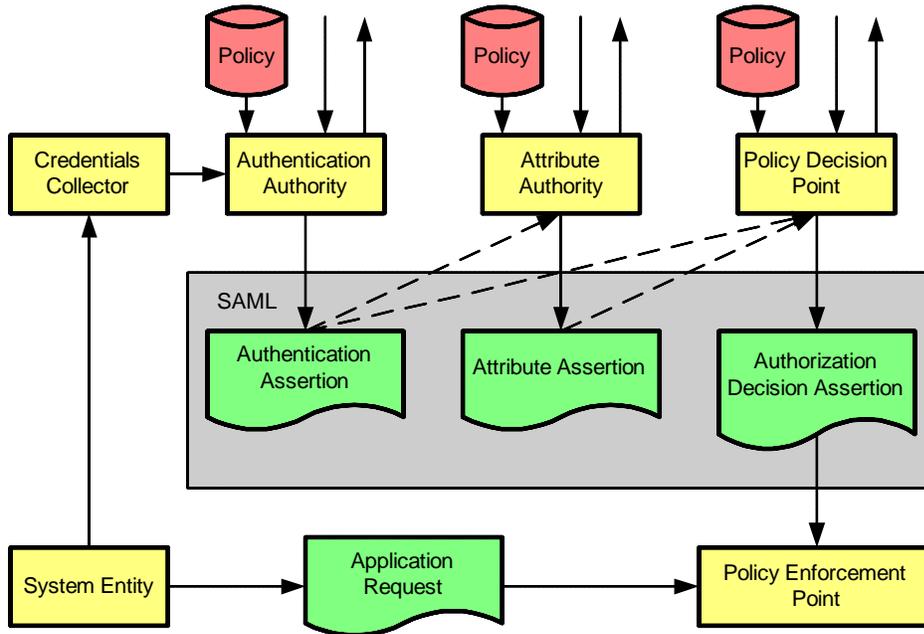
207 The Security Assertion Markup Language (SAML) is an XML-based framework for exchanging
208 security information. This security information is expressed in the form of assertions about subjects,
209 where a subject is an entity (either human or computer) that has an identity in some security
210 domain.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

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

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

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

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



228

229

Figure 1 The SAML Domain Model

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

Deleted: March 29th 2002

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

234 The assertion and protocol data formats are defined in this specification. The bindings and profiles
235 are defined in a separate specification [**SAMLSBind**]. A conformance program for SAML is defined
236 in the conformance specification [**SAMLSConform**]. Security issues are discussed in a separate
237 security and privacy considerations specification [**SAMLSecure**].

238 1.3.2. SAML and URI-Based Identifiers

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

242 **urn:ietf:rfc:1510**

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

245 **urn:oasis:names:tc:SAML:1.0:action:rwdc**

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

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

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

257 1.3.3. SAML and Extensibility

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

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

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

Deleted: #

262

2. SAML Assertions

263

An assertion is a package of information that supplies one or more statements made by an issuer.

264

SAML allows issuers to make three different kinds of assertion statement:

265

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

266

267

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

268

269

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

270

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

271

272

273

2.1. Schema Header and Namespace Declarations

274

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

275

276

```
<schema
  targetNamespace="urn:oasis:names:tc:SAML:1.0:assertion"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
  xmlns="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="unqualified">
  <import namespace="http://www.w3.org/2000/09/xmldsig#"
    schemaLocation="xmldsig-core-schema.xsd" />
  <annotation>
    <documentation>draft-sstc-schema-assertion-29.xsd</documentation>
  </annotation>
  ...
</schema>
```

277

278

279

280

281

282

283

284

285

286

287

288

Comment: Page: 1
Update with final name spaces

Deleted: http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-assertion-29.xsd

Deleted: http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-assertion-29.xsd

Comment: schema location bug

289

2.2. Simple Types

290

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

291

2.2.1. Simple Types IDType and IDReferenceType

292

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

293

294

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

295

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

296

297

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

298

299

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

300

301

302

303

Deleted: application

Deleted:

Deleted: March 29th 2002

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

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

```
308 <simpleType name="IDType">  
309   <restriction base="string"/>  
310 </simpleType>  
311 <simpleType name="IDReferenceType">  
312   <restriction base="string"/>  
313 </simpleType>
```

314 2.2.2. Simple Type DecisionType

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

317 Permit

318 The specified action is permitted.

319 Deny

320 The specified action is denied.

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

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

326

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

```
328 <simpleType name="DecisionType">  
329   <restriction base="string">  
330     <enumeration value="Permit"/>  
331     <enumeration value="Deny"/>  
332     <enumeration value="Indeterminate"/>  
333   </restriction>  
334 </simpleType>
```

335 2.3. Assertions

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

337 2.3.1. Element <AssertionID>

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

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

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

342 2.3.2. Element <Assertion>

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

345 `MajorVersion` [Required]

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

Deleted: March 29th 2002

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

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

354 **Issuer** [Required]
 355 The issuer of the assertion. The name of the issuer is provided as a string. The issuer
 356 name SHOULD be unambiguous to the intended relying parties. SAML authorities may use
 357 an identifier such as a URI reference that is designed to be unambiguous regardless of
 358 context.

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

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

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

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

368 One or more of the following statement elements:

369 **<Statement>**
 370 A statement defined in an extension schema.

371 **<SubjectStatement>**
 372 A subject statement defined in an extension schema.

373 **<AuthenticationStatement>**
 374 An authentication statement.

375 **<AuthorizationDecisionStatement>**
 376 An authorization decision statement.

377 **<AttributeStatement>**
 378 An attribute statement.

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

```

381 <element name="Assertion" type="saml:AssertionType"/>
382 <complexType name="AssertionType">
383   <sequence>
384     <element ref="saml:Conditions" minOccurs="0"/>
385     <element ref="saml:Advice" minOccurs="0"/>
386     <choice maxOccurs="unbounded">
387       <element ref="saml:Statement"/>
388       <element ref="saml:SubjectStatement"/>
389       <element ref="saml:AuthenticationStatement"/>
390       <element ref="saml:AuthorizationDecisionStatement"/>
391       <element ref="saml:AttributeStatement"/>
392     </choice>
393     <element ref="ds:Signature" minOccurs="0"/>
394   </sequence>
395   <attribute name="MajorVersion" type="integer" use="required"/>
396   <attribute name="MinorVersion" type="integer" use="required"/>
397   <attribute name="AssertionID" type="saml:IDType" use="required"/>

```

Deleted: pplications

Deleted: March 29th 2002

```

398     <attribute name="Issuer" type="string" use="required"/>
399     <attribute name="IssueInstant" type="dateTime" use="required"/>
400 </complexType>

```

401 2.3.2.1. Element <Conditions>

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

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

405 NotBefore [Optional]

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

408 NotOnOrAfter [Optional]

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

411 <Condition> [Any Number]

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

413 <AudienceRestrictionCondition> [Any Number]

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

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

```

417     <element name="Conditions" type="saml:ConditionsType"/>
418     <complexType name="ConditionsType">
419       <choice minOccurs="0" maxOccurs="unbounded">
420         <element ref="saml:AudienceRestrictionCondition"/>
421         <element ref="saml:Condition"/>
422       </choice>
423       <attribute name="NotBefore" type="dateTime" use="optional"/>
424       <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
425     </complexType>

```

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

- 430 1. If no sub-elements or attributes are supplied in the <Conditions> element, then the
431 assertion is considered to be **Valid**.
- 432 2. If any sub-element or attribute of the <Conditions> element is determined to be invalid,
433 then the assertion is **Invalid**.
- 434 3. If any sub-element or attribute of the <Conditions> element cannot be evaluated, then
435 the validity of the assertion cannot be determined and is deemed to be **Indeterminate**.
- 436 4. If all sub-elements and attributes of the <Conditions> element are determined to be
437 **Valid**, then the assertion is considered to be **Valid**.

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

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

Deleted: The validity status of an assertion is the conjunction of the validity status of each of the conditions it contains, as follows:¶
 <#>If any condition evaluates to Invalid, the assertion status is Invalid.¶
 <#>If no condition evaluates to Invalid and one or more conditions evaluate to Indeterminate, the assertion status is Indeterminate.¶
 <#>If no conditions are supplied or all the specified conditions evaluate to Valid, the assertion status is Valid.¶
 Note that an assertion that has validity status 'Valid' may not be trustworthy by reasons such as not being issued by a trustworthy issuer or not being authenticated by a trustworthy signature.¶
 The <Conditions> element MAY be extended to contain additional conditions. If an element contained within a <Conditions> element is encountered that is not understood, the status of the condition MUST be evaluated to **Indeterminate**.¶

Deleted: <element ref="saml:AudienceRestrictionCondition"/>¶

Deleted: March 29th 2002

Deleted: ¶

444 2.3.2.1.1 *Attributes NotBefore and NotOnOrAfter*

445 The ~~NotBefore and NotOnOrAfter~~ attributes specify time limits on the validity of the assertion.

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

448 If the value for either `NotBefore` or `NotOnOrAfter` is omitted it is considered unspecified. If the
449 `NotBefore` attribute is unspecified (and if any other conditions that are supplied evaluate to
450 `Valid`), the assertion is valid at any time before the time instant specified by the `NotOnOrAfter`
451 attribute. If the `NotOnOrAfter` attribute is unspecified (and if any other conditions that are supplied
452 evaluate to `Valid`), the assertion is valid from the time instant specified by the `NotBefore`
453 attribute with no expiry. If neither attribute is specified (and if any other conditions that are supplied
454 evaluate to `Valid`), the assertion is valid at any time.

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

459 2.3.2.1.2 *Element <Condition>*

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

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

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

467 2.3.2.1.3 *Elements <AudienceRestrictionCondition> and <Audience>*

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

```
473 <Audience>  
474     A URI reference that identifies an intended audience. The URI reference MAY identify a  
475     document that describes the terms and conditions of audience membership.
```

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

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

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

```
486 <element name="AudienceRestrictionCondition"  
487     type="saml:AudienceRestrictionConditionType"/>  
488 <complexType name="AudienceRestrictionConditionType">  
489     <complexContent>  
490     <extension base="saml:ConditionAbstractType">
```

Deleted: March 29th 2002

```

491     <sequence>
492       <element ref="saml:Audience" maxOccurs="unbounded" />
493     </sequence>
494   </extension>
495 </complexContent>
496 </complexType>
497 <element name="Audience" type="anyURI" />

```

2.3.2.2. Elements <Advice> and <AdviceElement>

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

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

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

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

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

```

511 <element name="Advice" type="saml:AdviceType" />
512 <complexType name="AdviceType">
513   <choice minOccurs="0" maxOccurs="unbounded">
514     <element ref="saml:AssertionIDReference" />
515     <element ref="saml:Assertion" />
516     <any namespace="##other" processContents="lax" />
517   </choice>
518 </complexType>

```

Deleted: , <AdviceElement> elements

Deleted: , along with the <AdviceElement> element and its **AdviceAbstractType** complex type

Deleted: <element ref="saml:AdviceElement" />

Deleted: <element name="AdviceElement" type="saml:AdviceAbstractType" />
<complexType name="AdviceAbstractType" />

2.4. Statements

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

2.4.1. Element <Statement>

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

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

```

527 <element name="Statement" type="saml:StatementAbstractType" />
528 <complexType name="StatementAbstractType" abstract="true" />

```

2.4.2. Element <SubjectStatement>

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

Deleted: March 29th 2002

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

```
537 <element name="SubjectStatement" type="saml:SubjectStatementAbstractType" />  
538 <complexType name="SubjectStatementAbstractType" abstract="true">  
539   <complexContent>  
540     <extension base="saml:StatementAbstractType">  
541       <sequence>  
542         <element ref="saml:Subject" />  
543       </sequence>  
544     </extension>  
545   </complexContent>  
546 </complexType>
```

547 2.4.2.1. Element <Subject>

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

550 <NameIdentifier>

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

552 <SubjectConfirmation>

553 Information that allows the subject to be authenticated.

554 If the <Subject> element contains both a <NameIdentifier> and a

555 <SubjectConfirmation>, the issuer is asserting that if the relying party performs the specified
556 <SubjectConfirmation>, it can be confident that the entity presenting the assertion to the
557 relying party is the entity that the issuer associates with the <NameIdentifier>. A <Subject>
558 element SHOULD NOT identify more than one principal.

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

```
561 <element name="Subject" type="saml:SubjectType" />  
562 <complexType name="SubjectType">  
563   <choice>  
564     <sequence>  
565       <element ref="saml:NameIdentifier" />  
566       <element ref="saml:SubjectConfirmation" minOccurs="0" />  
567     </sequence>  
568     <element ref="saml:SubjectConfirmation" />  
569   </choice>  
570 </complexType>
```

571 2.4.2.2. Element <NameIdentifier>

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

574 NameQualifier [Optional]

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

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

578 Format [Optional]

579 The syntax used to describe the name of the subject

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

583 #emailAddress

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

Deleted: March 29th 2002

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

589 #X509SubjectName
590 Indicates that the content of the NameIdentifier element is in the form specified for
591 the contents of <ds:X509SubjectName> element in [DSIG]. Implementors should note that
592 [DSIG] specifies encoding rules for X.509 subject names that differ from the rules given in
593 RFC2253 [RFC2253].

594 #WindowsDomainQualifiedName
595 Indicates that the content of the NameIdentifier element is a Windows domain qualified
596 name. A Windows domain qualified user name is a string of the form
597 "DomainName\UserName". The domain name and "\" separator may be omitted.

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

```
600 <element name="NameIdentifier" type="saml:NameIdentifierType"/>  
601 <complexType name="NameIdentifierType">  
602   <simpleContent>  
603     <extension base="string">  
604       <attribute name="NameQualifier" type="string" use="optional"/>  
605       <attribute name="Format" type="anyURI" use="optional"/>  
606     </extension>  
607   </simpleContent>  
608 </complexType>
```

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

611 Regardless of format, issues of anonymity, pseudonymity, and the persistence of
612 the identifier with respect to the asserting and relying parties, are also
613 implementation-specific.

614 2.4.2.3. Elements <SubjectConfirmation>, <ConfirmationMethod>, and 615 <SubjectConfirmationData>

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

618 <ConfirmationMethod> [One or more]
619 A URI reference that identifies a protocol to be used to authenticate the subject. URI
620 references identifying common authentication protocols are listed in Section 7.

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

623 <ds:KeyInfo> [Optional]
624 An XML Signature [XMLSig] element that specifies a cryptographic key held by the
625 subject.

626 The following schema fragment defines the <SubjectConfirmation> element and its
627 **SubjectConfirmationType** complex type, along with the <SubjectConfirmationData>
628 element and the <ConfirmationMethod> element:

```
629 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>  
630 <complexType name="SubjectConfirmationType">  
631   <sequence>  
632     <element ref="saml:ConfirmationMethod" maxOccurs="unbounded"/>  
633     <element ref="saml:SubjectConfirmationData" minOccurs="0"/>  
634     <element ref="ds:KeyInfo" minOccurs="0"/>  
635   </sequence>
```

Deleted: March 29th 2002

```
636 </complexType>
637 <element name="SubjectConfirmationData" type="string"/>
638 <element name="ConfirmationMethod" type="anyURI"/>
```

639 2.4.3. Element <AuthenticationStatement>

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

644 AuthenticationMethod [Optional]

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

647 AuthenticationInstant [Optional]

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

650 <SubjectLocality> [Optional]

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

Deleted: AuthenticationLocality

653 <AuthorityBinding> [Any Number]

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

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

```
657 <element name="AuthenticationStatement"
658 type="saml:AuthenticationStatementType"/>
659 <complexType name="AuthenticationStatementType">
660 <complexContent>
661 <extension base="saml:SubjectStatementAbstractType">
662 <sequence>
663 <element ref="saml:SubjectLocality" minOccurs="0"/>
664 <element ref="saml:AuthorityBinding"
665 minOccurs="0" maxOccurs="unbounded"/>
666 </sequence>
667 <attribute name="AuthenticationMethod" type="anyURI"/>
668 <attribute name="AuthenticationInstant" type="dateTime"/>
669 </extension>
670 </complexContent>
671 </complexType>
```

Deleted: Authentication

Deleted: Locality

Deleted: AuthenticationLocality

672 2.4.3.1. Element <SubjectLocality>

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

Deleted: AuthenticationLocality

675 IPAddress [Optional]

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

677 DNSAddress [Optional]

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

679 This element is entirely advisory, since both these fields are quite easily "spoofed" but current
680 practice appears to require its inclusion.

Deleted: AuthenticationLocality

Deleted: AuthenticationLocalityType

681 The following schema fragment defines the <SubjectLocality> element and its

682 **SubjectLocalityType** complex type:

```
683 <element name="SubjectLocality"
```

Deleted: AuthenticationLocality

Deleted: March 29th 2002

```

684     type="saml: SubjectLocalityType"/>
685 <complexType name="SubjectLocalityType">
686   <attribute name="IPAddress" type="string" use="optional"/>
687   <attribute name="DNSAddress" type="string" use="optional"/>
688 </complexType>

```

Deleted: AuthenticationLocality

Deleted: AuthenticationLocality

689 2.4.3.2. Element <AuthorityBinding>

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

695 AuthorityKind [Required]

696 The type of SAML Protocol queries to which the authority described by this element will
697 respond. The value is specified as an XML Schema QName. The acceptable values for
698 AuthorityKind are the namespace-qualified names of element types or elements
699 derived from the SAML Protocol Query element (see Section 3.3). For example, an
700 attribute authority would be identified by AuthorityKind="samlp:AttributeQuery".
701 For extension schemas, where the actual type of the samlp:Query would be identified by
702 an xsi:type attribute, the value of AuthorityKind MUST be the same as the value of
703 the xsi:type attribute for the corresponding query.

704 Location [Required]

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

708 Binding [Required]

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

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

```

713 <element name="AuthorityBinding" type="saml:AuthorityBindingType"/>
714 <complexType name="AuthorityBindingType">
715   <attribute name="AuthorityKind" type="QName" use="required"/>
716   <attribute name="Location" type="anyURI" use="required"/>
717   <attribute name="Binding" type="anyURI" use="required"/>
718 </complexType>

```

719 2.4.4. Element <AuthorizationDecisionStatement>

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

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

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

Deleted: March 29th 2002

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

Deleted: applications

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

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

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

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

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

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

751 Resource [Required]

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

757 Decision [Required]

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

760 <Action> [One or more]

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

762 <Evidence> [Any Number]

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

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

```
766 <element name="AuthorizationDecisionStatement"  
767 type="saml:AuthorizationDecisionStatementType" />  
768 <complexType name="AuthorizationDecisionStatementType">  
769 <complexContent>  
770 <extension base="saml:SubjectStatementAbstractType">  
771 <sequence>  
772 <element ref="saml:Action" maxOccurs="unbounded" />  
773 <element ref="saml:Evidence" minOccurs="0" />  
774 </sequence>  
775 <attribute name="Resource" type="anyURI" use="required" />  
776 <attribute name="Decision" type="saml:DecisionType" use="required" />  
777 </extension>  
778 </complexContent>  
779 </complexType>
```

Deleted: March 29th 2002

780 2.4.4.1. Element <Action>

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

783 Namespace [Optional]

784 A URI reference representing the namespace in which the name of the specified action is
785 to be interpreted. If this element is absent, the namespace
786 [urn:oasis:names:tc:SAML:1.0:action:rwdc-negation](http://www.oasis-open.org/committees/security/docs/draft-sstc-core-28#rwdc-negation), specified in section 7.2.2 is in effect.

787 string data [Required]

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

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

```
790 <element name="Action" type="saml:ActionType"/>  
791 <complexType name="ActionType">  
792   <simpleContent>  
793     <extension base="string">  
794       <attribute name="Namespace" type="anyURI"/>  
795     </extension>  
796   </simpleContent>  
797 </complexType>
```

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

798 2.4.4.2. Element <Evidence>

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

802 <AssertionIDReference>

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

804 <Assertion>

805 Specifies an assertion by value.

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

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

```
813 <element name="Evidence" type="saml:EvidenceType"/>  
814 <complexType name="EvidenceType">  
815   <choice maxOccurs="unbounded">  
816     <element ref="saml:AssertionIDReference"/>  
817     <element ref="saml:Assertion"/>  
818   </choice>  
819 </complexType>
```

820 2.4.5. Element <AttributeStatement>

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

824 <Attribute> [One or More]

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

Deleted: March 29th 2002

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

```
828 <element name="AttributeStatement" type="saml:AttributeStatementType" />
829 <complexType name="AttributeStatementType">
830   <complexContent>
831     <extension base="saml:SubjectStatementAbstractType">
832       <sequence>
833         <element ref="saml:Attribute" maxOccurs="unbounded" />
834       </sequence>
835     </extension>
836   </complexContent>
837 </complexType>
```

838 2.4.5.1. Elements <AttributeDesignator> and <Attribute>

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

843 AttributeNamespace [Optional]

844 The namespace in which the AttributeName elements are interpreted.

845 AttributeName [Optional]

846 The name of the attribute.

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

```
849 <element name="AttributeDesignator" type="saml:AttributeDesignatorType" />
850 <complexType name="AttributeDesignatorType">
851   <attribute name="AttributeName" type="string" use="required" />
852   <attribute name="AttributeNamespace" type="anyURI" use="required" />
853 </complexType>
```

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

857 <AttributeValue> [Any Number]

858 The value of the attribute.

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

```
861 <element name="Attribute" type="saml:AttributeType" />
862 <complexType name="AttributeType">
863   <complexContent>
864     <extension base="saml:AttributeDesignatorType">
865       <sequence>
866         <element ref="saml:AttributeValue" maxOccurs="unbounded" />
867       </sequence>
868     </extension>
869   </complexContent>
870 </complexType>
```

871 2.4.5.1.1 Element <AttributeValue>

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

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

Deleted: March 29th 2002

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

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

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

3. SAML Protocol

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

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



887
888 Figure 2: SAML Request-Response Protocol

3.1. Schema Header and Namespace Declarations

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

```
892 <schema  
893   targetNamespace="urn:oasis:names:tc:SAML:1.0:protocol"  
894   xmlns="http://www.w3.org/2001/XMLSchema"  
895   xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protoco"  
896   xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertior"  
897   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
898   elementFormDefault="unqualified">  
899   <import namespace="urn:oasis:names:tc:SAML:1.0:assertio"  
900     schemaLocation="draft-sstc-schema-assertion-29.xsd" />  
901   <import namespace="http://www.w3.org/2000/09/xmldsig#"  
902     schemaLocation="xmldsig-core-schema.xsd" />  
903   <annotation>  
904     <documentation>draft-sstc-schema-protocol-29.xsd</documentation>  
905   </annotation>  
906   ...  
907 </schema>
```

- Deleted: http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-protocol-29.xsd
- Deleted: http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-protocol-29.xsd
- Deleted: http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-protocol-29.xsd
- Deleted: http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-assertion-29.xsd

3.2. Requests

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

3.2.1. Complex Type RequestAbstractType

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

- 915 RequestID [Required]
916 An identifier for the request. It is of type **IDType**, and MUST follow the requirements
917 specified by that type for identifier uniqueness. The values of the RequestID attribute in a
918 request and the InResponseTo attribute in the corresponding response MUST match.
- 919 MajorVersion [Required]
920 The major version of this request. The identifier for the version of SAML defined in this
921 specification is 1. Processing of this attribute is specified in Section 3.4.2.

Deleted: March 29th 2002

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

925 IssueInstant [Required]
926 The time instant of issue of the request. The time value is encoded in UTC as described in
927 section 1.2.1.

928 <RespondWith> [Any Number]
929 Each <RespondWith> element specifies a type of response that is acceptable to the
930 requestor.

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

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

```
934 <complexType name="RequestAbstractType" abstract="true">  
935 <sequence>  
936 <element ref="saml:RespondWith"  
937 minOccurs="0" maxOccurs="unbounded"/>  
938 <element ref="ds:Signature" minOccurs="0"/>  
939 </sequence>  
940 <attribute name="RequestID" type="saml:IDType" use="required"/>  
941 <attribute name="MajorVersion" type="integer" use="required"/>  
942 <attribute name="MinorVersion" type="integer" use="required"/>  
943 <attribute name="IssueInstant" type="dateTime" use="required"/>  
944 </complexType>
```

945 3.2.1.1. Element <RespondWith>

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

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

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

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

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

```
964 <element name="RespondWith" type="QName"/>
```

965 3.2.2. Element <Request>

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

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

- 970 <Query>
971 An extension point that allows extension schemas to define new types of query.
- 972 <SubjectQuery>
973 An extension point that allows extension schemas to define new types of query that specify
974 a single SAML subject.
- 975 <AuthenticationQuery>
976 Makes a query for authentication information.
- 977 <AttributeQuery>
978 Makes a query for attribute information.
- 979 <AuthorizationDecisionQuery>
980 Makes a query for an authorization decision.
- 981 <AssertionIDReference> [One or more]
982 Requests assertions by reference to its assertion identifier.
- 983 <AssertionArtifact> [One or more]
984 Requests assertions by supplying an assertion artifact that represents it.

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

```
987 <element name="Request" type="samlp:RequestType"/>  
988 <complexType name="RequestType">  
989 <complexContent>  
990 <extension base="samlp:RequestAbstractType">  
991 <choice>  
992 <element ref="samlp:Query"/>  
993 <element ref="samlp:SubjectQuery"/>  
994 <element ref="samlp:AuthenticationQuery"/>  
995 <element ref="samlp:AttributeQuery"/>  
996 <element ref="samlp:AuthorizationDecisionQuery"/>  
997 <element ref="saml:AssertionIDReference" maxOccurs="unbounded"/>  
998 <element ref="samlp:AssertionArtifact" maxOccurs="unbounded"/>  
999 </choice>  
1000 </extension>  
1001 </complexContent>  
1002 </complexType>
```

1003 3.2.3. Element <AssertionArtifact>

1004 The <AssertionArtifact> element is used to specify the assertion artifact that represents an
1005 assertion.

1006 The following schema fragment defines the <AssertionArtifact> element:

```
1007 <element name="AssertionArtifact" type="string"/>
```

1008 3.3. Queries

1009 The following sections define the SAML constructs that contain query information.

1010 3.3.1. Element <Query>

1011 The <Query> element is an extension point that allows new SAML queries to be defined. Its
1012 **QueryAbstractType** is abstract; extension elements MUST use the `xsi:type` attribute to indicate

Deleted: March 29th 2002

1013 the derived type. **QueryAbstractType** is the base type from which all SAML query elements are
1014 derived.

1015 The following schema fragment defines the <Query> element and its **QueryAbstractType**
1016 complex type:

```
1017 <element name="Query" type="saml:QueryAbstractType" />  
1018 <complexType name="QueryAbstractType" abstract="true" />
```

1019 3.3.2. Element <SubjectQuery>

1020 The <SubjectQuery> element is an extension point that allows new SAML queries that specify a
1021 single SAML subject. Its **SubjectQueryAbstractType** complex type is abstract; extension elements
1022 MUST use the `xsi:type` attribute to indicate the derived type. **SubjectQueryAbstractType** adds
1023 the <Subject> element.

1024 The following schema fragment defines the <SubjectQuery> element and its
1025 **SubjectQueryAbstractType** complex type:

```
1026 <element name="SubjectQuery" type="saml:SubjectQueryAbstractType" />  
1027 <complexType name="SubjectQueryAbstractType" abstract="true">  
1028 <complexContent>  
1029 <extension base="saml:QueryAbstractType">  
1030 <sequence>  
1031 <element ref="saml:Subject" />  
1032 </sequence>  
1033 </extension>  
1034 </complexContent>  
1035 </complexType>
```

1036 3.3.3. Element <AuthenticationQuery>

1037 The <AuthenticationQuery> element is used to make the query "What assertions containing
1038 authentication statements are available for this subject?" A successful response will be in the form
1039 of assertions containing authentication statements.

1040 Note: The <AuthenticationQuery> MAY NOT be used as a request for a new authentication
1041 using credentials provided in the request. The <AuthenticationQuery> is a request for
1042 statements about authentication acts which have occurred in a previous interaction between the
1043 indicated principal and the Authentication Authority.

1044 This element is of type **AuthenticationQueryType**, which extends **SubjectQueryAbstractType**
1045 with the addition of the following element:

1046 <AuthenticationMethod> [Optional]

1047 A filter for possible responses. If it is present, the query made is "What assertions
1048 containing authentication statements do you have for this subject with the supplied
1049 authentication method?"

Deleted: ConfirmationMethod

Deleted: confirmation

1050 In response to an authentication query, a responder returns assertions with authentication
1051 statements as follows:

1052 ? First, rules given in section 3.4.4 for matching against the <Subject> element of the query
1053 identify the assertions that may be returned.

1054 ? Further, if the <AuthenticationMethod> element is present in the query, at least one
1055 <AuthenticationMethod> element in the set of returned assertions MUST match. It is
1056 OPTIONAL for the complete set of all such matching assertions to be returned in the
1057 response.

1058 The <Subject> element in the returned assertions MUST be identical to the <Subject> element
1059 of the query. If the <ConfirmationMethod> element is present in the query, at least one

Deleted: March 29th 2002

1060 <ConfirmationMethod> element in the response MUST match. It is OPTIONAL for the complete
1061 set of all such matching assertions to be returned in the response.

1062 The following schema fragment defines the <AuthenticationQuery> type and its
1063 **AuthenticationQueryType** complex type:

```
1064 <element name="AuthenticationQuery" type="saml:AuthenticationQueryType"/>  
1065 <complexType name="AuthenticationQueryType">  
1066 <complexContent>  
1067 <extension base="saml:SubjectQueryAbstractType">  
1068 <attribute name="AuthenticationMethod" type="anyURI"/>  
1069 </extension>  
1070 </complexContent>  
1071 </complexType>
```

Deleted: <sequence>¶
<element
ref="saml:ConfirmationMethod
" minOccurs="0"/>¶
</sequence>¶

1072 3.3.4. Element <AttributeQuery>

1073 The <AttributeQuery> element is used to make the query "Return the requested attributes for
1074 this subject." A successful response will be in the form of assertions containing attribute statements.
1075 This element is of type **AttributeQueryType**, which extends **SubjectQueryAbstractType** with the
1076 addition of the following element and attribute:

1077 Resource [Optional]

1078 The Resource attribute if present specifies that the attribute query is made in response to a
1079 specific authorization decision relating to the resource. The responder MAY use the
1080 resource attribute to establish the scope of the request. It is permitted for this attribute to
1081 have the value of the empty URI reference (""), and the meaning is defined to be "the start
1082 of the current document", as specified by [RFC 2396]§ 4.2.

1083 If the resource attribute is specified and the responder does not wish to support resource-
1084 specific attribute queries, or if the resource value provided is invalid or unrecognized, then it
1085 SHOULD respond with a top-level StatusCode value of Responder and a second-level
1086 code value of ResourceNotRecognized

Deleted: SHOULD respond with a
SAML status of
"Error.Receiver.ResourceNotRecogni
zed".

1087 <AttributeDesignator> [Any Number] (see Section 2.4.5.1)

1088 Each <AttributeDesignator> element specifies an attribute whose value is to be
1089 returned. If no attributes are specified, it indicates that all attributes allowed by policy are
1090 requested.

Deleted: If no attributes are
specified, the list of desired attributes
is implicit and application-specific.

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

```
1093 <element name="AttributeQuery" type="saml:AttributeQueryType"/>  
1094 <complexType name="AttributeQueryType">  
1095 <complexContent>  
1096 <extension base="saml:SubjectQueryAbstractType">  
1097 <sequence>  
1098 <element ref="saml:AttributeDesignator"  
1099 minOccurs="0" maxOccurs="unbounded"/>  
1100 </sequence>  
1101 <attribute name="Resource" type="anyURI reference" use="optional"/>  
1102 </extension>  
1103 </complexContent>  
1104 </complexType>
```

1105 3.3.5. Element <AuthorizationDecisionQuery>

1106 The <AuthorizationDecisionQuery> element is used to make the query "Should these
1107 actions on this resource be allowed for this subject, given this evidence?" A successful response
1108 will be in the form of assertions containing authorization decision statements. This element is of
1109 type **AuthorizationDecisionQueryType**, which extends **SubjectQueryAbstractType** with the
1110 addition of the following elements and attribute:

Deleted: March 29th 2002

- 1111 Resource [Required]
- 1112 A URI reference indicating the resource for which authorization is requested.
- 1113 <Action> [One or More]
- 1114 The actions for which authorization is requested.
- 1115 <Evidence> [Any Number]
- 1116 An assertion that the responder MAY rely on in making its response.

1117 The following schema fragment defines the <AuthorizationDecisionQuery> element and its
 1118 **AuthorizationDecisionQueryType** complex type:

```

1119 <element name="AuthorizationDecisionQuery"
1120 type="samlp:AuthorizationDecisionQueryType"/>
1121 <complexType name="AuthorizationDecisionQueryType">
1122   <complexContent>
1123     <extension base="samlp:SubjectQueryAbstractType">
1124       <sequence>
1125         <element ref="saml:Action" maxOccurs="unbounded"/>
1126         <element ref="saml:Evidence"
1127           minOccurs="0" maxOccurs="unbounded"/>
1128       </sequence>
1129       <attribute name="Resource" type="anyURI" use="required"/>
1130     </extension>
1131   </complexContent>
1132 </complexType>
```

1133 3.4. Responses

1134 The following sections define the SAML constructs that contain response information.

1135 3.4.1. Complex Type ResponseAbstractType

1136 All SAML responses are of types that are derived from the abstract **ResponseAbstractType**
 1137 complex type. This type defines common attributes and elements that are associated with all SAML
 1138 responses:

- 1139 ResponseID [Required]
- 1140 An identifier for the response. It is of type **IDType**, and MUST follow the requirements
 1141 specified by that type for identifier uniqueness.
- 1142 InResponseTo [Optional]
- 1143 A reference to the identifier of the request to which the response corresponds, if any. If the
 1144 response is not generated in response to a request, or if the RequestID of a request cannot
 1145 be determined (because the request is malformed), then this attribute MUST NOT be
 1146 present. Otherwise, it MUST be present and match the value of the corresponding
 1147 RequestID attribute.
- 1148 MajorVersion [Required]
- 1149 The major version of this response. The identifier for the version of SAML defined in this
 1150 specification is 1. Processing of this attribute is specified in Section 3.4.4.
- 1151 MinorVersion [Required]
- 1152 The minor version of this response. The identifier for the version of SAML defined in this
 1153 specification is 0. Processing of this attribute is specified in Section 3.4.4.
- 1154 IssueInstant [Optional]
- 1155 The time instant of issue of the request. The time value is encoded in UTC as described in
 1156 section 1.2.1.

Deleted: Required

Deleted: A reference to the identifier of the request to which the response corresponds. The value of this attribute MUST match the value of the corresponding RequestID attribute.

Deleted: ¶

Deleted: March 29th 2002

1157 Recipient [Optional]
1158 The intended recipient of this response. This is useful to prevent malicious forwarding of
1159 responses to unintended recipients, a protection that is required by some use profiles. It is
1160 set by the generator of the response to a URI reference that identifies the intended
1161 recipient. If present, the actual recipient MUST check that the URI reference identifies the
1162 recipient or a resource managed by the recipient. If it does not, the response MUST be
1163 discarded.

1164 <Signature> [Optional]
1165 An XML Signature that authenticates the assertion, see section 5.

1166 The following schema fragment defines the **ResponseAbstractType** complex type:

```
1167 <complexType name="ResponseAbstractType" abstract="true">  
1168 <sequence>  
1169 <element ref="ds:Signature" minOccurs="0"/>  
1170 </sequence>  
1171 <attribute name="ResponseID" type="saml:IDType" use="required"/>  
1172 <attribute name="InResponseTo" type="saml:IDReferenceType"  
1173 use="optional"/>  
1174 <attribute name="MajorVersion" type="integer" use="required"/>  
1175 <attribute name="MinorVersion" type="integer" use="required"/>  
1176 <attribute name="IssueInstant" type="dateTime" use="required"/>  
1177 <attribute name="Recipient" type="anyURI" use="optional"/>  
1178 </complexType>
```

Deleted: required

Deleted: dateTime

1179 3.4.2. Element <Response>

1180 The <Response> element specifies the status of the corresponding SAML request and a list of
1181 zero or more assertions that answer the request. It has the complex type **ResponseType**, which
1182 extends **ResponseAbstractType** by adding the following elements (in an unbounded mixture):

1183 <Status> [Required] (see Section 3.4.3)
1184 A code representing the status of the corresponding request.

1185 <Assertion> [Any Number] (see Section 2.3.2)
1186 Specifies an assertion by value.

1187 The following schema fragment defines the <Response> element and its **ResponseType** complex
1188 type:

```
1189 <element name="Response" type="samlp:ResponseType"/>  
1190 <complexType name="ResponseType">  
1191 <complexContent>  
1192 <extension base="samlp:ResponseAbstractType">  
1193 <sequence>  
1194 <element ref="samlp:Status"/>  
1195 <element ref="saml:Assertion"  
1196 minOccurs="0" maxOccurs="unbounded"/>  
1197 </sequence>  
1198 </extension>  
1199 </complexContent>  
1200 </complexType>
```

1201 3.4.3. Element <Status>

1202 The <Status> element :

1203 <StatusCode> [Required]
1204 A code representing the status of the corresponding request.

1205 <StatusMessage> [Any Number]
1206 A message which MAY be returned to an operator.

Deleted: March 29th 2002

1207 <StatusDetail> [Optional]
1208 Specifies additional information concerning an error condition.

1209 The following schema fragment defines the <Status> element and its **StatusType** complex type:

```
1210 <element name="Status" type="samlp:StatusType" />  
1211 <complexType name="StatusType">  
1212   <sequence>  
1213     <element ref="samlp:StatusCode" />  
1214     <element ref="samlp:StatusMessage"  
1215       minOccurs="0" maxOccurs="unbounded" />  
1216     <element ref="samlp:StatusDetail" minOccurs="0" />  
1217   </sequence>  
1218 </complexType>
```

1219 3.4.3.1. Element <StatusCode>

1220 The <StatusCode> element specifies one or more nested codes representing the status of the
1221 corresponding request. top-most code value MUST be one of the values defined below.
1222 Subsequent nested code values, if present, may provide more specific information concerning a
1223 particular error.

Deleted: a

1224 Value [Required]
1225 The status code value as defined below.

Deleted: and an option sub code providing more specific information concerning a particular error status:[]

1226 <StatusCode> [Optional]
1227 An optional subordinate status code value that provides more specific information on an
1228 error condition.

Deleted: Sub

1229 The following top-level StatusCode Value QNames are defined. The responder MUST NOT
1230 include a code not listed below except by nesting it below one of the listed values.

Deleted: The following StatusCode values are defined:

1231 Success

1232 The request succeeded.

1233 VersionMismatch

1234 The receiver could not process the request because the version was incorrect.

1235 Receiver

1236 The request could not be performed due to an error at the receiving end.

1237 Sender

1238 The request could not be performed due to an error in the sender or in the request

1239 The following second-level status codes are referenced at various places in the specification.
1240 Additional subcodes MAY be defined in future versions of the SAML specification.

1241 RequestVersionTooHigh

1242 The protocol version specified in the request is a major upgrade from the highest protocol
1243 version supported by the responder.

1244 RequestVersionTooLow

1245 The responder cannot respond to the particular request using the SAML version specified
1246 in the request because it is too low.

1247 RequestVersionDeprecated

1248 The responder does not respond to any requests with the protocol version specified in the
1249 request.

1250 TooManyResponses

1251 The response would contain more elements than the responder will return.

1252 RequestDenied

1253 The responder is able to process the request but has chosen not to respond. MAY be used

Deleted: March 29th 2002

1254 when the responder is concerned about the security context of the request or the sequence
1255 of requests received from a particular client.

1256 All status code values defined in this document are QNames associated with the SAML protocol
1257 namespace [SAML] and MUST be prefixed appropriately when they appear in SAML messages.
1258 SAML extensions and SAML Responders are free to define more specific status codes in other
1259 namespaces, but MAY NOT define additional codes in either the SAML assertion or protocol
1260 namespaces.

1261 The QNames defined as status codes SHOULD only be used in the StatusCode element's Value
1262 attribute and have the above semantics only in that context.

1263 The following schema fragment defines the <StatusCode> element and its **StatusCodeType**
1264 complex type;

```
1265 <element name="StatusCode" type="samlp:StatusCodeType"/>  
1266 <complexType name="StatusCodeType">  
1267 <sequence>  
1268 <element ref="samlp:StatusCode" minOccurs="0"/>  
1269 </sequence>  
1270 <attribute name="Value" type="QName" use="required"/>  
1271 </complexType>
```

1272 3.4.3.2. Element <StatusMessage>

1273 The <StatusMessage> element specifies a message that MAY be returned to an operator:

1274 The following schema fragment defines the <StatusMessage> element and its
1275 **StatusMessageType** complex type:

```
1276 <element name="StatusMessage" type="string"/>
```

1277 3.4.3.3. Element <StatusDetail>

1278 The <StatusDetail> element MAY be used to specify additional information concerning an error
1279 condition.

1280 The following schema fragment defines the <StatusDetail> element and its **StatusDetailType**
1281 complex type:

```
1282 <element name="StatusDetail" type="samlp:StatusDetailType"/>  
1283 <complexType name="StatusDetailType">  
1284 <sequence>  
1285 <any namespace="##any"  
1286 processContents="lax" minOccurs="0" maxOccurs="unbounded"/>  
1287 </sequence>  
1288 </complexType>
```

1289 3.4.4. Responses to <AuthenticationQuery> and <AttributeQuery>

1290 Responses to Authentication and Attribute queries are constructed by matching against the
1291 <saml:Subject> element found within the <AuthenticationQuery> or <AttributeQuery>
1292 elements. In response to these queries, every assertion returned by a SAML responder MUST
1293 contain at least one statement whose <saml:Subject> element **strongly matches** the
1294 <saml:Subject> element found in the query.

1295 A <saml:Subject> element S1 strongly matches S2 if and only if:

- 1296 1 If S2 includes a <saml:NameIdentifier> element, then S1 must include an identical
1297 <saml:NameIdentifier> element.
- 1298 2 If S2 includes a <saml:SubjectConfirmation> element, then S1 must include an
1299 identical <saml:SubjectConfirmation> element.

Deleted: and the
StatusCodeEnumType simple type

Deleted: Sub

Deleted: samlp:StatusCodeEnum
Type

```
Deleted: <simpleType  
name="StatusCodeEnumType">¶  
<restriction  
base="QName">¶  
<enumeration  
value="samlp:Success"/>¶  
<enumeration  
value="samlp:VersionMismatch  
"/>¶  
<enumeration  
value="samlp:Receiver"/>¶  
<enumeration  
value="samlp:Sender"/>¶  
</restriction>¶  
</simpleType>¶  
<#>Element <SubStatusCode>¶
```

The <SubStatusCode> element
specifies an additional code
representing the status of the
corresponding request:¶
Value [Required]

The status code value as defined
below.¶

<SubStatusCode> [Optional]

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

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

RequestVersionTooHigh

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

RequestVersionTooLow

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

RequestVersionDeprecated

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

TooManyResponses

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

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

```
<element  
name="SubStatusCode"  
type="samlp:SubStatusCodeTyp  
e"/>¶  
<complexType  
name="SubStatusCodeType">¶  
<sequence>¶
```

... [2]

Deleted: March 29th 2002

1300 If the responder cannot provide an assertion with any statement(s) satisfying the constraints
1301 expressed by a query, the <saml:Response> element MUST NOT contain an <assertion> element
1302 and MUST include a <saml:StatusCode> with value "Success". It MAY return a
1303 <saml:StatusMessage> with additional information.

4. SAML Versioning

1304

1305 SAML version information appears in the following elements:

1306 ? <Assertion>

1307 ? <Request>

1308 ? <Response>

1309 The version numbering of the SAML assertion is independent of the version number of the SAML
1310 request-response protocol. The version information for each consists of a major version number
1311 and a minor version number, both of which are integers. In accordance with industry practice a
1312 version number SHOULD be presented to the user in the form *Major.Minor*. This document defines
1313 SAML Assertions 1.0 and SAML Protocol 1.0.

1314 The version number $Major_B.Minor_B$ is higher than the version number $Major_A.Minor_A$ if and only if:

1315 $Major_B > Major_A ? ((Major_B = Major_A) ? Minor_B > Minor_A)$

1316 Each revision of SAML SHALL assign version numbers to assertions, requests, and responses that
1317 are the same as or higher than the corresponding version number in the SAML version that
1318 immediately preceded it.

1319 New versions of SAML SHALL assign new version numbers as follows:

1320 ? **Documentation change:** $(Major_B = Major_A) ? (Minor_B > Minor_A)$

1321 If the major and minor version numbers are unchanged, the new version *B* only introduces
1322 changes to the documentation that raise no compatibility issues with an implementation of
1323 version *A*.

1324 ? **Minor upgrade:** $(Major_B = Major_A) ? (Minor_B > Minor_A)$

1325 If the major version number of versions *A* and *B* are the same and the minor version
1326 number of *B* is higher than that of *A*, the new SAML version MAY introduce changes to the
1327 SAML schema and semantics but any changes that are introduced in *B* SHALL be
1328 compatible with version *A*.

1329 ? **Major upgrade:** $Major_B > Major_A$

1330 If the major version of *B* number is higher than the major version of *A*, Version *B* MAY
1331 introduce changes to the SAML schema and semantics that are incompatible with *A*.

4.1. Assertion Version

1333 A SAML authority MUST NOT issue any assertion whose version number is not supported.

Deleted: application

1334 A SAML authority MUST reject any assertion whose major version number is not supported.

Deleted: application

1335 A SAML authority MAY reject any assertion whose version number is higher than the highest
1336 supported version.

Deleted: application

4.2. Request Version

1338 A SAML authority SHOULD issue requests that specify the highest SAML version supported by
1339 both the sender and recipient.

Deleted: application

1340 If the SAML authority does not know the capabilities of the recipient it should assume that it
1341 supports the highest SAML version supported by the sender.

Deleted: application

Deleted: March 29th 2002

1342 4.3. Response Version

1343 A SAML authority MUST NOT issue responses that specify a higher SAML version number than the
1344 corresponding request.

Deleted: application

1345 A SAML authority MUST NOT issue a response that has a major version number that is lower than
1346 the major version number of the corresponding request except to report the error
1347 RequestVersionTooHigh

Deleted: application

1348 An error response resulting from incompatible protocol versions MUST result in reporting a top-level
1349 StatusCode value of VersionMismatch, and MAY result in reporting one of the following second-
1350 level values:

1351 RequestVersionTooHigh

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

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

1354 RequestVersionTooLow

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

1357 RequestVersionDeprecated

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

Deleted: March 29th 2002

5. SAML & XML-Signature Syntax and Processing

1360

1361

1362 SAML Assertions, Request and Response messages may be signed, with the following benefits:

1363 ? An Assertion signed by the asserting party (AP). This supports :

1364 (1) Message integrity

1365 (2) Authentication of the asserting party to a relying party (RP)

1366 (3) If the signature is based on the asserting party's public-private key pair, then it
1367 also provides for non-repudiation of origin.

Deleted: issuer

Deleted: issuer

Deleted: issuer

1368 ? A SAML request or a SAML response message signed by the message originator. This
1369 supports :

1370 (1) Message integrity

1371 (2) Authentication of message origin to a destination

1372 (3) If the signature is based on the originator's public-private key pair, then it also
1373 provides for non-repudiation of origin.

1374 Note :

1375 ? SAML documents may be the subject of signatures from different packaging contexts.
1376 **[XMLSig]** provides a framework for signing in XML and is the framework of choice.
1377 However, signing may also take place in the context of S/MIME or Java objects that
1378 contain SAML documents. One goal is to ensure compatibility with this type of "foreign"
1379 digital signing.

1380 ? It is useful to characterize situations when a digital signature is NOT required in SAML.

1381 Assertions:

1382 The asserting party has provided the assertion to the relying party, authenticated by means
1383 other than digital signature and the channel is secure. In other words, the RP has obtained the
1384 assertion from the AP directly (no intermediaries) through a secure channel and the AP has
1385 authenticated to the RP.

1386 Request/Response messages:

1387 The originator has authenticated to the destination and the destination has obtained the
1388 assertion directly from the originator (no intermediaries) through secure channel(s).

1389 Many different techniques are available for "direct" authentication and secure channel between
1390 two parties. The list includes SSL, HMAC, password-based login etc. Also the security
1391 requirement depends on the communicating applications and the nature of the assertion
1392 transported.

1393 All other contexts require the use of digital signature for assertions and request and response
1394 messages. Specifically:

1395 (1) An assertion obtained by a relying party from an entity other than the asserting party MUST
1396 be signed by the asserting party.

1397 (2) A SAML message arriving at a destination from an entity other than the originating site
1398 MUST be signed by the origin site.

Deleted: issuer

5.1. Signing Assertions

1399

1400 All SAML assertions MAY be signed using the XML Signature. This is reflected in the assertion
1401 schema – Section 2.3.

Deleted: March 29th 2002

1402 5.2. Request/Response Signing

1403 All SAML requests and responses MAY be signed using the XML Signature. This is reflected in the
1404 schema – Section 3.2 & 3.4.

1405 5.3. Signature Inheritance

1406 5.3.1. Rationale

1407 SAML assertions may be embedded within request or response messages or other XML
1408 messages, which may be signed. Request or response messages may themselves be contained
1409 within other messages that are based on other XML messaging frameworks (e.g., SOAP) and the
1410 composite object may be the subject of a signature. Another possibility is that SAML assertions or
1411 request/response messages are embedded within a non-XML messaging object (e.g., MIME
1412 package) and signed.

1413 In such a case, the SAML sub-message (Assertion, request, response) may be viewed as inheriting
1414 a signature from the "super-signature" over the enclosing object, provided certain constraints are
1415 met.

1416 (1) An assertion may be viewed as inheriting a signature from a super signature, if the super
1417 signature applies all the elements within the assertion.

1418 A SAML request or response may be viewed as inheriting a signature from a super signature, if the
1419 super signature applies to all of the elements within the response.

1420 5.3.2. Rules for SAML Signature Inheritance

1421 Signature inheritance occurs when SAML message (assertion/request/response) is not signed but
1422 is enclosed within signed SAML such that the signature applies to all of the elements within the
1423 message. In such a case, the SAML message is said to inherit the signature and may be
1424 considered equivalent to the case where it is explicitly signed. The SAML message inherits the
1425 "closest enclosing signature".

1426 But if SAML messages need to be passed around by themselves, or embedded in other messages,
1427 they would need to be signed as per section 5.1

1428 5.4. XML Signature Profile

1429 The XML Signature [XMLSig] specification calls out a general XML syntax for signing data with
1430 many flexibilities and choices. This section details the constraints on these facilities so that SAML
1431 processors do not have to deal with the full generality of XML Signature processing.

1432 5.4.1. Signing formats

1433 XML Signature has three ways of representing signature in a document viz: enveloping, enveloped
1434 and detached.

1435 SAML assertions and protocols MUST use the enveloped signatures for signing assertions and
1436 protocols. SAML processors should support use of RSA signing and verification for public key
1437 operations.

1438 5.4.2. CanonicalizationMethod

1439 XML Signature REQUIRES the Canonical XML (omits comments)
1440 (<http://www.w3.org/TR/2001/REC-xml-c14n-20010315>). SAML implementations SHOULD use
1441 Canonical XML with no comments.

Deleted: March 29th 2002

1442 **5.4.3. Transforms**

1443 [XMLSig] REQUIRES the enveloped signature transform
1444 <http://www.w3.org/2000/09/xmldsig#enveloped-signature>

1445 **5.4.4. KeyInfo**

1446 SAML does not restrict or impose any restrictions in this area. Therefore following [XMLSig]
1447 keyInfo may be absent.

1448 **5.4.5. Binding between statements in a multi-statement assertion**

1449 Use of signing does not affect semantics of statements within assertions in any way, as stated in
1450 this document Sections 1 through 4.

1451 6. SAML Extensions

1452 The SAML schemas support extensibility. An example of an application that extends SAML
1453 assertions is the XTAML system for management of embedded trust roots [XTAML]. The following
1454 sections explain how to use the extensibility features in SAML to create extension schemas.

1455 Note that elements in the SAML schemas are not blocked from substitution, so that all SAML
1456 elements MAY serve as the head element of a substitution group. Also, types are not defined as
1457 final, so that all SAML types MAY be extended and restricted. The following sections discuss
1458 only elements that have been specifically designed to support extensibility.

1459 6.1. Assertion Schema Extension

1460 The SAML assertion schema is designed to permit separate processing of the assertion package
1461 and the statements it contains, if the extension mechanism is used for either part.

1462 The following elements are intended specifically for use as extension points in an extension
1463 schema; their types are set to abstract, so that the use of an xsi:type attribute with these
1464 elements is REQUIRED:

1465 ? <Assertion>

1466 ? <Condition>

1467 ? <Statement>

1468 ? <SubjectStatement>

1469 ? <AdviceElement>

1470 In addition, the following elements that are directly usable as part of SAML MAY be extended:

1471 ? <AuthenticationStatement>

1472 ? <AuthorizationDecisionStatement>

1473 ? <AttributeStatement>

1474 ? <AudienceRestrictionCondition>

1475 Finally, the following elements are defined to allow elements from arbitrary namespaces within
1476 them, which serves as a built-in extension point without requiring an extension schema:

1477 ? <AttributeValue>

1478 ? <Advice>

1479 6.2. Protocol Schema Extension

1480 The following elements are intended specifically for use as extension points in an extension
1481 schema; their types are set to abstract, so that the use of an xsi:type attribute with these
1482 elements is REQUIRED:

1483 ? <Query>

1484 ? <SubjectQuery>

1485 In addition, the following elements that are directly usable as part of SAML MAY be extended:

1486 ? <Request>

- 1487 ? <AuthenticationQuery>
- 1488 ? <AuthorizationDecisionQuery>
- 1489 ? <AttributeQuery>
- 1490 ? <Response>

1491 6.3. Use of Type Derivation and Substitution Groups

1492 W3C XML Schema [Schema1] provides two principal mechanisms for specifying an element of an
1493 extended type: type derivation and substitution groups.

1494 For example, a <Statement> element can be assigned the type **NewStatementType** by means of
1495 the `xsi:type` attribute. For such an element to be schema-valid, **NewStatementType** needs to be
1496 derived from **StatementType**. The following example of a SAML assertion assumes that the
1497 extension schema (represented by the `new:` prefix) has defined this new type:

```
1498 <saml:Assertion ...>  
1499   <saml:Statement xsi:type="new:NewStatementType">  
1500     ...  
1501   </saml:Statement>  
1502 </saml:Assertion>
```

1503 Alternatively, the extension schema can define a <NewStatement> element that is a member of a
1504 substitution group that has <Statement> as a head element. For the substituted element to be
1505 schema-valid, it needs to have a type that matches or is derived from the head element's type. The
1506 following is an example of an extension schema fragment that defines this new element:

```
1507 <xsd:element "NewStatement" type="new:NewStatementType"  
1508   substitutionGroup="saml:Statement" />
```

1509 The substitution group declaration allows the <NewStatement> element to be used anywhere the
1510 SAML <Statement> element can be used. The following is an example of a SAML assertion that
1511 uses the extension element:

```
1512 <saml:Assertion ...>  
1513   <new:NewStatement>  
1514     ...  
1515   </new:NewStatement>  
1516 </saml:Assertion>
```

1517 The choice of extension method has no effect on the semantics of the XML document but does
1518 have implications for interoperability.

1519 The advantages of type derivation are as follows:

- 1520 ? A document can be more fully interpreted by a parser that does not have access to the
1521 extension schema because a "native" SAML element is available.
- 1522 ? At the time of writing, some W3C XML Schema validators do not support substitution
1523 groups, whereas the `xsi:type` attribute is widely supported.

1524 The advantage of substitution groups is that a document can be explained without the need to
1525 explain the functioning of the `xsi:type` attribute.

1526 7. SAML-Defined Identifiers

1527 The following sections define URI-based identifiers for common authentication protocols and
1528 actions.

1529 Where possible an existing URN is used to specify a protocol. In the case of IETF protocols the
1530 URN of the most current RFC that specifies the protocol is used. URI references created
1531 specifically for SAML have the initial stem:

1532 `urn:oasis:names:tc:SAML:1.0;`

Deleted: <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-29>

1533 7.1. Authentication Method and Confirmation Method 1534 Identifiers

1535 The `<AuthenticationMethod>` and `<SubjectConfirmationMethod>` elements perform
1536 different functions within the SAML architecture although both can contain some of the same
1537 values. `<AuthenticationMethod>` is a part of an Authentication Statement, which describes an
1538 authentication act which occurred in the past. The `<AuthenticationMethod>` indicates how that
1539 authentication was done. Note that the authentication statement does not provide the means to
1540 perform that authentication, such as a password, key or certificate.

1541 In contrast, `<SubjectConfirmationMethod>` is a part of the `<SubjectConfirmation>`, which
1542 is used to allow the Relying Party to confirm that the request or message came from the System
1543 Entity that corresponds to the Subject in the statement. The `<SubjectConfirmationMethod>`
1544 indicates the method which the Relying Party can use to do this in the future. This may or may not
1545 have any relationship to an authentication that was performed previously. Unlike the Authentication
1546 Method, the `<SubjectConfirmationMethod>` will usually be accompanied with some piece of
1547 information, such as a certificate or key, which will allow the Relying Party to perform the necessary
1548 check.

1549 There are many `<SubjectConfirmationMethod>`, because there are many different SAML
1550 usage scenarios. A few examples are:

1551 1. A user logs in with a password, but a temporary passcode or cookie is issued for confirmation
1552 purposes to avoid repeated exposure of the long term password.

1553 2. There is no login, but an application request is digitally signed. The associated public key is used
1554 for confirmation.

1555 3. The user logs in using Kerberos and a Kerberos ticket is used subsequently for confirmation.
1556 Notice that in this case although both the Authentication Method and the
1557 `<SubjectConfirmationMethod>` are Kerberos, what happens at each step is actually different.
1558 (See [RFC 1510](#))

Deleted: RFC 1510

Comment: <http://lists.oasis-open.org/archives/security-services/200203/msg00043.html>

1559 The following identifiers are defined to refer to common authentication protocols. Where Base64
1560 encoding is specified the data is encoded as specified by [RFC 2045](#).

Deleted: `<#>SAML Artifact:¶`
URI: <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-28#artifact¶>
`<SubjectConfirmationData: Base64 (Artifact) ¶`
The subject of the assertion is the party that can present the SAML Artifact value specified in `<SubjectConfirmationData>¶`

1561 7.1.1. SAML Artifact (SHA-1):

1562 URI: `urn:oasis:names:tc:SAML:1.0:cm:artifact-sha1`

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

1563 `<SubjectConfirmationData: Base64 (SHA1 (Artifact))`

1564 The subject of the assertion is the party that can present a SAML Artifact such that the SHA1 digest
1565 of the specified artifact matches the value specified in `<SubjectConfirmationData>`.

Deleted: #

Deleted: March 29th 2002

1566 **7.1.2. Holder of Key:**

1567 URI: <urn:oasis:names:tc:SAML:1.0:cm:Holder-Of-Key>

1568 <ds:KeyInfo>: Any cryptographic key

1569 The subject of the assertion is the party that can demonstrate that it is the holder of the private
1570 component of the key specified in <ds:KeyInfo>.

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

Deleted: #

1571 **7.1.3. Bearer Indication:**

1572 URI: <urn:oasis:names:tc:SAML:1.0:cm:BearerIndication>

1573 The subject of the assertion is the bearer of the assertion.

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

Deleted: #

1574 **7.1.4. Sender Vouches:**

1575 URI: <urn:oasis:names:tc:SAML:1.0:cm:sender-vouches>

1576 Indicates that no other information is available about the context of use of the assertion. The
1577 Relying party SHOULD utilize other means to determine if it should process the assertion further.

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

Deleted: #

1578 **7.1.5. Password (Pass-Through):**

1579 URI: <urn:oasis:names:tc:SAML:1.0:cm:password>

1580 <SubjectConfirmationData>: *Base64 (Password)*

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

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

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

Deleted: #

1584 **7.1.6. Password (One-Way-Function SHA-1):**

1585 URI: <urn:oasis:names:tc:SAML:1.0:cm:password-sha1>

1586 <SubjectConfirmationData>: *Base64 (SHA1 (Password))*

1587 The subject of the assertion is the party that can present the password such that the SHA1 digest of
1588 the specified password matches the value specified in <SubjectConfirmationData>.

1589 The username of the subject is specified by means of the <NameIdentifier> element.

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

Deleted: #

1590 **7.1.7. Kerberos**

1591 URI: <urn:ietf:rfc:1510>

1592 <SubjectConfirmationData>: A Kerberos Ticket

1593 The subject is authenticated by means of the Kerberos protocol [RFC 1510], an instantiation of the
1594 Needham-Schroeder symmetric key authentication mechanism [Needham78].

Deleted: Error! Reference source not found.

1595 **7.1.8. SSL/TLS Certificate Based Client Authentication:**

1596 URI: <urn:ietf:rfc:2246>

1597 <ds:KeyInfo>: Any cryptographic key

Deleted: March 29th 2002

1598 **7.1.9. Object Authenticator (SHA-1):**

1599 **URI:** <urn:oasis:names:tc:SAML:1.0:cm:object-sha1>

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

1600 `<SubjectConfirmationData>`: *Base64 (SHA1 (Object))*

Deleted: #

1601 This authenticator element is the result of computing a digest, using the SHA-1 hash algorithm. It is
1602 used when the subject can be represented as a binary string, for example when it is an XML
1603 document or the disk image of executable code. Any preprocessing of the subject prior to
1604 computation of the digest is out of scope. The name of the subject should be conveyed in an
1605 accompanying `NameIdentifier` element.

1606 **7.1.10. PKCS#7**

1607 **URI:** <urn:ietf:rfc:2315>

1608 `<SubjectConfirmationData>`: *Base64 (PKCS#7 (Object))*

1609 This authenticator element is signed data in PKCS#7 format [PKCS#7]. The posited identity of the
1610 signer must be conveyed in an accompanying `NameIdentifier` element. This subject type may be
1611 included in the subject field of an authentication query, in which case the corresponding response
1612 indicates whether the posited signer is, indeed, the signer. It may be included in an attribute query,
1613 in which case, the requested attribute values for the subject authenticated by the signed data are
1614 returned. It may be included in an authorization query, in which case, the access request
1615 represented by the signed data shall be identified by the accompanying object element, and the
1616 corresponding assertion containing an authorization decision statement indicates whether the
1617 signer is authorized for the access request represented by the object element.

1618 **7.1.11. Cryptographic Message Syntax**

1619 **URI:** <urn:ietf:rfc:2630>

1620 `<SubjectConfirmationData>`: *Base64 (CMS (Object))*

Deleted: 7.1.11

1621 This authenticator element is signed data in CMS format [CMS]. See also [7.1.10](#).

1622 **7.1.12. XML Digital Signature**

1623 **URI:** <urn:ietf:rfc:3075>

1624 `<SubjectConfirmationData>`: *Base64 (XML-SIG (Object))*

1625 `<ds:KeyInfo>`: A cryptographic signing key

Deleted: 7.1.11

1626 This authenticator element is signed data in XML Signature format. See also [7.1.10](#).

1627 **7.2. Action Namespace Identifiers**

1628 The following identifiers MAY be used in the [Namespace attribute of the <Action> element](#), (see
1629 Section 2.4.4.1) to refer to common sets of actions to perform on resources.

Deleted: ActionNamespace attribute

1630 **7.2.1. Read/Write/Execute/Delete/Control:**

1631 **URI:** <urn:oasis:names:tc:SAML:1.0:action:rwedc>

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

1632 Defined actions:

Deleted: #

Deleted: March 29th 2002

1633 Read Write Execute Delete Control
 1634 These actions are interpreted in the normal manner, i.e.
 1635 Read
 1636 The subject may read the resource
 1637 Write
 1638 The subject may modify the resource
 1639 Execute
 1640 The subject may execute the resource
 1641 Delete
 1642 The subject may delete the resource
 1643 Control
 1644 The subject may specify the access control policy for the resource

1645 **7.2.2. Read/Write/Execute/Delete/Control with Negation:**

1646 **URI:** [urn:oasis:names:tc:SAML:1.0:action:rwdc-negation](#)

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

1647 Defined actions:

1648 Read Write Execute Delete Control ~Read ~Write ~Execute ~Delete ~Control

1649 The actions specified in section 7.2.1 are interpreted in the same manner described there. Actions prefixed with a tilde ~ are negated permissions and are used to affirmatively specify that the stated permission is denied. Thus a subject described as being authorized to perform the action ~Read is affirmatively denied read permission.

1653 A [SAML authority](#) MUST NOT authorize both an action and its negated form.

Deleted: n application

1654 **7.2.3. Get/Head/Put/Post:**

1655 **URI:** [urn:oasis:names:tc:SAML:1.0:action:ghpp](#)

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

1656 Defined actions:

1657 GET HEAD PUT POST

1658 These actions bind to the corresponding HTTP operations. For example a subject authorized to perform the GET action on a resource is authorized to retrieve it.

1660 The GET and HEAD actions loosely correspond to the conventional read permission and the PUT and POST actions to the write permission. The correspondence is not exact however since a HTTP GET operation may cause data to be modified and a POST operation may cause modification to a resource other than the one specified in the request. For this reason a separate Action URI reference specifier is provided.

1665 **7.2.4. UNIX File Permissions:**

1666 **URI:** [urn:oasis:names:tc:SAML:1.0:action:unix](#)

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

1667 The defined actions are the set of UNIX file access permissions expressed in the numeric (octal) notation.

1669 The action string is a four digit numeric code:

1670 *extended user group world*

1671 Where the *extended* access permission has the value

Deleted: March 29th 2002

1672 +2 if sgid is set
1673 +4 if suid is set
1674 The *user group* and *world* access permissions have the value
1675 +1 if execute permission is granted
1676 +2 if write permission is granted
1677 +4 if read permission is granted
1678 For example 0754 denotes the UNIX file access permission: user read, write and execute, group
1679 read and execute and world read.

1680 8. SAML Schema Listings

1681 The following sections contain complete listings of the assertion and protocol schemas for SAML.

1682 8.1. Assertion Schema

1683 Following is a complete listing of the SAML assertion schema [SAML-XSD].

```
1684 <?xml version="1.0" encoding="UTF-8"?>
1685 <!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Phill HallamBaker
1686 (VeriSign Inc.) -->
1687 <schema
1688   targetNamespace="urn:oasis:names:tc:SAML:1.0:assertion"
1689   xmlns="http://www.w3.org/2001/XMLSchema"
1690   xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
1691   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
1692   elementFormDefault="unqualified">
1693   <import namespace="http://www.w3.org/2000/09/xmldsig#"
1694     schemaLocation="xmldsig-core-schema.xsd"/>
1695   <annotation>
1696     <documentation>draft-sstc-schema-assertion-29.xsd</documentation>
1697   </annotation>
1698   <simpleType name="IDType">
1699     <restriction base="string"/>
1700   </simpleType>
1701   <simpleType name="IDReferenceType">
1702     <restriction base="string"/>
1703   </simpleType>
1704   <simpleType name="DecisionType">
1705     <restriction base="string">
1706       <enumeration value="Permit"/>
1707       <enumeration value="Deny"/>
1708       <enumeration value="Indeterminate"/>
1709     </restriction>
1710   </simpleType>
1711   <element name="AssertionIDReference" type="saml:IDReferenceType"/>
1712   <element name="Assertion" type="saml:AssertionType"/>
1713   <complexType name="AssertionType">
1714     <sequence>
1715       <element ref="saml:Conditions" minOccurs="0"/>
1716       <element ref="saml:Advice" minOccurs="0"/>
1717       <choice maxOccurs="unbounded">
1718         <element ref="saml:Statement"/>
1719         <element ref="saml:SubjectStatement"/>
1720         <element ref="saml:AuthenticationStatement"/>
1721         <element ref="saml:AuthorizationDecisionStatement"/>
1722         <element ref="saml:AttributeStatement"/>
1723       </choice>
1724       <element ref="ds:Signature" minOccurs="0"/>
1725     </sequence>
1726     <attribute name="MajorVersion" type="integer" use="required"/>
1727     <attribute name="MinorVersion" type="integer" use="required"/>
1728     <attribute name="AssertionID" type="saml:IDType" use="required"/>
1729     <attribute name="Issuer" type="string" use="required"/>
1730     <attribute name="IssueInstant" type="dateTime" use="required"/>
1731   </complexType>
1732   <element name="Conditions" type="saml:ConditionsType"/>
1733   <complexType name="ConditionsType">
1734     <choice minOccurs="0" maxOccurs="unbounded">
1735       <element ref="saml:AudienceRestrictionCondition"/>
1736       <element ref="saml:Condition"/>
```

Deleted: <http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-assertion-29.xsd>

Deleted: <http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-assertion-29.xsd>

Deleted: March 29th 2002

```

1737 </choice>
1738 <attribute name="NotBefore" type="dateTime" use="optional"/>
1739 <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
1740 </complexType>
1741 <element name="Condition" type="saml:ConditionAbstractType"/>
1742 <complexType name="ConditionAbstractType" abstract="true"/>
1743 <element name="AudienceRestrictionCondition"
1744     type="saml:AudienceRestrictionConditionType"/>
1745 <complexType name="AudienceRestrictionConditionType">
1746 <complexContent>
1747 <extension base="saml:ConditionAbstractType">
1748 <sequence>
1749 <element ref="saml:Audience" maxOccurs="unbounded"/>
1750 </sequence>
1751 </extension>
1752 </complexContent>
1753 </complexType>
1754 <element name="Audience" type="anyURI"/>
1755 <element name="Advice" type="saml:AdviceType"/>
1756 <complexType name="AdviceType">
1757 <choice minOccurs="0" maxOccurs="unbounded">
1758 <element ref="saml:AssertionIDReference"/>
1759 <element ref="saml:Assertion"/>
1760 <any namespace="##other" processContents="lax"/>
1761 </choice>
1762 </complexType>
1763 <element name="Statement" type="saml:StatementAbstractType"/>
1764 <complexType name="StatementAbstractType" abstract="true"/>
1765 <element name="SubjectStatement" type="saml:SubjectStatementAbstractType"/>
1766 <complexType name="SubjectStatementAbstractType" abstract="true">
1767 <complexContent>
1768 <extension base="saml:StatementAbstractType">
1769 <sequence>
1770 <element ref="saml:Subject"/>
1771 </sequence>
1772 </extension>
1773 </complexContent>
1774 </complexType>
1775 <element name="Subject" type="saml:SubjectType"/>
1776 <complexType name="SubjectType">
1777 <choice>
1778 <sequence>
1779 <element ref="saml:NameIdentifier"/>
1780 <element ref="saml:SubjectConfirmation" minOccurs="0"/>
1781 </sequence>
1782 <element ref="saml:SubjectConfirmation"/>
1783 </choice>
1784 </complexType>
1785 <element name="NameIdentifier" type="saml:NameIdentifierType"/>
1786 <complexType name="NameIdentifierType">
1787 <simpleContent>
1788 <extension base="string">
1789 <attribute name="NameQualifier" type="string" use="optional"/>
1790 <attribute name="Format" type="anyURI" use="optional"/>
1791 </extension>
1792 </simpleContent>
1793 </complexType>
1794 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>
1795 <complexType name="SubjectConfirmationType">
1796 <sequence>
1797 <element ref="saml:ConfirmationMethod" maxOccurs="unbounded"/>
1798 <element ref="saml:SubjectConfirmationData" minOccurs="0"/>
1799 <element ref="ds:KeyInfo" minOccurs="0"/>

```

Deleted: <element ref="saml:AudienceRestrictionCondition"/>¶

Deleted: <element ref="saml:AdviceElement"/>¶

Deleted: <element name="AdviceElement" type="saml:AdviceAbstractType"/>¶
<complexType name="AdviceAbstractType"/>¶

Deleted: March 29th 2002

```

1800     </sequence>
1801 </complexType>
1802 <element name="SubjectConfirmationData" type="string"/>
1803 <element name="ConfirmationMethod" type="anyURI"/>
1804 <element name="AuthenticationStatement"
1805     type="saml:AuthenticationStatementType"/>
1806 <complexType name="AuthenticationStatementType">
1807     <complexContent>
1808         <extension base="saml:SubjectStatementAbstractType">
1809             <sequence>
1810                 <element ref="saml:SubjectLocality" minOccurs="0"/>
1811                 <element ref="saml:AuthorityBinding"
1812                     minOccurs="0" maxOccurs="unbounded"/>
1813             </sequence>
1814             <attribute name="AuthenticationMethod" type="anyURI"/>
1815             <attribute name="AuthenticationInstant" type="dateTime"/>
1816         </extension>
1817     </complexContent>
1818 </complexType>
1819 <element name="SubjectLocality"
1820     type="saml:SubjectLocalityType"/>
1821 <complexType name="SubjectLocalityType">
1822     <attribute name="IPAddress" type="string" use="optional"/>
1823     <attribute name="DNSAddress" type="string" use="optional"/>
1824 </complexType>
1825 <element name="AuthorityBinding" type="saml:AuthorityBindingType"/>
1826 <complexType name="AuthorityBindingType">
1827     <attribute name="AuthorityKind" type="QName" use="required"/>
1828     <attribute name="Location" type="anyURI" use="required"/>
1829     <attribute name="Binding" type="anyURI" use="required"/>
1830 </complexType>
1831 <element name="AuthorizationDecisionStatement"
1832     type="saml:AuthorizationDecisionStatementType"/>
1833 <complexType name="AuthorizationDecisionStatementType">
1834     <complexContent>
1835         <extension base="saml:SubjectStatementAbstractType">
1836             <sequence>
1837                 <element ref="saml:Action" maxOccurs="unbounded"/>
1838                 <element ref="saml:Evidence" minOccurs="0"/>
1839             </sequence>
1840             <attribute name="Resource" type="anyURI" use="required"/>
1841             <attribute name="Decision" type="saml:DecisionType" use="required"/>
1842         </extension>
1843     </complexContent>
1844 </complexType>
1845 <element name="Action" type="saml:ActionType"/>
1846 <complexType name="ActionType">
1847     <simpleContent>
1848         <extension base="string">
1849             <attribute name="Namespace" type="anyURI"/>
1850         </extension>
1851     </simpleContent>
1852 </complexType>
1853 <element name="Evidence" type="saml:EvidenceType"/>
1854 <complexType name="EvidenceType">
1855     <choice maxOccurs="unbounded">
1856         <element ref="saml:AssertionIDReference"/>
1857         <element ref="saml:Assertion"/>
1858     </choice>
1859 </complexType>
1860 <element name="AttributeStatement" type="saml:AttributeStatementType"/>
1861 <complexType name="AttributeStatementType">
1862     <complexContent>

```

Deleted: AuthenticationLocality

Deleted: AuthenticationLocality

Deleted:

Inserted: SubjectLocality

Deleted: AuthenticationLocality

Deleted: AuthenticationLocalityType

Deleted: March 29th 2002

```

1863     <extension base="saml:SubjectStatementAbstractType">
1864         <sequence>
1865             <element ref="saml:Attribute" maxOccurs="unbounded"/>
1866         </sequence>
1867     </extension>
1868 </complexContent>
1869 </complexType>
1870 <element name="AttributeDesignator" type="saml:AttributeDesignatorType"/>
1871 <complexType name="AttributeDesignatorType">
1872     <attribute name="AttributeName" type="string" use="required"/>
1873     <attribute name="AttributeNamespace" type="anyURI" use="required"/>
1874 </complexType>
1875 <element name="Attribute" type="saml:AttributeType"/>
1876 <complexType name="AttributeType">
1877     <complexContent>
1878         <extension base="saml:AttributeDesignatorType">
1879             <sequence>
1880                 <element ref="saml:AttributeValue" maxOccurs="unbounded"/>
1881             </sequence>
1882         </extension>
1883     </complexContent>
1884 </complexType>
1885 <element name="AttributeValue" type="saml:anyType"/>
1886 </schema>

```

1887 8.2. Protocol Schema

1888 Following is a complete listing of the SAML protocol schema [SAMPL-XSD].

```

1889 <?xml version="1.0" encoding="UTF-8"?>
1890 <!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Phill HallamBaker
1891 (VeriSign Inc.) -->
1892 <schema
1893     targetNamespace="urn:oasis:names:tc:SAML:1.0:protocol"
1894     xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
1895     xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
1896     xmlns:sampl="urn:oasis:names:tc:SAML:1.0:protocol"
1897     xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="unqualified">
1898     <import
1899         namespace="urn:oasis:names:tc:SAML:1.0:assertion"
1900         schemaLocation="draft-sstc-schema-assertion-29.xsd"/>
1901     <import namespace="http://www.w3.org/2000/09/xmldsig#"
1902         schemaLocation="xmldsig-core-schema.xsd"/>
1903     <annotation>
1904         <documentation>draft-sstc-schema-protocol-29.xsd</documentation>
1905     </annotation>
1906     <complexType name="RequestAbstractType" abstract="true">
1907         <sequence>
1908             <element ref="sampl:RespondWith"
1909                 minOccurs="0" maxOccurs="unbounded"/>
1910             <element ref="ds:Signature" minOccurs="0"/>
1911         </sequence>
1912         <attribute name="RequestID" type="saml:IDType" use="required"/>
1913         <attribute name="MajorVersion" type="integer" use="required"/>
1914         <attribute name="MinorVersion" type="integer" use="required"/>
1915         <attribute name="IssueInstant" type="dateTime" use="required"/>
1916     </complexType>
1917     <element name="RespondWith" type="QName"/>
1918     <element name="Request" type="sampl:RequestType"/>
1919     <complexType name="RequestType">
1920         <complexContent>
1921             <extension base="sampl:RequestAbstractType">
1922                 <choice>

```

Deleted: <http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-protocol-29.xsd>

Deleted: <http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-assertion-29.xsd>

Deleted: <http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-protocol-29.xsd>

Deleted: <http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-assertion-29.xsd>

Deleted: <attribute name="Recipient" type="dateTime" use="optional"/>

Deleted: March 29th 2002

```

1923     <element ref="samlp:Query" />
1924     <element ref="samlp:SubjectQuery" />
1925     <element ref="samlp:AuthenticationQuery" />
1926     <element ref="samlp:AttributeQuery" />
1927     <element ref="samlp:AuthorizationDecisionQuery" />
1928     <element ref="saml:AssertionID" minOccurs="unbounded" />
1929     <element ref="samlp:AssertionArtifact" minOccurs="unbounded" />
1930   </choice>
1931 </extension>
1932 </complexContent>
1933 </complexType>
1934 <element name="AssertionArtifact" type="string" />
1935 <element name="Query" type="samlp:QueryAbstractType" />
1936 <complexType name="QueryAbstractType" abstract="true" />
1937 <element name="SubjectQuery" type="samlp:SubjectQueryAbstractType" />
1938 <complexType name="SubjectQueryAbstractType" abstract="true">
1939   <complexContent>
1940     <extension base="samlp:QueryAbstractType">
1941       <sequence>
1942         <element ref="saml:Subject" />
1943       </sequence>
1944     </extension>
1945   </complexContent>
1946 </complexType>
1947 <element name="AuthenticationQuery" type="samlp:AuthenticationQueryType" />
1948 <complexType name="AuthenticationQueryType">
1949   <complexContent>
1950     <extension base="samlp:SubjectQueryAbstractType">
1951       <attribute name="AuthenticationMethod" type="anyURI" />
1952     </extension>
1953   </complexContent>
1954 </complexType>
1955 <element name="AttributeQuery" type="samlp:AttributeQueryType" />
1956 <complexType name="AttributeQueryType">
1957   <complexContent>
1958     <extension base="samlp:SubjectQueryAbstractType">
1959       <sequence>
1960         <element ref="saml:AttributeDesignator"
1961           minOccurs="0" maxOccurs="unbounded" />
1962       </sequence>
1963       <attribute name="Resource" type="anyURI" use="optional" />
1964     </extension>
1965   </complexContent>
1966 </complexType>
1967 <element name="AuthorizationDecisionQuery"
1968   type="samlp:AuthorizationDecisionQueryType" />
1969 <complexType name="AuthorizationDecisionQueryType">
1970   <complexContent>
1971     <extension base="samlp:SubjectQueryAbstractType">
1972       <sequence>
1973         <element ref="saml:Action" maxOccurs="unbounded" />
1974         <element ref="saml:Evidence"
1975           minOccurs="0" maxOccurs="unbounded" />
1976       </sequence>
1977       <attribute name="Resource" type="anyURI" use="required" />
1978     </extension>
1979   </complexContent>
1980 </complexType>
1981 <complexType name="ResponseAbstractType" abstract="true">
1982   <sequence>
1983     <element ref="ds:Signature" minOccurs="0" />
1984   </sequence>
1985   <attribute name="ResponseID" type="saml:IDType" use="required" />

```

Deleted: <sequence>¶
 <element
 ref="saml:ConfirmationMethod
 " minOccurs="0" />¶
 </sequence>¶

Deleted: March 29th 2002

1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030

```
<attribute name="InResponseTo" type="saml:IDReferenceType"
  use="optional"/>
<attribute name="MajorVersion" type="integer" use="required"/>
<attribute name="MinorVersion" type="integer" use="required"/>
<attribute name="IssueInstant" type="dateTime" use="required"/>
<attribute name="Recipient" type="anyURI" use="optional"/>
</complexType>
<element name="Response" type="saml:ResponseType"/>
<complexType name="ResponseType">
  <complexContent>
    <extension base="saml:ResponseAbstractType">
      <sequence>
        <element ref="saml:Status"/>
        <element ref="saml:Assertion"
          minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<element name="Status" type="saml:StatusType"/>
<complexType name="StatusType">
  <sequence>
    <element ref="saml:StatusCode"/>
    <element ref="saml:StatusMessage"
      minOccurs="0" maxOccurs="unbounded"/>
    <element ref="saml:StatusDetail" minOccurs="0"/>
  </sequence>
</complexType>
<element name="StatusCode" type="saml:StatusCodeType"/>
<complexType name="StatusCodeType">
  <sequence>
    <element ref="saml:StatusCode" minOccurs="0"/>
  </sequence>
  <attribute name="Value" type="QName" use="required"/>
</complexType>
<element name="StatusMessage" type="string"/>
<element name="StatusDetail" type="saml:StatusDetailType"/>
<complexType name="StatusDetailType">
  <sequence>
    <any namespace="##any"
      processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</complexType>
</schema>
```

Deleted: required

Deleted: Sub

Deleted: saml:StatusCodeEnum Type

Deleted: <simpleType name="StatusCodeEnumType">¶
<restriction base="QName">¶
<enumeration value="saml:Success"/>¶
<enumeration value="saml:VersionMismatch"/>¶
<enumeration value="saml:Receiver"/>¶
<enumeration value="saml:Sender"/>¶
</restriction>¶
</simpleType>¶
<element name="SubStatusCode" type="saml:SubStatusCodeType"/>¶
<complexType name="SubStatusCodeType">¶
<sequence>¶
<element ref="saml:SubStatusCode" minOccurs="0"/>¶
</sequence>¶
<attribute name="Value" type="QName" use="required"/>¶
</complexType>¶

Deleted: March 29th 2002

9. References

2031

- 2032 [Kern-84] B. Kernighan, Rob Pike *The UNIX Programming Environment*, (March
2033 1984) Prentice Hall Computer Books;
- 2034 [Needham78] R. Needham et al., *Using Encryption for Authentication in Large Networks*
2035 *of Computers*, Communications of the ACM, Vol. 21 (12), pp. 993-999,
2036 December 1978.
- 2037 [PKCS1] B. Kaliski, *PKCS #1: RSA Encryption Version 2.0*, RSA Laboratories, also
2038 IETF RFC 2437, October 1998. <http://www.ietf.org/rfc/rfc2437.txt>
- 2039 [PKCS7] B. Kaliski., "PKCS #7: Cryptographic Message Syntax, Version 1.5.", RFC
2040 2315, March 1998.
- 2041 [RFC 1510] J. Kohl, C. Neuman. *The Kerberos Network Authentication Requestor (V5)*.
2042 September 1993. <http://www.ietf.org/rfc/rfc1510.txt>
- 2043 [RFC 2045] ~~N. Freed, N. Borenstein. *Multipurpose Internet Mail Extensions (MIME)*
2044 *Part One: Format of Internet Message Bodies*
2045 <http://www.ietf.org/rfc/rfc2045.txt> IETF RFC 2045, November 1996.~~
- 2046 [RFC 2104] H. Krawczyk et al., *HMAC: Keyed Hashing for Message Authentication*,
2047 <http://www.ietf.org/rfc/rfc2104.txt>, IETF RFC 2104, February 1997.
- 2048 [RFC 2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,
2049 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997
- 2050 [RFC 2246] T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. January 1999.
2051 <http://www.ietf.org/rfc/rfc2246.txt>
- 2052 [RFC 2396] T. Berners-Lee et. al., *Uniform Resource Identifiers (URI): Generic Syntax*
2053 <http://www.ietf.org/rfc/rfc2396.txt> IETF?
- 2054 [RFC 2630] R. Housley. *Cryptographic Message Syntax*. June 1999.
2055 <http://www.ietf.org/rfc/rfc630.txt>
- 2056 [RFC 2648] R. Moats. *A URN Namespace for IETF Documents*. August 1999.
2057 <http://www.ietf.org/rfc/rfc2648.txt>
- 2058 [RFC 3075] D. Eastlake, J. Reagle, D. Solo. *XML-Signature Syntax and Processing*.
2059 March 2001. <http://www.ietf.org/rfc/rfc3075.txt>
- 2060 [SAMLBind] P. Mishra et al., *Bindings and Profiles for the OASIS Security Assertion*
2061 *Markup Language (SAML)*, [http://www.oasis-](http://www.oasis-open.org/committees/security/docs/draft-sstc-bindings-model-07.pdf)
2062 [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),
2063 OASIS, December 2001.
- 2064 [SAMLConform] **TBS**
- 2065 [SAMLGloss] J. Hodges et al., *Glossary for the OASIS Security Assertion Markup*
2066 *Language (SAML)*, [http://www.oasis-](http://www.oasis-open.org/committees/security/docs/draft-sstc-glossary-02.pdf)
2067 [open.org/committees/security/docs/draft-sstc-glossary-02.pdf](http://www.oasis-open.org/committees/security/docs/draft-sstc-glossary-02.pdf), OASIS,
2068 December 2001.
- 2069 [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)
2070 [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),
2071 OASIS, December 2001.
- 2072 [SAMLSecure] **TBS**
- 2073 [SAMLXSD] P. Hallam-Baker et al., *SAML assertion schema*, [http://www.oasis-](http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-assertion-21.xsd)
2074 [open.org/committees/security/docs/draft-sstc-schema-assertion-21.xsd](http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-assertion-21.xsd),
2075 OASIS, December 2001.

Deleted: ¶

Deleted: March 29th 2002

2076	[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.
2077		
2078		
2079	[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.
2080		
2081		
2082	[UNICODE-C]	M. Davis, M. J. Dürst, http://www.unicode.org/unicode/reports/tr15/tr15-21.html , UNICODE Consortium
2083		
2084	[W3C-CHAR]	M. J. Dürst, <i>Requirements for String Identity Matching and String Indexing</i> http://www.w3.org/TR/W3C-charreq , World Wide Web Consortium.
2085		
2086	[W3C-CharMod]	M. J. Dürst, <i>Unicode Normalization Forms</i> http://www.w3.org/TR/charmod/ , World Wide Web Consortium.
2087		
2088	[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.
2089		
2090	[XMLEnc]	<i>XML Encryption Specification</i> , In development.
2091	[XMLSig]	D. Eastlake et al., <i>XML-Signature Syntax and Processing</i> , http://www.w3.org/TR/xmlsig-core/ , World Wide Web Consortium.
2092		
2093	[XMLSig-XSD]	XML Signature Schema available from http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd .
2094		
2095	[XTAML]	P. Hallam-Baker, <i>XML Trust Axiom Markup Language 1.0</i> , http://www.xmltrustcenter.org/ , VeriSign Inc. September 2001.
2096		

2149
2150

Deleted: Joe Pato, Hewlett Packard

Deleted: Darren Platt, RSA Security
Irving Reid, Baltimore Technologies
Krishna Sankar, Cisco Systems Inc

Deleted: March 29th 2002

2151 Appendix A. Notices

2152 OASIS takes no position regarding the validity or scope of any intellectual property or other rights
2153 that might be claimed to pertain to the implementation or use of the technology described in this
2154 document or the extent to which any license under such rights might or might not be available;
2155 neither does it represent that it has made any effort to identify any such rights. Information on
2156 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS
2157 website. Copies of claims of rights made available for publication and any assurances of licenses to
2158 be made available, or the result of an attempt made to obtain a general license or permission for
2159 the use of such proprietary rights by implementors or users of this specification, can be obtained
2160 from the OASIS Executive Director.

2161 OASIS invites any interested party to bring to its attention any copyrights, patents or patent
2162 applications, or other proprietary rights which may cover technology that may be required to
2163 implement this specification. Please address the information to the OASIS Executive Director.

2164 Copyright © The Organization for the Advancement of Structured Information Standards [OASIS]
2165 2001. All Rights Reserved.

2166 This document and translations of it may be copied and furnished to others, and derivative works
2167 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,
2168 published and distributed, in whole or in part, without restriction of any kind, provided that the above
2169 copyright notice and this paragraph are included on all such copies and derivative works. However,
2170 this document itself may not be modified in any way, such as by removing the copyright notice or
2171 references to OASIS, except as needed for the purpose of developing OASIS specifications, in
2172 which case the procedures for copyrights defined in the OASIS Intellectual Property Rights
2173 document must be followed, or as required to translate it into languages other than English.

2174 The limited permissions granted above are perpetual and will not be revoked by OASIS or its
2175 successors or assigns.

2176 This document and the information contained herein is provided on an "AS IS" basis and OASIS
2177 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO
2178 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY
2179 RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A
2180 PARTICULAR PURPOSE.

Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)	1
1. Introduction	6
1.1. Notation	6
1.2. Schema Organization and Namespaces	6
1.2.1. Time Values.	7
1.2.2. Comparing SAML values	7
1.3. SAML Concepts (Non-Normative)	7
1.3.1. Overview	7
1.3.2. SAML and URI-Based Identifiers	9
1.3.3. SAML and Extensibility	9
2. SAML Assertions	10
2.1. Schema Header and Namespace Declarations	10
2.2. Simple Types	10
2.2.1. Simple Types IDType and IDReferenceType	10
2.2.2. Simple Type DecisionType	11
2.3. Assertions	11
2.3.1. Element <AssertionID>	11
2.3.2. Element <Assertion>	11
2.3.2.1. Element <Conditions>	13
2.3.2.1.1. Attributes NotBefore and NotOnOrAfter	13
2.3.2.1.2. Element <Condition>	14
2.3.2.1.3. Elements <AudienceRestrictionCondition> and <Audience>	14
2.3.2.2. Elements <Advice> and <AdviceElement>	15
2.4. Statements	15
2.4.1. Element <Statement>	15
2.4.2. Element <SubjectStatement>	15
2.4.2.1. Element <Subject>	16
2.4.2.2. Element <NameIdentifier>	16
2.4.2.3. Elements <SubjectConfirmation>, <ConfirmationMethod>, and <SubjectConfirmationData>	17
2.4.3. Element <AuthenticationStatement>	18
2.4.3.1. Element <AuthenticationLocality>	18
2.4.3.2. Element <AuthorityBinding>	19
2.4.4. Element <AuthorizationDecisionStatement>	19
2.4.4.1. Element <Action>	20
2.4.4.2. Element <Evidence>	21
2.4.5. Element <AttributeStatement>	21
2.4.5.1. Elements <AttributeDesignator> and <Attribute>	22

2.4.5.1.1	Element <AttributeValue>	22
3.	SAML Protocol	23
3.1.	Schema Header and Namespace Declarations	23
3.2.	Requests	23
3.2.1.	Complex Type RequestAbstractType	23
3.2.1.1.	Element <RespondWith>	24
3.2.2.	Element <Request>	25
3.2.3.	Element <AssertionArtifact>	25
3.3.	Queries	25
3.3.1.	Element <Query>	26
3.3.2.	Element <SubjectQuery>	26
3.3.3.	Element <AuthenticationQuery>	26
3.3.4.	Element <AttributeQuery>	27
3.3.5.	Element <AuthorizationDecisionQuery>	27
3.4.	Responses	28
3.4.1.	Complex Type ResponseAbstractType	28
3.4.2.	Element <Response>	29
3.4.3.	Element <Status>	29
3.4.3.1.	Element <StatusCode>	30
3.4.3.2.	Element <SubStatusCode>	30
3.4.3.3.	Element <StatusMessage>	31
3.4.3.4.	Element <StatusDetail>	31
3.4.4.	Responses to <AuthenticationQuery> and <AttributeQuery>	31
4.	SAML Versioning	33
4.1.	Assertion Version	33
4.2.	Request Version	33
4.3.	Response Version	34
5.	SAML & XML-Signature Syntax and Processing	35
5.1.	Signing Assertions	35
5.2.	Request/Response Signing	36
5.3.	Signature Inheritance	36
5.3.1.	Rationale	36
5.3.2.	Rules for SAML Signature Inheritance	36
5.4.	XML Signature Profile	36
5.4.1.	Signing formats	36

5.4.2. CanonicalizationMethod	36
5.4.3. Transforms	37
5.4.4. KeyInfo	37
5.4.5. Binding between statements in a multi-statement assertion	37
6. SAML Extensions	38
6.1. Assertion Schema Extension	38
6.2. Protocol Schema Extension	38
6.3. Use of Type Derivation and Substitution Groups	39
7. SAML-Defined Identifiers	40
7.1. Authentication Method and Confirmation Method Identifiers	40
7.1.1. SAML Artifact:	40
7.1.2. SAML Artifact (SHA-1):	41
7.1.3. Holder of Key:	41
7.1.4. Bearer Indication:	41
7.1.5. Sender Vouches:	41
7.1.6. Password (Pass-Through):	41
7.1.7. Password (One-Way-Function SHA-1):	41
7.1.8. Kerberos	41
7.1.9. SSL/TLS Certificate Based Client Authentication:	42
7.1.10. Object Authenticator (SHA-1):	42
7.1.11. PKCS#7	42
7.1.12. Cryptographic Message Syntax	42
7.1.13. XML Digital Signature	42
7.2. Action Namespace Identifiers	43
7.2.1. Read/Write/Execute/Delete/Control:	43
7.2.2. Read/Write/Execute/Delete/Control with Negation:	43
7.2.3. Get/Head/Put/Post:	43
7.2.4. UNIX File Permissions:	44
8. SAML Schema Listings	45
8.1. Assertion Schema	45
8.2. Protocol Schema	48
9. References	52
10. Acknowledgements	54
Appendix A. Notices	56

```

<simpleType name="StatusCodeEnumType">
  <restriction base="QName">
    <enumeration value="samlp:Success"/>
    <enumeration value="samlp:VersionMismatch"/>
    <enumeration value="samlp:Receiver"/>
    <enumeration value="samlp:Sender"/>
  </restriction>
</simpleType>

```

3.4.3.2.Element <SubStatusCode>

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

Value [Required]

The status code value as defined below.

<SubStatusCode> [Optional]

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

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

RequestVersionTooHigh

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

RequestVersionTooLow

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

RequestVersionDeprecated

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

TooManyResponses

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

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

```

<element name="SubStatusCode" type="samlp:SubStatusCodeType"/>
<complexType name="SubStatusCodeType">
  <sequence>
    <element ref="samlp:SubStatusCode" minOccurs="0"/>
  </sequence>
  <attribute name="Value" type="QName" use="required"/>
</complexType>

```