



# Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1

## Committee Specification, 18 July 2003

### Document identifier:

sstc-saml-core-1.1-cs-03

### Location:

[http://www.oasis-open.org/committees/documents.php?wg\\_abbrev=security](http://www.oasis-open.org/committees/documents.php?wg_abbrev=security)

### Editors:

Eve Maler, Sun Microsystems ([eve.maler@sun.com](mailto:eve.maler@sun.com))  
Prateek Mishra, Netegrity, Inc. ([pmishra@netegrity.com](mailto:pmishra@netegrity.com))  
Rob Philpott, RSA Security ([rphilpott@rsasecurity.com](mailto:rphilpott@rsasecurity.com))

### Contributors:

Stephen Farrell, Baltimore Technologies  
Irving Reid, Baltimore Technologies  
Hal Lockhart, BEA Systems  
David Orchard, BEA Systems  
Krishna Sankar, Cisco Systems  
Carlisle Adams, Entrust  
Tim Moses, Entrust  
Nigel Edwards, Hewlett-Packard  
Joe Pato, Hewlett-Packard  
Bob Blakley, IBM  
Marlena Erdos, IBM  
Scott Cantor, individual  
RL "Bob" Morgan, individual  
Marc Chanliau, Netegrity  
Chris McLaren, Netegrity  
Charles Knouse, Oblix  
Simon Godik, Overkeer  
Darren Platt, formerly of RSA Security  
Jahan Moreh, Sigaba  
Jeff Hodges, Sun Microsystems  
Phillip Hallam-Baker, VeriSign (former editor)

### Abstract:

This specification defines the syntax and semantics for XML-encoded assertions about authentication, attributes and authorization, and for the protocol that conveys this information.

39 **Status:**

40 This document is a **Committee Specification** of the OASIS Security Services Technical  
41 Committee. This document is updated periodically on no particular schedule. Send comments to  
42 the editors.

43 Committee members should send comments on this specification to the [security-](mailto:security-services@lists.oasis-open.org)  
44 [services@lists.oasis-open.org](mailto:security-services@lists.oasis-open.org) list. Others should subscribe to and send comments to the  
45 [security-services-comment@lists.oasis-open.org](mailto:security-services-comment@lists.oasis-open.org) list. To subscribe, send an email message to  
46 [security-services-comment-request@lists.oasis-open.org](mailto:security-services-comment-request@lists.oasis-open.org) with the word "subscribe" as the body of  
47 the message.

48 For information on whether any patents have been disclosed that may be essential to  
49 implementing this specification, and any offers of patent licensing terms, please refer to the  
50 Intellectual Property Rights section of the Security Services TC web page ([http://www.oasis-](http://www.oasis-open.org/committees/security/)  
51 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)).

52 For information on errata discovered in this specification, please refer to the most recent errata  
53 document which can be found in the document repository at the Security Services TC web page  
54 (<http://www.oasis-open.org/committees/security/>).

# Table of Contents

56	1	Introduction.....	6
57	1.1	Notation.....	6
58	1.2	Schema Organization and Namespaces.....	6
59	1.2.1	String and URI Values.....	7
60	1.2.2	Time Values.....	7
61	1.2.3	ID and ID Reference Values.....	7
62	1.2.4	Comparing SAML Values.....	7
63	1.3	SAML Concepts (Non-Normative).....	8
64	1.3.1	Overview.....	8
65	1.3.2	SAML and URI-Based Identifiers.....	9
66	1.3.3	SAML and Extensibility.....	10
67	2	SAML Assertions.....	11
68	2.1	Schema Header and Namespace Declarations.....	11
69	2.2	Simple Types.....	11
70	2.2.1	Simple Type DecisionType.....	12
71	2.3	Assertions.....	12
72	2.3.1	Element <AssertionIDReference>.....	12
73	2.3.2	Element <Assertion>.....	12
74	2.3.2.1	Element <Conditions>.....	14
75	2.3.2.1.1	Attributes NotBefore and NotOnOrAfter.....	15
76	2.3.2.1.2	Element <Condition>.....	15
77	2.3.2.1.3	Elements <AudienceRestrictionCondition> and <Audience>.....	15
78	2.3.2.1.4	Element <DoNotCacheCondition>.....	16
79	2.3.2.2	Element <Advice>.....	16
80	2.4	Statements.....	17
81	2.4.1	Element <Statement>.....	17
82	2.4.2	Element <SubjectStatement>.....	17
83	2.4.2.1	Element <Subject>.....	17
84	2.4.2.2	Element <NameIdentifier>.....	18
85	2.4.2.3	Elements <SubjectConfirmation>, <ConfirmationMethod>, and <SubjectConfirmationData>.....	18
86	2.4.3	Element <AuthenticationStatement>.....	19
87	2.4.3.1	Element <SubjectLocality>.....	20
88	2.4.3.2	Element <AuthorityBinding>.....	20
89	2.4.4	Element <AttributeStatement>.....	21
90	2.4.4.1	Elements <AttributeDesignator> and <Attribute>.....	21
91	2.4.4.1.1	Element <AttributeValue>.....	22
92	2.4.5	Element <AuthorizationDecisionStatement>.....	22
93	2.4.5.1	Element <Action>.....	23
94	2.4.5.2	Element <Evidence>.....	24
95	3	SAML Protocol.....	25
96	3.1	Schema Header and Namespace Declarations.....	25
97	3.2	Requests.....	25

98	3.2.1 Complex Type RequestAbstractType .....	26
99	3.2.1.1 Element <RespondWith> .....	26
100	3.2.2 Element <Request> .....	27
101	3.2.2.1 Requests for Assertions by Reference .....	28
102	3.2.2.2 Element <AssertionArtifact> .....	28
103	3.3 Queries .....	28
104	3.3.1 Element <Query> .....	28
105	3.3.2 Element <SubjectQuery> .....	28
106	3.3.3 Element <AuthenticationQuery> .....	28
107	3.3.4 Element <AttributeQuery> .....	29
108	3.3.5 Element <AuthorizationDecisionQuery> .....	30
109	3.4 Responses .....	31
110	3.4.1 Complex Type ResponseAbstractType .....	31
111	3.4.2 Element <Response> .....	32
112	3.4.3 Element <Status> .....	33
113	3.4.3.1 Element <StatusCode> .....	33
114	3.4.3.2 Element <StatusMessage> .....	34
115	3.4.3.3 Element <StatusDetail> .....	35
116	3.4.4 Responses to Queries .....	35
117	4 SAML Versioning .....	36
118	4.1 SAML Specification Set Version .....	36
119	4.1.1 Schema Version .....	36
120	4.1.2 SAML Assertion Version .....	36
121	4.1.3 SAML Protocol Version .....	37
122	4.1.3.1 Request Version .....	37
123	4.1.4 Response Version .....	37
124	4.1.5 Permissible Version Combinations .....	37
125	4.2 SAML Namespace Version .....	38
126	4.2.1 Schema Evolution .....	38
127	5 SAML and XML Signature Syntax and Processing .....	39
128	5.1 Signing Assertions .....	39
129	5.2 Request/Response Signing .....	40
130	5.3 Signature Inheritance .....	40
131	5.4 XML Signature Profile .....	40
132	5.4.1 Signing Formats and Algorithms .....	40
133	5.4.2 References .....	40
134	5.4.3 Canonicalization Method .....	40
135	5.4.4 Transforms .....	41
136	5.4.5 KeyInfo .....	41
137	5.4.6 Binding Between Statements in a Multi-Statement Assertion .....	41
138	5.4.7 Interoperability with SAML V1.0 .....	41
139	5.4.8 Example .....	41
140	6 SAML Extensions .....	44
141	6.1 Assertion Schema Extension .....	44
142	6.2 Protocol Schema Extension .....	44

143	6.3 Use of Type Derivation and Substitution Groups .....	45
144	7 SAML-Defined Identifiers .....	46
145	7.1 Authentication Method Identifiers .....	46
146	7.1.1 Password.....	46
147	7.1.2 Kerberos .....	46
148	7.1.3 Secure Remote Password (SRP).....	46
149	7.1.4 Hardware Token.....	47
150	7.1.5 SSL/TLS Certificate Based Client Authentication: .....	47
151	7.1.6 X.509 Public Key .....	47
152	7.1.7 PGP Public Key .....	47
153	7.1.8 SPKI Public Key .....	47
154	7.1.9 XKMS Public Key .....	47
155	7.1.10 XML Digital Signature.....	47
156	7.1.11 Unspecified.....	47
157	7.2 Action Namespace Identifiers .....	47
158	7.2.1 Read/Write/Execute/Delete/Control .....	48
159	7.2.2 Read/Write/Execute/Delete/Control with Negation .....	48
160	7.2.3 Get/Head/Put/Post .....	48
161	7.2.4 UNIX File Permissions .....	48
162	7.3 NameIdentifier Format Identifiers .....	49
163	7.3.1 Unspecified.....	49
164	7.3.2 Email Address .....	49
165	7.3.3 X.509 Subject Name .....	49
166	7.3.4 Windows Domain Qualified Name.....	49
167	8 References .....	50
168	Appendix A. Acknowledgments .....	52
169	Appendix B. Notices.....	53
170		

---

# 171 1 Introduction

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

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

## 179 1.1 Notation

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

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

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

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

190         Listings of SAML schemas appear like this.

191         Example code listings appear like this.

193 In cases of disagreement between the SAML schema files [**SAML-XSD**] [**SAML-P-XSD**] and this  
194 specification, the schema files take precedence.

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

- 198 • The prefix `saml:` stands for the SAML assertion namespace.
- 199 • The prefix `samlp:` stands for the SAML request-response protocol namespace.
- 200 • The prefix `ds:` stands for the W3C XML Signature namespace [**XMLSig-XSD**].
- 201 • The prefix `xsd:` stands for the W3C XML Schema namespace [**Schema1**] in example listings. In  
202 schema listings, this is the default namespace and no prefix is shown.

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

## 205 1.2 Schema Organization and Namespaces

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

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

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

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

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

214 <http://www.w3.org/2000/09/xmlsig#>

215 See Section 4.2 for information on SAML namespace versioning.

## 216 1.2.1 String and URI Values

217 All SAML string and URI reference values have the types **xsd:string** and **xsd:anyURI** respectively, which  
218 are built in to the W3C XML Schema Datatypes specification [**Schema2**]. All strings in SAML messages  
219 MUST consist of at least one non-whitespace character (whitespace is defined in the XML  
220 Recommendation [**XML**] §2.3). Empty and whitespace-only values are disallowed. Also, unless otherwise  
221 indicated in this specification, all URI reference values MUST consist of at least one non-whitespace  
222 character, and are strongly RECOMMENDED to be absolute [**RFC 2396**].

## 223 1.2.2 Time Values

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

226 SAML system entities SHOULD NOT rely on other applications supporting time resolution finer than  
227 milliseconds. Implementations MUST NOT generate time instants that specify leap seconds.

## 228 1.2.3 ID and ID Reference Values

229 The **xsd:ID** simple type is used to declare SAML identifiers for assertions, requests, and responses.  
230 Values declared to be of type **xsd:ID** in this specification MUST satisfy the following properties:

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

235 The mechanism by which a SAML system entity ensures that the identifier is unique is left to the  
236 implementation. In the case that a pseudorandom technique is employed, the probability of two randomly  
237 chosen identifiers being identical MUST be less than  $2^{-128}$  and SHOULD be less than  $2^{-160}$ . This  
238 requirement MAY be met by encoding a randomly chosen value between 128 and 160 bits in length. The  
239 encoding must conform to the rules defining the **xsd:ID** datatype.

240 The **xsd:NCName** simple type is used in SAML to reference identifiers of type **xsd:ID**. Note that  
241 **xsd>IDREF** cannot be used for this purpose since, in SAML, the element referred to by a SAML reference  
242 identifier might actually be defined in a document separate from that in which the identifier reference is  
243 used. XML [**XML**] requires that names of type **xsd>IDREF** must match the value of an ID attribute on  
244 some element in the same XML document.

## 245 1.2.4 Comparing SAML Values

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

252 If an implementation is comparing values that are represented using different character encodings, the  
253 implementation MUST use a comparison method that returns the same result as converting both values  
254 to the Unicode character encoding, Normalization Form C [**UNICODE-C**], and then performing an exact  
255 binary comparison. This requirement is intended to conform to the W3C Character Model for the World  
256 Wide Web [**W3C-CharMod**], and in particular the rules for Unicode-normalized Text.

257 Applications that compare data received in SAML documents to data from external sources MUST take  
258 into account the normalization rules specified for XML. Text contained within elements is normalized so  
259 that line endings are represented using linefeed characters (ASCII code 10<sub>Decimal</sub>), as described in the  
260 XML Recommendation [XML] §2.11. Attribute values defined as strings (or types derived from strings)  
261 are normalized as described in [XML] §3.3.3. All white space characters are replaced with blanks (ASCII  
262 code 32<sub>Decimal</sub>).

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

## 266 **1.3 SAML Concepts (Non-Normative)**

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

### 269 **1.3.1 Overview**

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

275 Assertions can convey information about authentication acts that were previously performed by subjects,  
276 attributes of subjects, and authorization decisions about whether subjects are allowed to access certain  
277 resources. A single assertion might contain several different internal statements about authentication,  
278 authorization, and attributes.

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

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

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

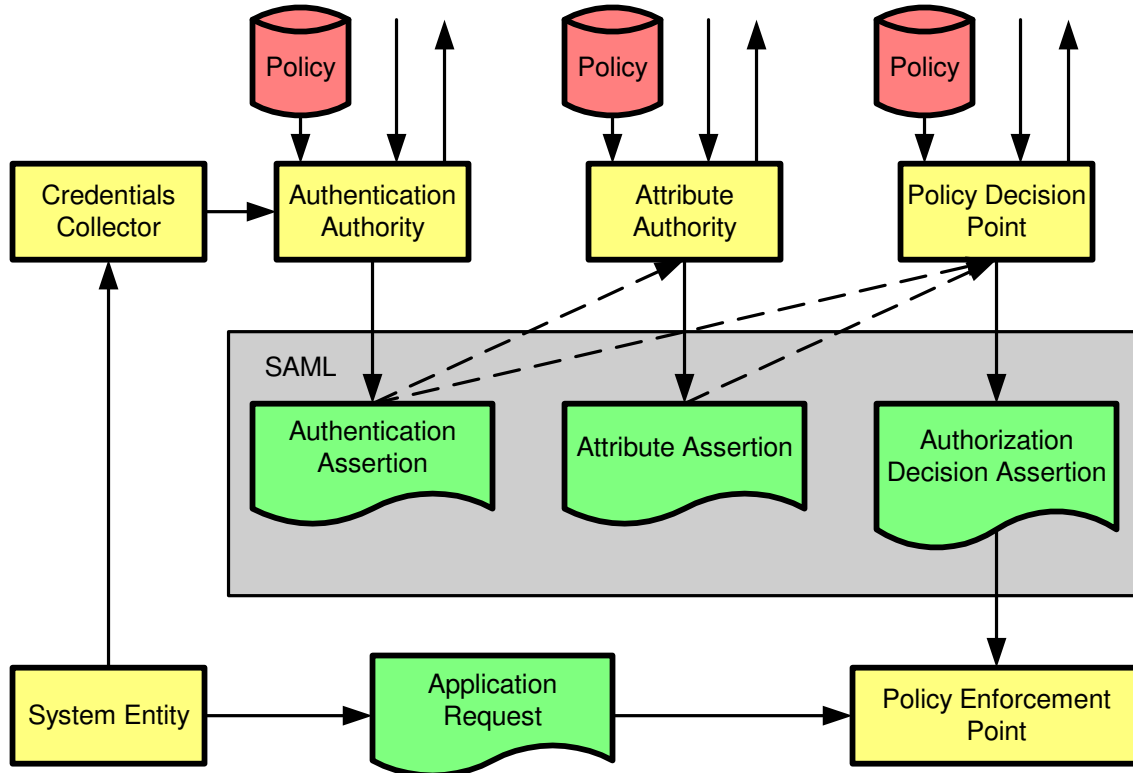


Figure 1 The SAML Domain Model

289  
290

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

295 The assertion and protocol data formats are defined in this specification. The bindings and profiles are  
296 defined in a separate specification [SAMLBind]. A conformance program for SAML is defined in the  
297 conformance specification [SAMLConform]. Security issues are discussed in a separate security and  
298 privacy considerations specification [SAMLSecure].

### 299 1.3.2 SAML and URI-Based Identifiers

300 SAML defines some identifiers to manage references to well-known concepts and sets of values. For  
301 example, the SAML-defined identifier for the password authentication method is as follows:

302 `urn:oasis:names:tc:SAML:1.0:am:password`

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

305 `urn:oasis:names:tc:SAML:1.0:action:rwedc`

306 These identifiers are defined as Uniform Resource Identifier (URI) references, but they are not  
307 necessarily able to be resolved to some Web resource. At times, SAML authorities need to use identifier  
308 strings of their own design, for example to define additional kinds of authentication methods not covered  
309 by SAML-defined identifiers. In the case where a form is used that is compatible with interpretation as a  
310 URI reference, it is not required to be resolvable to some Web resource. However, using URI references  
311 – particularly URLs based on the `http:` scheme or URNs based on the `urn:` scheme – is likely to  
312 mitigate problems with clashing identifiers to some extent.

313 The Read/Write/Execute/Delete/Control identifier above is an example of a namespace (not in the sense  
314 of an XML namespace). SAML uses this namespace mechanism to manage the universe of possible  
315 types of actions and possible names of attributes.  
316 See Section 7 for a list of SAML-defined identifiers.

### 317 **1.3.3 SAML and Extensibility**

318 The XML formats for SAML assertions and protocol messages have been designed to be extensible.  
319 Section 6 describes SAML's design for extensibility in more detail.  
320 However, it is possible that the use of extensions will harm interoperability and therefore the use of  
321 extensions should be carefully considered.

---

## 322 2 SAML Assertions

323 An assertion is a package of information that supplies one or more statements made by a SAML  
324 authority. This SAML specification defines three different kinds of assertion statement that can be created  
325 by a SAML authority. As mentioned above and described in Section 6, extensions are permitted by the  
326 SAML assertion schema, allowing user-defined extensions to assertions and SAML statements, as well  
327 as allowing the definition of new kinds of assertion statement. The three kinds of statement defined in this  
328 specification are:

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

333 The outer structure of an assertion is generic, providing information that is common to all of the  
334 statements within it. Within an assertion, a series of inner elements describe the authentication,  
335 authorization decision, attribute, or user-defined statements containing the specifics.

### 336 2.1 Schema Header and Namespace Declarations

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

```
339 <schema  
340   targetNamespace="urn:oasis:names:tc:SAML:1.0:assertion"  
341   xmlns="http://www.w3.org/2001/XMLSchema"  
342   xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"  
343   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
344   elementFormDefault="unqualified"  
345   attributeFormDefault="unqualified"  
346   version="1.1">  
347   <import namespace="http://www.w3.org/2000/09/xmldsig#"  
348     schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-  
349     schema.xsd"/>  
350   <annotation>  
351     <documentation>  
352       Document identifier: sstc-saml-schema-assertion-1.1-cs  
353       Location: http://www.oasis-  
354       open.org/committees/documents.php?wg_abbrev=security  
355       Revision history:  
356       V1.0 (November, 2002):  
357         Initial standard schema.  
358       V1.1 (May, 2003):  
359         * Note that V1.1 of this schema has the same XML  
360         namespace as V1.0.  
361         Rebased ID content directly on XML Schema types  
362         Added DoNotCacheCondition element and  
363         DoNotCacheConditionType  
364     </documentation>  
365   </annotation>  
366   ...  
367 </schema>
```

### 368 2.2 Simple Types

369 The following section(s) define the SAML assertion-related simple types.

## 370 2.2.1 Simple Type DecisionType

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

373 Permit

374 The specified action is permitted.

375 Deny

376 The specified action is denied.

377 Indeterminate

378 The SAML authority cannot determine whether the specified action is permitted or denied.

379 The `Indeterminate` decision value is used in situations where the SAML authority requires the ability to  
380 provide an affirmative statement that it is not able to issue a decision. Additional information as to the  
381 reason for the refusal or inability to provide a decision MAY be returned as `<StatusDetail>` elements.

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

```
383 <simpleType name="DecisionType">  
384   <restriction base="string">  
385     <enumeration value="Permit"/>  
386     <enumeration value="Deny"/>  
387     <enumeration value="Indeterminate"/>  
388   </restriction>  
389 </simpleType>
```

## 390 2.3 Assertions

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

### 392 2.3.1 Element <AssertionIDReference>

393 The `<AssertionIDReference>` element makes a reference to a SAML assertion.

394 The following schema fragment defines the `<AssertionIDReference>` element:

```
395 <element name="AssertionIDReference" type="NCName"/>
```

### 396 2.3.2 Element <Assertion>

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

399 MajorVersion [Required]  
 400 The major version of this assertion. The identifier for the version of SAML defined in this specification  
 401 is 1. SAML versioning is discussed in Section 4.

402 MinorVersion [Required]  
 403 The minor version of this assertion. The identifier for the version of SAML defined in this specification  
 404 is 1. SAML versioning is discussed in Section 4.

405 AssertionID [Required]  
 406 The identifier for this assertion. It is of type **xsd:ID**, and MUST follow the requirements specified in  
 407 Section 1.2.3 for identifier uniqueness.

408 Issuer [Required]  
 409 The SAML authority that created the assertion. The name of the issuer is provided as a string. The  
 410 issuer name SHOULD be unambiguous to the intended relying parties. SAML authorities may use an  
 411 identifier such as a URI reference that is designed to be unambiguous regardless of context.

412 IssueInstant [Required]  
 413 The time instant of issue in UTC, as described in Section 1.2.2.

414 <Conditions> [Optional]  
 415 Conditions that MUST be taken into account in assessing the validity of the assertion.

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

419 <ds:Signature> [Optional]  
 420 An XML Signature that authenticates the assertion, as described in Section 5.

421 One or more of the following statement elements:

422 <Statement>  
 423 A statement defined in an extension schema.

424 <SubjectStatement>  
 425 A subject statement defined in an extension schema.

426 <AuthenticationStatement>  
 427 An authentication statement.

428 <AuthorizationDecisionStatement>  
 429 An authorization decision statement.

430 <AttributeStatement>  
 431 An attribute statement.

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

```

433 <element name="Assertion" type="saml:AssertionType"/>
434 <complexType name="AssertionType">
435   <sequence>
436     <element ref="saml:Conditions" minOccurs="0"/>
437     <element ref="saml:Advice" minOccurs="0"/>
438     <choice maxOccurs="unbounded">
439       <element ref="saml:Statement"/>
440       <element ref="saml:SubjectStatement"/>
441       <element ref="saml:AuthenticationStatement"/>
442       <element ref="saml:AuthorizationDecisionStatement"/>
443       <element ref="saml:AttributeStatement"/>
444     </choice>
445     <element ref="ds:Signature" minOccurs="0"/>
  
```

```

446     </sequence>
447     <attribute name="MajorVersion" type="integer" use="required"/>
448     <attribute name="MinorVersion" type="integer" use="required"/>
449     <attribute name="AssertionID" type="ID" use="required"/>
450     <attribute name="Issuer" type="string" use="required"/>
451     <attribute name="IssueInstant" type="dateTime" use="required"/>
452 </complexType>

```

### 453 2.3.2.1 Element <Conditions>

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

455 NotBefore [Optional]

456 Specifies the earliest time instant at which the assertion is valid. The time value is encoded in UTC as  
457 described in Section 1.2.2.

458 NotOnOrAfter [Optional]

459 Specifies the time instant at which the assertion has expired. The time value is encoded in UTC as  
460 described in Section 1.2.2.

461 <Condition> [Any Number]

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

463 <AudienceRestrictionCondition> [Any Number]

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

465 <DoNotCacheCondition> [Any Number]

466 Specifies that the assertion SHOULD be used immediately and MUST NOT be retained for future use.

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

```

469 <element name="Conditions" type="saml:ConditionsType"/>
470 <complexType name="ConditionsType">
471   <choice minOccurs="0" maxOccurs="unbounded">
472     <element ref="saml:AudienceRestrictionCondition"/>
473     <element ref="saml:DoNotCacheCondition"/>
474     <element ref="saml:Condition"/>
475   </choice>
476   <attribute name="NotBefore" type="dateTime" use="optional"/>
477   <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
478 </complexType>

```

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

- 483 1. If no sub-elements or attributes are supplied in the <Conditions> element, then the assertion is  
484 considered to be **Valid**.
- 485 2. If any sub-element or attribute of the <Conditions> element is determined to be invalid, then the  
486 assertion is **Invalid**.
- 487 3. If any sub-element or attribute of the <Conditions> element cannot be evaluated, then the validity  
488 of the assertion cannot be determined and is deemed to be **Indeterminate**.
- 489 4. If all sub-elements and attributes of the <Conditions> element are determined to be **Valid**, then the  
490 assertion is considered to be **Valid**.

491 The <Conditions> element MAY be extended to contain additional conditions. If an element contained  
492 within a <Conditions> element is encountered that is not understood, the status of the condition cannot

493 be evaluated and the validity status of the assertion MUST be deemed to be **Indeterminate** in  
494 accordance with rule 3 above.  
495 Note that an assertion that has validity status **Valid** may not be trustworthy for reasons such as not being  
496 issued by a trustworthy SAML authority or not being authenticated by a trustworthy means.

### 497 **2.3.2.1.1 Attributes NotBefore and NotOnOrAfter**

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

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

501 If the value for either `NotBefore` or `NotOnOrAfter` is omitted it is considered unspecified. If the  
502 `NotBefore` attribute is unspecified (and if any other conditions that are supplied evaluate to **Valid**), the  
503 assertion is valid at any time before the time instant specified by the `NotOnOrAfter` attribute. If the  
504 `NotOnOrAfter` attribute is unspecified (and if any other conditions that are supplied evaluate to **Valid**),  
505 the assertion is valid from the time instant specified by the `NotBefore` attribute with no expiry. If neither  
506 attribute is specified (and if any other conditions that are supplied evaluate to **Valid**), the assertion is valid  
507 at any time.

508 The `NotBefore` and `NotOnOrAfter` attributes are defined to have the **dateTime** simple type that is built  
509 in to the W3C XML Schema Datatypes specification [**Schema2**]. All time instants are specified in  
510 Universal Coordinated Time (UTC) as described in Section 1.2.2. Implementations MUST NOT generate  
511 time instants that specify leap seconds.

### 512 **2.3.2.1.2 Element <Condition>**

513 The `<Condition>` element serves as an extension point for new conditions. Its **ConditionAbstractType**  
514 complex type is abstract and is thus usable only as the base of a derived type.

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

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

### 519 **2.3.2.1.3 Elements <AudienceRestrictionCondition> and <Audience>**

520 The `<AudienceRestrictionCondition>` element specifies that the assertion is addressed to one or  
521 more specific audiences identified by `<Audience>` elements. Although a SAML relying party that is  
522 outside the audiences specified is capable of drawing conclusions from an assertion, the SAML authority  
523 explicitly makes no representation as to accuracy or trustworthiness to such a party. It contains the  
524 following elements:

525 `<Audience>`

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

528 The audience restriction condition evaluates to **Valid** if and only if the SAML relying party is a member of  
529 one or more of the audiences specified.

530 The SAML authority cannot prevent a party to whom the assertion is disclosed from taking action on the  
531 basis of the information provided. However, the `<AudienceRestrictionCondition>` element allows  
532 the SAML authority to state explicitly that no warranty is provided to such a party in a machine- and  
533 human-readable form. While there can be no guarantee that a court would uphold such a warranty  
534 exclusion in every circumstance, the probability of upholding the warranty exclusion is considerably  
535 improved.

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

```

538 <element name="AudienceRestrictionCondition"
539     type="saml:AudienceRestrictionConditionType"/>
540 <complexType name="AudienceRestrictionConditionType">
541     <complexContent>
542         <extension base="saml:ConditionAbstractType">
543             <sequence>
544                 <element ref="saml:Audience" maxOccurs="unbounded"/>
545             </sequence>
546         </extension>
547     </complexContent>
548 </complexType>
549 <element name="Audience" type="anyURI"/>

```

#### 550 2.3.2.1.4 Element <DoNotCacheCondition>

551 Indicates that the assertion SHOULD be used immediately by the relying party and MUST NOT be  
552 retained for future use. A SAML authority SHOULD NOT include more than one  
553 <DoNotCacheCondition> element within a <Conditions> element of an assertion. Note that no  
554 Relying Party implementation is required to perform caching. However, any that do so MUST observe this  
555 condition. If multiple <DoNotCacheCondition> elements appear within a <Conditions> element, a  
556 Relying Party MUST treat the multiple elements as though a single <DoNotCacheCondition> element  
557 was specified. For the purposes of determining the validity of the <Conditions> element, the  
558 <DoNotCacheCondition> (see Section 2.3.2.1) is considered to always be valid.

559

```

560 <element name="DoNotCacheCondition" type="saml:DoNotCacheConditionType" />
561 <complexType name="DoNotCacheConditionType">
562     <complexContent>
563         <extension base="saml:ConditionAbstractType"/>
564     </complexContent>
565 </complexType>

```

#### 566 2.3.2.2 Element <Advice>

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

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

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

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

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

```

579 <element name="Advice" type="saml:AdviceType"/>
580 <complexType name="AdviceType">
581     <choice minOccurs="0" maxOccurs="unbounded">
582         <element ref="saml:AssertionIDReference"/>
583         <element ref="saml:Assertion"/>
584         <any namespace="##other" processContents="lax"/>
585     </choice>
586 </complexType>

```

## 587 2.4 Statements

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

### 589 2.4.1 Element <Statement>

590 The <Statement> element is an extension point that allows other assertion-based applications to reuse  
591 the SAML assertion framework. Its **StatementAbstractType** complex type is abstract and is thus usable  
592 only as the base of a derived type.

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

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

### 597 2.4.2 Element <SubjectStatement>

598 The <SubjectStatement> element is an extension point that allows other assertion-based applications  
599 to reuse the SAML assertion framework. It contains a <Subject> element that allows a SAML authority  
600 to describe a subject. Its **SubjectStatementAbstractType** complex type, which extends  
601 **StatementAbstractType**, is abstract and is thus usable only as the base of a derived type.

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

```
604 <element name="SubjectStatement" type="saml:SubjectStatementAbstractType"/>  
605 <complexType name="SubjectStatementAbstractType" abstract="true">  
606   <complexContent>  
607     <extension base="saml:StatementAbstractType">  
608       <sequence>  
609         <element ref="saml:Subject"/>  
610       </sequence>  
611     </extension>  
612   </complexContent>  
613 </complexType>
```

#### 614 2.4.2.1 Element <Subject>

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

617 <NameIdentifier>

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

619 <SubjectConfirmation>

620 Information that allows the subject to be authenticated.

621 If the <Subject> element contains both a <NameIdentifier> and a <SubjectConfirmation>, the  
622 SAML authority is asserting that if the SAML relying party performs the specified

623 <SubjectConfirmation>, it can be confident that the entity presenting the assertion to the relying  
624 party is the entity that the SAML authority associates with the <NameIdentifier>. A <Subject>  
625 element SHOULD NOT identify more than one principal.

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

```
627 <element name="Subject" type="saml:SubjectType"/>  
628 <complexType name="SubjectType">  
629   <choice>  
630     <sequence>  
631       <element ref="saml:NameIdentifier"/>
```

```
632         <element ref="saml:SubjectConfirmation" minOccurs="0"/>
633     </sequence>
634     <element ref="saml:SubjectConfirmation"/>
635 </choice>
636 </complexType>
```

### 637 2.4.2.2 Element <NameIdentifier>

638 The <NameIdentifier> element specifies a subject by a combination of a name qualifier, a name,  
639 and a format. The name is provided as element content. The <NameIdentifier> element has the  
640 following attributes:

641 NameQualifier [Optional]

642 The security or administrative domain that qualifies the name of the subject. This attribute provides a  
643 means to federate names from disparate user stores without collision.

644 Format [Optional]

645 A URI reference representing the format in which the <NameIdentifier> information is provided.  
646 See Section 7.3 for some URI references that MAY be used as the value of the Format attribute. If  
647 the Format attribute is not included, the identifier urn:oasis:names:tc:SAML:1.0:nameid-  
648 format:unspecified (see Section 7.3.1) is in effect. Regardless of format, issues of anonymity,  
649 pseudonymity, and the persistence of the identifier with respect to the asserting and relying parties  
650 are implementation-specific.

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

```
653 <element name="NameIdentifier" type="saml:NameIdentifierType"/>
654 <complexType name="NameIdentifierType">
655     <simpleContent>
656         <extension base="string">
657             <attribute name="NameQualifier" type="string" use="optional"/>
658             <attribute name="Format" type="anyURI" use="optional"/>
659         </extension>
660     </simpleContent>
661 </complexType>
```

662 When a Format other than those specified in Section 7.3 is used, the NameQualifier attribute and the  
663 <NameIdentifier> element's content are to be interpreted according to the specification of that format  
664 as defined outside of this specification.

### 665 2.4.2.3 Elements <SubjectConfirmation>, <ConfirmationMethod>, and 666 <SubjectConfirmationData>

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

669 <ConfirmationMethod> [One or more]  
 670 A URI reference that identifies a protocol to be used to authenticate the subject. URI references  
 671 identifying SAML-defined confirmation methods are currently defined with the SAML profiles in the  
 672 SAML bindings and profiles specification [**SAMLBind**]. Additional methods may be added by defining  
 673 new profiles or by private agreement.  
 674 <SubjectConfirmationData> [Optional]  
 675 Additional authentication information to be used by a specific authentication protocol.  
 676 <ds:KeyInfo> [Optional]  
 677 An XML Signature [**XMLSig**] element that provides access to a cryptographic key held by the subject.  
 678 The following schema fragment defines the <SubjectConfirmation> element and its  
 679 **SubjectConfirmationType** complex type, along with the <SubjectConfirmationData> element and  
 680 the <ConfirmationMethod> element:

```

681 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>
682 <complexType name="SubjectConfirmationType">
683   <sequence>
684     <element ref="saml:ConfirmationMethod" maxOccurs="unbounded"/>
685     <element ref="saml:SubjectConfirmationData" minOccurs="0"/>
686     <element ref="ds:KeyInfo" minOccurs="0"/>
687   </sequence>
688 </complexType>
689 <element name="SubjectConfirmationData" type="anyType"/>
690 <element name="ConfirmationMethod" type="anyURI"/>

```

### 691 2.4.3 Element <AuthenticationStatement>

692 The <AuthenticationStatement> element describes a statement by the SAML authority asserting  
 693 that the statement's subject was authenticated by a particular means at a particular time. It is of type  
 694 **AuthenticationStatementType**, which extends **SubjectStatementAbstractType** with the addition of the  
 695 following elements and attributes:

696 AuthenticationMethod [Required]  
 697 A URI reference that specifies the type of authentication that took place. URI references identifying  
 698 common authentication protocols are listed in Section 7.1.  
 699 AuthenticationInstant [Required]  
 700 Specifies the time at which the authentication took place. The time value is encoded in UTC as  
 701 described in Section 1.2.2.  
 702 <SubjectLocality> [Optional]  
 703 Specifies the DNS domain name and IP address for the system entity from which the subject was  
 704 apparently authenticated.  
 705 <AuthorityBinding> [Any Number]  
 706 Indicates that additional information about the subject of the statement may be available.  
 707 The following schema fragment defines the <AuthenticationStatement> element and its  
 708 **AuthenticationStatementType** complex type:

```

709 <element name="AuthenticationStatement"
710         type="saml:AuthenticationStatementType"/>
711 <complexType name="AuthenticationStatementType">
712   <complexContent>
713     <extension base="saml:SubjectStatementAbstractType">
714       <sequence>
715         <element ref="saml:SubjectLocality" minOccurs="0"/>

```

```

716         <element ref="saml:AuthorityBinding"
717             minOccurs="0" maxOccurs="unbounded"/>
718     </sequence>
719     <attribute name="AuthenticationMethod" type="anyURI"
720 use="required"/>
721     <attribute name="AuthenticationInstant" type="dateTime"
722 use="required"/>
723 </extension>
724 </complexContent>
725 </complexType>

```

### 726 2.4.3.1 Element <SubjectLocality>

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

729 IPAddress [Optional]

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

731 DNSAddress [Optional]

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

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

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

```

737 <element name="SubjectLocality"
738     type="saml: SubjectLocalityType"/>
739 <complexType name="SubjectLocalityType">
740     <attribute name="IPAddress" type="string" use="optional"/>
741     <attribute name="DNSAddress" type="string" use="optional"/>
742 </complexType>

```

### 743 2.4.3.2 Element <AuthorityBinding>

744 The <AuthorityBinding> element MAY be used to indicate to a SAML relying party processing an  
745 AuthenticationStatement that a SAML authority may be available to provide additional information about  
746 the subject of the statement. A single SAML authority may advertise its presence over multiple protocol  
747 bindings, at multiple locations, and as more than one kind of authority by sending multiple elements as  
748 needed.

749 NOTE: This element is deprecated; use of this element SHOULD be avoided because it is planned to be  
750 removed in the next major version of SAML.

751 The <AuthorityBinding> element has the following attributes:

752 AuthorityKind [Required]

753 The type of SAML protocol queries to which the authority described by this element will respond. The  
754 value is specified as an XML Schema QName. The AuthorityKind value is either the QName of the  
755 desired SAML protocol query element or, in the case of an extension schema, the QName of the  
756 SAML **QueryAbstractType** complex type or some extension type that was derived from it. In the  
757 case of an extension schema, the authority will respond to all query elements of the specified type.

758 For example, an attribute authority would be identified by

759 AuthorityKind="samlp:AttributeQuery", where there is a namespace declaration in the  
760 scope of this attribute that binds the samlp: prefix to the SAML protocol namespace.

761 Location [Required]

762 A URI reference describing how to locate and communicate with the authority, the exact syntax of

763 which depends on the protocol binding in use. For example, a binding based on HTTP will be a web  
764 URL, while a binding based on SMTP might use the `mailto:` scheme.

765 Binding [Required]

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

768 The following schema fragment defines the `<AuthorityBinding>` element and its  
769 **AuthorityBindingType** complex type:

```
770 <element name="AuthorityBinding" type="saml:AuthorityBindingType"/>  
771 <complexType name="AuthorityBindingType">  
772   <attribute name="AuthorityKind" type="QName" use="required"/>  
773   <attribute name="Location" type="anyURI" use="required"/>  
774   <attribute name="Binding" type="anyURI" use="required"/>  
775 </complexType>
```

## 776 2.4.4 Element `<AttributeStatement>`

777 The `<AttributeStatement>` element describes a statement by the SAML authority asserting that the  
778 statement's subject is associated with the specified attributes. It is of type **AttributeStatementType**,  
779 which extends **SubjectStatementAbstractType** with the addition of the following element:

780 `<Attribute>` [One or More]

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

782 The following schema fragment defines the `<AttributeStatement>` element and its  
783 **AttributeStatementType** complex type:

```
784 <element name="AttributeStatement" type="saml:AttributeStatementType"/>  
785 <complexType name="AttributeStatementType">  
786   <complexContent>  
787     <extension base="saml:SubjectStatementAbstractType">  
788       <sequence>  
789         <element ref="saml:Attribute" maxOccurs="unbounded"/>  
790       </sequence>  
791     </extension>  
792   </complexContent>  
793 </complexType>
```

### 794 2.4.4.1 Elements `<AttributeDesignator>` and `<Attribute>`

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

799 AttributeNamespace [Required]

800 The namespace in which the `AttributeName` elements are interpreted.

801 AttributeName [Required]

802 The name of the attribute.

803 The following schema fragment defines the `<AttributeDesignator>` element and its  
804 **AttributeDesignatorType** complex type:

```
805 <element name="AttributeDesignator" type="saml:AttributeDesignatorType"/>  
806 <complexType name="AttributeDesignatorType">  
807   <attribute name="AttributeName" type="string" use="required"/>  
808   <attribute name="AttributeNamespace" type="anyURI" use="required"/>  
809 </complexType>
```

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

813 <AttributeValue> [Any Number]

814 The value of the attribute.

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

```
816 <element name="Attribute" type="saml:AttributeType"/>
817 <complexType name="AttributeType">
818   <complexContent>
819     <extension base="saml:AttributeDesignatorType">
820       <sequence>
821         <element ref="saml:AttributeValue"
822           maxOccurs="unbounded"/>
823       </sequence>
824     </extension>
825   </complexContent>
826 </complexType>
```

#### 827 2.4.4.1.1 Element <AttributeValue>

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

830 If the data content of an AttributeValue element is of an XML Schema simple type (such as **xsd:integer**  
831 or **xsd:string**), the data type MAY be declared explicitly by means of an `xsi:type` declaration in the  
832 <AttributeValue> element. If the attribute value contains structured data, the necessary data  
833 elements MAY be defined in an extension schema.

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

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

#### 836 2.4.5 Element <AuthorizationDecisionStatement>

837 The <AuthorizationDecisionStatement> element describes a statement by the SAML authority  
838 asserting that a request for access by the statement's subject to the specified resource has resulted in the  
839 specified authorization decision on the basis of some optionally specified evidence.

840 The resource is identified by means of a URI reference. In order for the assertion to be interpreted  
841 correctly and securely, the SAML authority and SAML relying party MUST interpret each URI reference in  
842 a consistent manner. Failure to achieve a consistent URI reference interpretation can result in different  
843 authorization decisions depending on the encoding of the resource URI reference. Rules for normalizing  
844 URI references are to be found in IETF RFC 2396 [RFC 2396] §6:

845 In general, the rules for equivalence and definition of a normal form, if any, are scheme  
846 dependent. When a scheme uses elements of the common syntax, it will also use the common  
847 syntax equivalence rules, namely that the scheme and hostname are case insensitive and a URL  
848 with an explicit ".port", where the port is the default for the scheme, is equivalent to one where  
849 the port is elided.

850 To avoid ambiguity resulting from variations in URI encoding SAML system entities SHOULD employ the  
851 URI normalized form wherever possible as follows:

- 852 • SAML authorities SHOULD encode all resource URI references in normalized form.
- 853 • Relying parties SHOULD convert resource URI references to normalized form prior to processing.

854 Inconsistent URI reference interpretation can also result from differences between the URI reference  
855 syntax and the semantics of an underlying file system. Particular care is required if URI references are

856 employed to specify an access control policy language. The following security conditions should be  
857 satisfied by the system which employs SAML assertions:

- 858 • Parts of the URI reference syntax are case sensitive. If the underlying file system is case insensitive,  
859 a requester SHOULD NOT be able to gain access to a denied resource by changing the case of a  
860 part of the resource URI reference.
- 861 • Many file systems support mechanisms such as logical paths and symbolic links, which allow users to  
862 establish logical equivalences between file system entries. A requester SHOULD NOT be able to gain  
863 access to a denied resource by creating such an equivalence.

864 The `<AuthorizationDecisionStatement>` element is of type  
865 **AuthorizationDecisionStatementType**, which extends **SubjectStatementAbstractType** with the  
866 addition of the following elements (in order) and attributes:

867 **Resource** [Required]

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

871 **Decision** [Required]

872 The decision rendered by the SAML authority with respect to the specified resource. The value is of  
873 the **DecisionType** simple type.

874 **<Action>** [One or more]

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

876 **<Evidence>** [Optional]

877 A set of assertions that the SAML authority relied on in making the decision.

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

```
880 <element name="AuthorizationDecisionStatement"  
881 type="saml:AuthorizationDecisionStatementType"/>  
882 <complexType name="AuthorizationDecisionStatementType">  
883   <complexContent>  
884     <extension base="saml:SubjectStatementAbstractType">  
885       <sequence>  
886         <element ref="saml:Action" maxOccurs="unbounded"/>  
887         <element ref="saml:Evidence" minOccurs="0"/>  
888       </sequence>  
889       <attribute name="Resource" type="anyURI" use="required"/>  
890       <attribute name="Decision" type="saml:DecisionType"  
891 use="required"/>  
892     </extension>  
893   </complexContent>  
894 </complexType>
```

### 895 2.4.5.1 Element `<Action>`

896 The `<Action>` element specifies an action on the specified resource for which permission is sought. It  
897 has the following attribute and string-data content:

898 Namespace [Optional]

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

902 *string data* [Required]

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

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

```
905 <element name="Action" type="saml:ActionType"/>  
906 <complexType name="ActionType">  
907   <simpleContent>  
908     <extension base="string">  
909       <attribute name="Namespace" type="anyURI"/>  
910     </extension>  
911   </simpleContent>  
912 </complexType>
```

### 913 2.4.5.2 Element <Evidence>

914 The <Evidence> element contains an assertion or assertion reference that the SAML authority relied on  
915 in issuing the authorization decision. It has the **EvidenceType** complex type. It contains one of the  
916 following elements:

917 <AssertionIDReference>

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

919 <Assertion>

920 Specifies an assertion by value.

921 Providing an assertion as evidence MAY affect the reliance agreement between the SAML relying party  
922 and the SAML authority making the authorization decision. For example, in the case that the SAML  
923 relying party presented an assertion to the SAML authority in a request, the SAML authority MAY use that  
924 assertion as evidence in making its authorization decision without endorsing the <Evidence> element's  
925 assertion as valid either to the relying party or any other third party.

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

```
927 <element name="Evidence" type="saml:EvidenceType"/>  
928 <complexType name="EvidenceType">  
929   <choice maxOccurs="unbounded">  
930     <element ref="saml:AssertionIDReference"/>  
931     <element ref="saml:Assertion"/>  
932   </choice>  
933 </complexType>
```

934

## 3 SAML Protocol

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

938 SAML-aware requesters MAY in addition use the SAML request-response protocol defined by the  
939 <Request> and <Response> elements. The requester sends a <Request> element to a SAML  
940 responder, and the responder generates a <Response> element, as shown in Figure 2.



941

942

Figure 2: SAML Request-Response Protocol

943

### 3.1 Schema Header and Namespace Declarations

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

```
946 <schema  
947   targetNamespace="urn:oasis:names:tc:SAML:1.0:protocol"  
948   xmlns="http://www.w3.org/2001/XMLSchema"  
949   xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"  
950   xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"  
951   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
952   elementFormDefault="unqualified"  
953   attributeFormDefault="unqualified"  
954   version="1.1">  
955   <import namespace="urn:oasis:names:tc:SAML:1.0:assertion"  
956     schemaLocation="sstc-saml-schema-assertion-1.1-cs.xsd"/>  
957   <import namespace="http://www.w3.org/2000/09/xmldsig#"  
958     schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-  
959 schema.xsd "/>  
960   <annotation>  
961     <documentation>  
962       Document identifier: sstc-saml-schema-protocol-1.1-cs  
963       Location: http://www.oasis-  
964 open.org/committees/documents.php?wg_abbrev=security  
965       Revision history:  
966       V1.0 (November, 2002):  
967         Initial standard schema.  
968       V1.1 (May, 2003):  
969         * Note that V1.1 of this schema has the same XML  
970         namespace as V1.0.  
971         Rebased ID content directly on XML Schema types  
972     </documentation>  
973   </annotation>  
974   ...  
975 </schema>
```

976

### 3.2 Requests

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

## 978 **3.2.1 Complex Type RequestAbstractType**

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

981 **RequestID** [Required]

982 An identifier for the request. It is of type **xsd:ID** and MUST follow the requirements specified in  
983 Section 1.2.3 for identifier uniqueness. The values of the **RequestID** attribute in a request and the  
984 **InResponseTo** attribute in the corresponding response MUST match.

985 **MajorVersion** [Required]

986 The major version of this request. The identifier for the version of SAML defined in this specification is  
987 1. SAML versioning is discussed in Section 4.

988 **MinorVersion** [Required]

989 The minor version of this request. The identifier for the version of SAML defined in this specification is  
990 1. SAML versioning is discussed in Section 4.

991 **IssueInstant** [Required]

992 The time instant of issue of the request. The time value is encoded in UTC as described in Section  
993 1.2.2.

994 **<RespondWith>** [Any Number]

995 Each **<RespondWith>** element specifies a type of response that is acceptable to the requester.

996 **<ds:Signature>** [Optional]

997 An XML Signature that authenticates the request, as described in Section 5.

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

```
999 <complexType name="RequestAbstractType" abstract="true">  
1000 <sequence>  
1001 <element ref="samlp:RespondWith"  
1002 minOccurs="0" maxOccurs="unbounded"/>  
1003 <element ref="ds:Signature" minOccurs="0"/>  
1004 </sequence>  
1005 <attribute name="RequestID" type="ID" use="required"/>  
1006 <attribute name="MajorVersion" type="integer" use="required"/>  
1007 <attribute name="MinorVersion" type="integer" use="required"/>  
1008 <attribute name="IssueInstant" type="dateTime" use="required"/>  
1009 </complexType>
```

### 1010 **3.2.1.1 Element <RespondWith>**

1011 The **<RespondWith>** element specifies the type of statement the SAML relying party wants from the  
1012 SAML authority. Multiple **<RespondWith>** elements MAY be included to indicate that the relying party  
1013 will accept assertions containing any of the specified types. If no **<RespondWith>** element is given, the  
1014 SAML authority MAY return assertions containing statements of any type.

1015 NOTE: This element is deprecated; use of this element SHOULD be avoided because it is planned to be  
1016 removed in the next major version of SAML.

1017 If the **<Request>** element contains one or more **<RespondWith>** elements, the SAML authority MUST  
1018 NOT respond with assertions containing statements of any type not specified in one of the  
1019 **<RespondWith>** elements.

1020 Inability to find assertions that meet **<RespondWith>** criteria should be treated as identical to any other  
1021 query for which no assertions are available. In both cases a status of success MUST be returned in the  
1022 Response message, but no assertions will be included.

1023 The content of each <RespondWith> element is an XML QName. The <RespondWith> content is  
1024 either the QName of the desired SAML statement element name or, in the case of an extension schema,  
1025 it is the QName of the SAML **StatementAbstractType** complex type or some type that was derived from  
1026 it. In the case of an extension schema, all statements of the specified type are requested.

1027 For example, a relying party that wishes to receive assertions containing only attribute statements would  
1028 specify <RespondWith>saml:AttributeStatement</RespondWith>, where the prefix is bound to  
1029 the SAML assertion namespace in a namespace declaration that is in the scope of this element.

1030 The following schema fragment defines the <RespondWith> element:

```
1031 <element name="RespondWith" type="QName"/>
```

### 1032 3.2.2 Element <Request>

1033 The <Request> element specifies a SAML request. It provides either a query or a request for a specific  
1034 assertion identified by <AssertionIDReference> or <AssertionArtifact>. It has the complex  
1035 type **RequestType**, which extends **RequestAbstractType** by adding a choice of one of the following  
1036 elements:

1037 <Query>

1038 An extension point that allows extension schemas to define new types of query.

1039 <SubjectQuery>

1040 An extension point that allows extension schemas to define new types of query that specify a single  
1041 SAML subject.

1042 <AuthenticationQuery>

1043 Makes a query for authentication information.

1044 <AttributeQuery>

1045 Makes a query for attribute information.

1046 <AuthorizationDecisionQuery>

1047 Makes a query for an authorization decision.

1048 <AssertionIDReference> [One or more]

1049 Requests an assertion by reference to the value of its `AssertionID` attribute.

1050 <AssertionArtifact> [One or more]

1051 Requests assertions by supplying an assertion artifact that represents it.

1052 The following schema fragment defines the <Request> element and its **RequestType** complex type:

```
1053 <element name="Request" type="saml:RequestType"/>  
1054 <complexType name="RequestType">  
1055   <complexContent>  
1056     <extension base="saml:RequestAbstractType">  
1057       <choice>  
1058         <element ref="saml:Query"/>  
1059         <element ref="saml:SubjectQuery"/>  
1060         <element ref="saml:AuthenticationQuery"/>  
1061         <element ref="saml:AttributeQuery"/>  
1062         <element ref="saml:AuthorizationDecisionQuery"/>  
1063         <element ref="saml:AssertionIDReference" maxOccurs="unbounded"/>  
1064         <element ref="saml:AssertionArtifact" maxOccurs="unbounded"/>  
1065       </choice>  
1066     </extension>  
1067   </complexContent>  
1068 </complexType>
```

### 1071 3.2.2.1 Requests for Assertions by Reference

1072 In the context of a <Request> element, the <saml:AssertionIDReference> element is used to  
1073 request an assertion by means of its ID. See Section 2.3.1 for more information on this element.

### 1074 3.2.2.2 Element <AssertionArtifact>

1075 The <AssertionArtifact> element is used to specify the assertion artifact that represents an  
1076 assertion being requested. Its use is governed by the specific profile of SAML that is being used; see the  
1077 SAML specification for bindings and profiles [SAMLBind] for more information on the use of assertion  
1078 artifacts in profiles.

1079 The following schema fragment defines the <AssertionArtifact> element:

```
1080 <element name="AssertionArtifact" type="string"/>
```

## 1081 3.3 Queries

1082 The following sections define the SAML constructs that contain query information.

### 1083 3.3.1 Element <Query>

1084 The <Query> element is an extension point that allows new SAML queries to be defined. Its  
1085 **QueryAbstractType** is abstract and is thus usable only as the base of a derived type.  
1086 **QueryAbstractType** is the base type from which all SAML query elements are derived.

1087 The following schema fragment defines the <Query> element and its **QueryAbstractType** complex type:

```
1088 <element name="Query" type="saml:QueryAbstractType"/>  
1089 <complexType name="QueryAbstractType" abstract="true"/>
```

### 1090 3.3.2 Element <SubjectQuery>

1091 The <SubjectQuery> element is an extension point that allows new SAML queries that specify a single  
1092 SAML subject. Its **SubjectQueryAbstractType** complex type is abstract and is thus usable only as the  
1093 base of a derived type. **SubjectQueryAbstractType** adds the <Subject> element.

1094 The following schema fragment defines the <SubjectQuery> element and its  
1095 **SubjectQueryAbstractType** complex type:

```
1096 <element name="SubjectQuery" type="saml:SubjectQueryAbstractType"/>  
1097 <complexType name="SubjectQueryAbstractType" abstract="true">  
1098   <complexContent>  
1099     <extension base="saml:QueryAbstractType">  
1100       <sequence>  
1101         <element ref="saml:Subject"/>  
1102       </sequence>  
1103     </extension>  
1104   </complexContent>  
1105 </complexType>
```

### 1106 3.3.3 Element <AuthenticationQuery>

1107 The <AuthenticationQuery> element is used to make the query "What assertions containing  
1108 authentication statements are available for this subject?" A successful response will be in the form of  
1109 assertions containing authentication statements.

1110 The <AuthenticationQuery> element MUST NOT be used as a request for a new authentication  
1111 using credentials provided in the request. <AuthenticationQuery> is a request for statements about

1112 authentication acts that have occurred in a previous interaction between the indicated subject and the  
1113 Authentication Authority.

1114 This element is of type **AuthenticationQueryType**, which extends **SubjectQueryAbstractType** with the  
1115 addition of the following attribute:

1116 **AuthenticationMethod** [Optional]

1117 If present, specifies a filter for possible responses. Such a query asks the question “What assertions  
1118 containing authentication statements do you have for this subject with the supplied authentication  
1119 method?”

1120 In response to an authentication query, a SAML authority returns assertions with authentication  
1121 statements as follows:

- 1122 • Rules given in Section 3.4.4 for matching against the <Subject> element of the query identify the  
1123 assertions that may be returned.
- 1124 • If the **AuthenticationMethod** attribute is present in the query, at least one  
1125 <AuthenticationStatement> element in the set of returned assertions **MUST** contain an  
1126 **AuthenticationMethod** attribute that matches the **AuthenticationMethod** attribute in  
1127 the query. It is **OPTIONAL** for the complete set of all such matching assertions to be returned in  
1128 the response.
- 1129 • If any <RespondWith> elements are present and none of them contain  
1130 “saml:AuthenticationStatement”, then the SAML authority returns no assertions with  
1131 authentication statements. (See Section 3.2.1.1 for more information.)

1132 The following schema fragment defines the <AuthenticationQuery> element and its  
1133 **AuthenticationQueryType** complex type:

```
1134 <element name="AuthenticationQuery" type="samlp:AuthenticationQueryType"/>  
1135 <complexType name="AuthenticationQueryType">  
1136   <complexContent>  
1137     <extension base="samlp:SubjectQueryAbstractType">  
1138       <attribute name="AuthenticationMethod" type="anyURI"/>  
1139     </extension>  
1140   </complexContent>  
1141 </complexType>
```

### 1142 3.3.4 Element <AttributeQuery>

1143 The <AttributeQuery> element is used to make the query “Return the requested attributes for this  
1144 subject.” A successful response will be in the form of assertions containing attribute statements. This  
1145 element is of type **AttributeQueryType**, which extends **SubjectQueryAbstractType** with the addition of  
1146 the following element and attribute:

1147 Resource [Optional]

1148 If present, specifies that the attribute query is being made in order to evaluate a specific access  
1149 request relating to the resource. The SAML authority MAY use the resource attribute to establish the  
1150 scope of the request. It is permitted for this attribute to have the value of the empty URI reference (""),  
1151 and the meaning is defined to be "the start of the current document", as specified by [RFC 2396]  
1152 §4.2.

1153 If the resource attribute is specified and the SAML authority does not wish to support resource-  
1154 specific attribute queries, or if the resource value provided is invalid or unrecognized, then the  
1155 Attribute Authority SHOULD respond with a top-level <StatusCode> value of Responder and a  
1156 second-level <StatusCode> value of ResourceNotRecognized.

1157 <AttributeDesignator> [Any Number] (see Section 2.4.4.1)

1158 Each <AttributeDesignator> element specifies an attribute whose value is to be returned. If no  
1159 attributes are specified, it indicates that all attributes allowed by policy are requested.

1160 In response to an attribute query, a SAML authority returns assertions with attribute statements as  
1161 follows:

- 1162 • Rules given in Section 3.4.4 for matching against the <Subject> element of the query identify the  
1163 assertions that may be returned.
- 1164 • If any <AttributeDesignator> elements are present in the query, they constrain the attribute  
1165 values returned, as noted above.
- 1166 • The SAML authority MAY take the Resource attribute into account in further constraining the values  
1167 returned, as noted above.
- 1168 • The attribute values returned MAY be constrained by application-specific policy considerations.
- 1169 • If any <RespondWith> elements are present and none of them contain  
1170 "saml:AttributeStatement", then the SAML authority returns no assertions with attribute  
1171 statements. (See Section 3.2.1.1 for more information.)

1172

1173 The following schema fragment defines the <AttributeQuery> element and its **AttributeQueryType**  
1174 complex type:

```
1175 <element name="AttributeQuery" type="samlp:AttributeQueryType"/>  
1176 <complexType name="AttributeQueryType">  
1177   <complexContent>  
1178     <extension base="samlp:SubjectQueryAbstractType">  
1179       <sequence>  
1180         <element ref="saml:AttributeDesignator"  
1181           minOccurs="0" maxOccurs="unbounded"/>  
1182       </sequence>  
1183       <attribute name="Resource" type="anyURI" use="optional"/>  
1184     </extension>  
1185   </complexContent>  
1186 </complexType>
```

### 1187 3.3.5 Element <AuthorizationDecisionQuery>

1188 The <AuthorizationDecisionQuery> element is used to make the query "Should these actions on  
1189 this resource be allowed for this subject, given this evidence?" A successful response will be in the form  
1190 of assertions containing authorization decision statements. This element is of type  
1191 **AuthorizationDecisionQueryType**, which extends **SubjectQueryAbstractType** with the addition of the  
1192 following elements and attribute:

1193 Resource [Required]  
 1194 A URI reference indicating the resource for which authorization is requested.  
 1195 <Action> [One or More]  
 1196 The actions for which authorization is requested.  
 1197 <Evidence> [Optional]  
 1198 A set of assertions that the SAML authority MAY rely on in making its authorization decision.  
 1199 In response to an authorization decision query, a SAML authority returns assertions with authorization  
 1200 decision statements as follows:

- 1201 • Rules given in Section 3.4.4 for matching against the <Subject> element of the query identify the  
 1202 assertions that may be returned.
- 1203 • If any <RespondWith> elements are present and none of them contain  
 1204 "saml:AuthorizationDecisionStatement", then the SAML authority returns no assertions with  
 1205 authorization decision statements. (See Section 3.2.1.1 for more information.)

1206 The following schema fragment defines the <AuthorizationDecisionQuery> element and its  
 1207 **AuthorizationDecisionQueryType** complex type:

```

1208 <element name="AuthorizationDecisionQuery"
1209 type="samlp:AuthorizationDecisionQueryType"/>
1210 <complexType name="AuthorizationDecisionQueryType">
1211   <complexContent>
1212     <extension base="samlp:SubjectQueryAbstractType">
1213       <sequence>
1214         <element ref="saml:Action" maxOccurs="unbounded"/>
1215         <element ref="saml:Evidence" minOccurs="0"/>
1216       </sequence>
1217       <attribute name="Resource" type="anyURI" use="required"/>
1218     </extension>
1219   </complexContent>
1220 </complexType>
  
```

## 1221 3.4 Responses

1222 The following sections define the SAML constructs that contain response information.

### 1223 3.4.1 Complex Type ResponseAbstractType

1224 All SAML responses are of types that are derived from the abstract **ResponseAbstractType** complex  
 1225 type. This type defines common attributes and elements that are associated with all SAML responses:

1226 ResponseID [Required]  
 1227 An identifier for the response. It is of type **xsd:ID**, and MUST follow the requirements specified in  
 1228 Section 1.2.3 for identifier uniqueness.

1229 InResponseTo [Optional]  
 1230 A reference to the identifier of the request to which the response corresponds, if any. If the response  
 1231 is not generated in response to a request, or if the RequestID attribute value of a request cannot be  
 1232 determined (because the request is malformed), then this attribute MUST NOT be present.  
 1233 Otherwise, it MUST be present and its value MUST match the value of the corresponding  
 1234 RequestID attribute value.

1235 MajorVersion [Required]  
 1236 The major version of this response. The identifier for the version of SAML defined in this specification  
 1237 is 1. SAML versioning is discussed in Section 4.

1238 MinorVersion [Required]  
 1239 The minor version of this response. The identifier for the version of SAML defined in this specification  
 1240 is 1. SAML versioning is discussed in Section 4.

1241 IssueInstant [Required]  
 1242 The time instant of issue of the response. The time value is encoded in UTC as described in Section  
 1243 1.2.2.

1244 Recipient [Optional]  
 1245 The intended recipient of this response. This is useful to prevent malicious forwarding of responses to  
 1246 unintended recipients, a protection that is required by some use profiles. It is set by the generator of  
 1247 the response to a URI reference that identifies the intended recipient. If present, the actual recipient  
 1248 MUST check that the URI reference identifies the recipient or a resource managed by the recipient. If  
 1249 it does not, the response MUST be discarded.

1250 <ds:Signature> [Optional]  
 1251 An XML Signature that authenticates the response, as described in Section 5.  
 1252 The following schema fragment defines the **ResponseAbstractType** complex type:

```

1253 <complexType name="ResponseAbstractType" abstract="true">
1254   <sequence>
1255     <element ref = "ds:Signature" minOccurs="0"/>
1256   </sequence>
1257   <attribute name="ResponseID" type="ID" use="required"/>
1258   <attribute name="InResponseTo" type="NCName" use="optional"/>
1259   <attribute name="MajorVersion" type="integer" use="required"/>
1260   <attribute name="MinorVersion" type="integer" use="required"/>
1261   <attribute name="IssueInstant" type="dateTime" use="required"/>
1262   <attribute name="Recipient" type="anyURI" use="optional"/>
1263 </complexType>
  
```

### 1264 3.4.2 Element <Response>

1265 The <Response> element specifies the status of the corresponding SAML request and a list of zero or  
 1266 more assertions that answer the request. It has the complex type **ResponseType**, which extends  
 1267 **ResponseAbstractType** by adding the following elements in order:

1268 <Status> [Required]  
 1269 A code representing the status of the corresponding request.

1270 <Assertion> [Any Number]  
 1271 Specifies an assertion by value. (See Section 2.3.2 for more information.)

1272 The following schema fragment defines the <Response> element and its **ResponseType** complex type:

```

1273 <element name="Response" type="samlp:ResponseType"/>
1274 <complexType name="ResponseType">
1275   <complexContent>
1276     <extension base="samlp:ResponseAbstractType">
1277       <sequence>
1278         <element ref="samlp:Status"/>
1279         <element ref="saml:Assertion" minOccurs="0"
1280 maxOccurs="unbounded"/>
1281       </sequence>
1282     </extension>
1283   </complexContent>
1284 </complexType>

```

### 1285 3.4.3 Element <Status>

1286 The <Status> element contains the following elements:

1287 <StatusCode> [Required]

1288 A code representing the status of the corresponding request.

1289 <StatusMessage> [Optional]

1290 A message which MAY be returned to an operator.

1291 <StatusDetail> [Optional]

1292 Additional information concerning an error condition.

1293 The following schema fragment defines the <Status> element and its **StatusType** complex type:

```

1294 <element name="Status" type="samlp:StatusType"/>
1295 <complexType name="StatusType">
1296   <sequence>
1297     <element ref="samlp:StatusCode"/>
1298     <element ref="samlp:StatusMessage" minOccurs="0"/>
1299     <element ref="samlp:StatusDetail" minOccurs="0"/>
1300   </sequence>
1301 </complexType>

```

#### 1302 3.4.3.1 Element <StatusCode>

1303 The <StatusCode> element specifies one or more possibly nested, codes representing the status of the  
1304 corresponding request. The <StatusCode> element has the following element and attribute:

1305 Value [Required]

1306 The status code value. This attribute contains an XML Schema QName; a namespace prefix **MUST**  
1307 be provided. The value of the topmost <StatusCode> element **MUST** be from the top-level list  
1308 provided in this section.

1309 <StatusCode> [Optional]

1310 A subordinate status code that provides more specific information on an error condition.

1311 The top-level <StatusCode> values are QNames associated with the SAML protocol namespace. The  
1312 local parts of these QNames are as follows:

1313 Success  
1314     The request succeeded.

1315 VersionMismatch  
1316     The SAML responder could not process the request because the version of the request message was  
1317     incorrect.

1318 Requester  
1319     The request could not be performed due to an error on the part of the requester.

1320 Responder  
1321     The request could not be performed due to an error on the part of the SAML responder or SAML  
1322     authority.

1323 The following second-level status codes are referenced at various places in the specification. Additional  
1324 second-level status codes MAY be defined in future versions of the SAML specification.

1325 RequestVersionTooHigh  
1326     The SAML responder cannot process the request because the protocol version specified in the  
1327     request message is a major upgrade from the highest protocol version supported by the responder.

1328 RequestVersionTooLow  
1329     The SAML responder cannot process the request because the protocol version specified in the  
1330     request message is too low.

1331 RequestVersionDeprecated  
1332     The SAML responder can not process any requests with the protocol version specified in the request.

1333 TooManyResponses  
1334     The response message would contain more elements than the SAML responder will return.

1335 RequestDenied  
1336     The SAML responder or SAML authority is able to process the request but has chosen not to  
1337     respond. This status code MAY be used when there is concern about the security context of the  
1338     request message or the sequence of request messages received from a particular requester.

1339 ResourceNotRecognized  
1340     The SAML authority does not wish to support resource-specific attribute queries, or the resource  
1341     value provided in the request message is invalid or unrecognized.

1342 SAML system entities are free to define more specific status codes in other namespaces, but MUST NOT  
1343 define additional codes in the SAML assertion or protocol namespace.

1344 The QNames defined as status codes SHOULD be used only in the <StatusCode> element's Value  
1345 attribute and have the above semantics only in that context.

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

```
1348 <element name="StatusCode" type="samlp:StatusCodeType"/>  
1349 <complexType name="StatusCodeType">  
1350   <sequence>  
1351     <element ref="samlp:StatusCode" minOccurs="0"/>  
1352   </sequence>  
1353   <attribute name="Value" type="QName" use="required"/>  
1354 </complexType>
```

### 1355 3.4.3.2 Element <StatusMessage>

1356 The <StatusMessage> element specifies a message that MAY be returned to an operator:

1357 The following schema fragment defines the <StatusMessage> element and its **StatusMessageType**  
1358 complex type:

```
1359 <element name="StatusMessage" type="string"/>
```

### 1360 3.4.3.3 Element <StatusDetail>

1361 The <StatusDetail> element MAY be used to specify additional information concerning an error  
1362 condition.

1363 The following schema fragment defines the <StatusDetail> element and its **StatusDetailType**  
1364 complex type:

```
1365 <element name="StatusDetail" type="samlp:StatusDetailType"/>  
1366 <complexType name="StatusDetailType">  
1367 <sequence>  
1368 <any namespace="##any" processContents="lax" minOccurs="0"  
1369 maxOccurs="unbounded"/>  
1370 </sequence>  
1371 </complexType>
```

### 1372 3.4.4 Responses to Queries

1373 In response to a query, every assertion returned by a SAML authority MUST contain at least one  
1374 statement whose <saml:Subject> element **strongly matches** the <saml:Subject> element found in  
1375 the query.

1376 A <saml:Subject> element S1 strongly matches S2 if and only if the following two conditions both  
1377 apply:

- 1378 • If S2 includes a <saml:NameIdentifier> element, then S1 must include an identical  
1379 <saml:NameIdentifier> element.
- 1380 • If S2 includes a <saml:SubjectConfirmation> element, then S1 must include an identical  
1381 <saml:SubjectConfirmation> element.

1382 If the SAML authority cannot provide an assertion with any statements satisfying the constraints  
1383 expressed by a query, the <Response> element MUST NOT contain an <Assertion> element and  
1384 MUST include a <StatusCode> element with value *Success*. It MAY return a <StatusMessage>  
1385 element with additional information.

---

## 1386 4 SAML Versioning

1387 The SAML specification set is versioned in two independent ways. Each is discussed in the following  
1388 sections, along with processing rules for detecting and handling version differences, when applicable.  
1389 Also included are guidelines on when and why specific version information is expected to change in future  
1390 revisions of the specification.

1391 When version information is expressed as both a Major and Minor version, it may be expressed  
1392 discretely, or in the form *Major.Minor*. The version number *Major<sub>B</sub>.Minor<sub>B</sub>* is higher than the version  
1393 number *Major<sub>A</sub>.Minor<sub>A</sub>* if and only if:

1394  $Major_B > Major_A \vee ( ( Major_B = Major_A ) \wedge Minor_B > Minor_A )$

### 1395 4.1 SAML Specification Set Version

1396 Each release of the SAML specification set will contain a major and minor version designation describing  
1397 its relationship to earlier and later versions of the specification set. The version will be expressed in the  
1398 content and filenames of published materials, including the specification set document(s), and XML  
1399 schema instance(s). There are no normative processing rules surrounding specification set versioning,  
1400 since it merely encompasses the collective release of normative specification documents which  
1401 themselves contain processing rules.

1402 The overall size and scope of changes to the specification set document(s) will informally dictate whether  
1403 a set of changes constitutes a major or minor revision. In general, if the specification set is backwards  
1404 compatible with an earlier specification set (that is, valid older messages, protocols, and semantics  
1405 remain valid), then the new version will be a minor revision. Otherwise, the changes will constitute a major  
1406 revision. Note that SAML V1.1 has made one backwards-incompatible change to SAML V1.0, described  
1407 in Section 5.4.7.

#### 1408 4.1.1 Schema Version

1409 As a non-normative documentation mechanism, any XML schema instances published as part of the  
1410 specification set will contain a schema "version" attribute in the form *Major.Minor*, reflecting the  
1411 specification set version in which it has been published. Validating implementations MAY use the attribute  
1412 as a means of distinguishing which version of a schema is being used to validate messages, or to support  
1413 a multiplicity of versions of the same logical schema.

#### 1414 4.1.2 SAML Assertion Version

1415 The SAML <Assertion> element contains attributes for expressing the major and minor version of the  
1416 assertion using a pair of integers. Each version of the SAML specification set will be construed so as to  
1417 document the syntax, semantics, and processing rules of the assertions of the same version. That is,  
1418 specification set version 1.0 describes assertion version 1.0, and so on.

1419 There is explicitly NO relationship between the assertion version and the SAML assertion XML  
1420 namespace that contains the schema definitions for that assertion version.

1421 The following processing rules apply:

- 1422 • A SAML authority MUST NOT issue any assertion with an assertion version number not supported by  
1423 the authority.
- 1424 • A SAML relying party MUST NOT process any assertion with a major assertion version number not  
1425 supported by the relying party.
- 1426 • A SAML relying party MAY process or MAY reject an assertion whose minor assertion version  
1427 number is higher than the minor assertion version number supported by the relying party. However,  
1428 all assertions that share a major assertion version number MUST share the same general processing

1429 rules and semantics, and MAY be treated in a uniform way by an implementation. That is, if a V1.1  
1430 assertion shares the syntax of a V1.0 assertion, an implementation MAY treat the assertion as a V1.0  
1431 assertion without ill effect.

### 1432 **4.1.3 SAML Protocol Version**

1433 The SAML protocol `<Request>` and `<Response>` elements contain attributes for expressing the major  
1434 and minor version of the request or response message using a pair of integers. Each version of the SAML  
1435 specification set will be construed so as to document the syntax, semantics, and processing rules of the  
1436 protocol messages of the same version. That is, specification set version 1.0 describes request and  
1437 response version V1.0, and so on.

1438 There is explicitly NO relationship between the protocol version and the SAML protocol XML namespace  
1439 that contains the schema definitions for protocol messages for that protocol version.

1440 The version numbers used in SAML protocol `<Request>` and `<Response>` elements will be the same  
1441 for any particular revision of the SAML specification set.

#### 1442 **4.1.3.1 Request Version**

1443 The following processing rules apply to requests:

- 1444 • A SAML requester SHOULD issue requests with the highest request version supported by both the  
1445 SAML requester and the SAML responder.
- 1446 • If the SAML requester does not know the capabilities of the SAML responder, then it should assume  
1447 that it supports requests with the highest request version supported by the requester.
- 1448 • A SAML requester MUST NOT issue a request message with a request version number matching a  
1449 response version number that the requester does not support.
- 1450 • A SAML responder MUST reject any request with a major request version number not supported by  
1451 the responder.
- 1452 • A SAML responder MAY process or MAY reject any request whose minor request version number is  
1453 higher than the highest supported request version that it supports. However, all requests that share a  
1454 major request version number MUST share the same general processing rules and semantics, and  
1455 MAY be treated in a uniform way by an implementation. That is, if a V1.1 request shares the syntax of  
1456 a V1.0 request, a responder MAY treat the request message as a V1.0 request without ill effect.

#### 1457 **4.1.4 Response Version**

1458 The following processing rules apply to responses:

- 1459 • A SAML responder MUST NOT issue a response message with a response version number higher  
1460 than the request version number of the corresponding request message.
- 1461 • A SAML responder MUST NOT issue a response message with a major response version number  
1462 lower than the major request version number of the corresponding request message except to report  
1463 the error `RequestVersionTooHigh`.

1464 An error response resulting from incompatible SAML protocol versions MUST result in reporting a top-  
1465 level `<StatusCode>` value of `VersionMismatch`, and MAY result in reporting one of the following  
1466 second-level values: `RequestVersionTooHigh`, `RequestVersionTooLow`, or  
1467 `RequestVersionDeprecated`.

#### 1468 **4.1.5 Permissible Version Combinations**

1469 In general, assertions of a particular major version may appear in response messages of the same major  
1470 version, as permitted by the importation of the SAML assertion namespace into the SAML protocol  
1471 schema. Future versions of this specification are expected to explicitly describe the permitted  
1472 combinations across major versions.

1473 Specifically, this permits a V1.1 assertion to appear in a V1.0 response message and a V1.0 assertion to  
1474 appear in a V1.1 response message.

## 1475 **4.2 SAML Namespace Version**

1476 XML schema instances and "qualified names" (QNames) published as part of the specification set contain  
1477 one or more target namespaces into which the type, element, and attribute definitions are placed. Each  
1478 namespace is distinct from the others, and represents, in shorthand, the structural and syntactical  
1479 definitions that make up that part of the specification.

1480 The namespace URIs defined by the specification set will generally contain version information of the  
1481 form *Major.Minor* somewhere in the URI. The major and minor version in the URI MUST correspond to  
1482 the major and minor version of the specification set in which the namespace is first introduced and  
1483 defined. This information is not typically consumed by an XML processor, which treats the namespace  
1484 opaquely, but is intended to communicate the relationship between the specification set and the  
1485 namespaces it defines.

1486 As a general rule, implementers can expect the namespaces (and the associated schema definitions)  
1487 defined by a major revision of the specification set to remain valid and stable across minor revisions of  
1488 the specification. New namespaces may be introduced, and when necessary, old namespaces replaced,  
1489 but this is expected to be rare. In such cases, the older namespaces and their associated definitions  
1490 should be expected to remain valid until a major specification set revision.

### 1491 **4.2.1 Schema Evolution**

1492 In general, maintaining namespace stability while adding or changing the content of a schema are  
1493 competing goals. While certain design strategies can facilitate such changes, it is complex to predict how  
1494 older implementations will react to any given change, making forward compatibility difficult to achieve.  
1495 Nevertheless, the right to make such changes in minor revisions is reserved, in the interest of namespace  
1496 stability. Except in special circumstances (for example to correct major deficiencies or fix errors),  
1497 implementations should expect forward compatible schema changes in minor revisions, allowing new  
1498 messages to validate against older schemas.

1499 Implementations SHOULD expect and be prepared to deal with new extensions and message types in  
1500 accordance with the processing rules laid out for those types. Minor revisions MAY introduce new types  
1501 that leverage the extension facilities described in Section 6. Older implementations SHOULD reject such  
1502 extensions gracefully when they are encountered in contexts that dictate mandatory semantics. Examples  
1503 include new query, statement, or condition types.

---

## 5 SAML and XML Signature Syntax and Processing

1504

1505 SAML assertions and SAML protocol request and response messages may be signed, with the following  
1506 benefits:

- 1507 • An assertion signed by the SAML authority supports:
  - 1508 – Assertion integrity.
  - 1509 – Authentication of the SAML authority to a SAML relying party.
  - 1510 – If the signature is based on the SAML authority's public-private key pair, then it also provides for  
1511 non-repudiation of origin.
- 1512 • A SAML protocol request or response message signed by the message originator supports:
  - 1513 – Message integrity.
  - 1514 – Authentication of message origin to a destination.
  - 1515 – If the signature is based on the originator's public-private key pair, then it also provides for non-  
1516 repudiation of origin.

1517 A digital signature is not always required in SAML. For example, it may not be required in the following  
1518 situations:

- 1519 • In some circumstances signatures may be "inherited," such as when an unsigned assertion gains  
1520 protection from a signature on the containing protocol response message. "Inherited" signatures  
1521 should be used with care when the contained object (such as the assertion) is intended to have a  
1522 non-transitory lifetime. The reason is that the entire context must be retained to allow validation,  
1523 exposing the XML content and adding potentially unnecessary overhead.
- 1524 • The SAML relying party or SAML requester may have obtained an assertion or protocol message  
1525 from the SAML authority or SAML responder directly (with no intermediaries) through a secure  
1526 channel, with the SAML authority or SAML responder having authenticated to the relying party or  
1527 SAML responder by some means other than a digital signature.

1528 Many different techniques are available for "direct" authentication and secure channel establishment  
1529 between two parties. The list includes TLS/SSL, HMAC, password-based mechanisms, etc. In addition,  
1530 the applicable security requirements depend on the communicating applications and the nature of the  
1531 assertion or message transported.

1532 It is recommended that, in all other contexts, digital signatures be used for assertions and request and  
1533 response messages. Specifically:

- 1534 • A SAML assertion obtained by a SAML relying party from an entity other than the SAML authority  
1535 SHOULD be signed by the SAML authority.
- 1536 • A SAML protocol message arriving at a destination from an entity other than the originating site  
1537 SHOULD be signed by the origin site.

1538 Profiles may specify alternative signature mechanisms such as S/MIME or signed Java objects that  
1539 contain SAML documents. Caveats about retaining context and interoperability apply. XML Signatures  
1540 are intended to be the primary SAML signature mechanism, but the specification attempts to ensure  
1541 compatibility with profiles that may require other mechanisms.

1542 Unless a profile specifies an alternative signature mechanism, enveloped XML Digital Signatures MUST  
1543 be used if signing.

### 5.1 Signing Assertions

1544

1545 All SAML assertions MAY be signed using the XML Signature. This is reflected in the assertion schema  
1546 as described in Section 2.3.

## 1547 5.2 Request/Response Signing

1548 All SAML protocol request and response messages MAY be signed using the XML Signature. This is  
1549 reflected in the schema as described in Sections 3.2 and 3.4.

## 1550 5.3 Signature Inheritance

1551 A SAML assertion may be embedded within another SAML element, such as an enclosing `<Assertion>`  
1552 or a `<Request>` or `<Response>`, which may be signed. When a SAML assertion does not contain a  
1553 `<ds:Signature>` element, but is contained in an enclosing SAML element that contains a  
1554 `<ds:Signature>` element, and the signature applies to the `<Assertion>` element and all its children,  
1555 then the assertion can be considered to inherit the signature from the enclosing element. The resulting  
1556 interpretation should be equivalent to the case where the assertion itself was signed with the same key  
1557 and signature options.

1558 Many SAML use cases involve SAML XML data enclosed within other protected data structures such as  
1559 signed SOAP messages, S/MIME packages, and authenticated SSL connections. SAML profiles may  
1560 define additional rules for interpreting SAML elements as inheriting signatures or other authentication  
1561 information from the surrounding context, but no such inheritance should be inferred unless specifically  
1562 identified by the profile.

## 1563 5.4 XML Signature Profile

1564 The XML Signature specification [XMLSig] calls out a general XML syntax for signing data with flexibility  
1565 and many choices. This section details the constraints on these facilities so that SAML processors do not  
1566 have to deal with the full generality of XML Signature processing. This usage makes specific use of the  
1567 `xsd:ID`-typed attributes optionally present on the root elements to which signatures can apply: the  
1568 `AssertionID` attribute on `<Assertion>`, the `RequestID` attribute on `<Request>`, and the  
1569 `ResponseID` attribute on `<Response>`. These three attributes are collectively referred to in this section  
1570 as the identifier attributes.

### 1571 5.4.1 Signing Formats and Algorithms

1572 XML Signature has three ways of relating a signature to a document: enveloping, enveloped, and  
1573 detached.

1574 SAML assertions and protocols MUST use enveloped signatures when signing assertions and protocol  
1575 messages. SAML processors SHOULD support the use of RSA signing and verification for public key  
1576 operations in accordance with the algorithm identified by <http://www.w3.org/2000/09/xmldsig#rsa-sha1>.

### 1577 5.4.2 References

1578 Signed SAML assertions and protocol messages MUST supply a value for the identifier attribute on the  
1579 root element (`<Assertion>`, `<Request>`, or `<Response>`). The assertion's or message's root element  
1580 may or may not be the root element of the actual XML document containing the signed assertion or  
1581 message.

1582 Signatures MUST contain a single `<ds:Reference>` containing a URI reference to the identifier attribute  
1583 value of the root element of the message being signed. For example, if the attribute value is "foo", then  
1584 the `URI` attribute in the `<ds:Reference>` element MUST be "#foo".

### 1585 5.4.3 Canonicalization Method

1586 SAML implementations SHOULD use Exclusive Canonicalization [Excl-C14N], with or without comments,  
1587 both in the `<ds:CanonicalizationMethod>` element of `<ds:SignedInfo>`, and as a  
1588 `<ds:Transform>` algorithm. Use of Exclusive Canonicalization ensures that signatures created over  
1589 SAML messages embedded in an XML context can be verified independent of that context.

#### 1590 5.4.4 Transforms

1591 Signatures in SAML messages SHOULD NOT contain transforms other than the enveloped signature  
1592 transform (with the identifier <http://www.w3.org/2000/09/xmldsig#enveloped-signature>) or the exclusive  
1593 canonicalization transforms (with the identifier <http://www.w3.org/2001/10/xml-exc-c14n#> or  
1594 <http://www.w3.org/2001/10/xml-exc-c14n#WithComments>).

1595 Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid. If they do  
1596 not, verifiers MUST ensure that no content of the SAML message is excluded from the signature. This  
1597 can be accomplished by establishing out-of-band agreement as to what transforms are acceptable, or by  
1598 applying the transforms manually to the content and reverifying the result as consisting of the same  
1599 SAML message.

#### 1600 5.4.5 KeyInfo

1601 XML Signature [XMLSig] defines usage of the `<ds:KeyInfo>` element. SAML does not require the  
1602 use of `<ds:KeyInfo>` nor does it impose any restrictions on its use. Therefore, `<ds:KeyInfo>` MAY  
1603 be absent.

#### 1604 5.4.6 Binding Between Statements in a Multi-Statement Assertion

1605 Use of signing does not affect semantics of statements within assertions in any way, as stated in Section  
1606 2.

#### 1607 5.4.7 Interoperability with SAML V1.0

1608 The use of XML Signature [XMLSig] described above is incompatible with the usage described in the  
1609 SAML V1.0 specification [SAMLCore1.0]. The original profile was underspecified and was insufficient to  
1610 ensure interoperability. It was constrained by the inability to use URI references to identify the SAML  
1611 content to be signed. With this limitation removed by the addition of SAML identifier attributes, a decision  
1612 has been made to forgo backwards compatibility with the older specification in this respect.

#### 1613 5.4.8 Example

1614 Following is an example of a signed response containing a signed assertion. Line breaks have been  
1615 added for readability; the signatures are not valid and cannot be successfully verified.

```
1616 <Response  
1617   IssueInstant="2003-04-17T00:46:02Z"  
1618   MajorVersion="1"  
1619   MinorVersion="1"  
1620   Recipient="www.opensaml.org"  
1621   ResponseID="_c7055387-af61-4fce-8b98-e2927324b306"  
1622   xmlns="urn:oasis:names:tc:SAML:1.0:protocol"  
1623   xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"  
1624   xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
1625   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">  
1626   <ds:Signature  
1627     xmlns:ds="http://www.w3.org/2000/09/xmldsig#">  
1628     <ds:SignedInfo>  
1629     <ds:CanonicalizationMethod  
1630       Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />  
1631     <ds:SignatureMethod  
1632       Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />  
1633     <ds:Reference  
1634       URI="#_c7055387-af61-4fce-8b98-e2927324b306">  
1635     <ds:Transforms>  
1636     <ds:Transform  
1637       Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />  
1638     <ds:Transform
```

```

1639     Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
1640 <InclusiveNamespaces
1641   PrefixList="#default saml samlp ds xsd xsi"
1642   xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />
1643 </ds:Transform>
1644 </ds:Transforms>
1645 <ds:DigestMethod
1646   Algorithm="http://www.w3.org/2000/09/xmlsig#sha1" />
1647 <ds:DigestValue>TCDVSuG6grhyHbzhQFWFzGrxIPE=</ds:DigestValue>
1648 </ds:Reference>
1649 </ds:SignedInfo>
1650 <ds:SignatureValue>
1651 x/GyPbzmfEe85pGD3c1aXG4Vspb9V9jGCjwcRCKrtwPS6vdVNCcY5rHaFPYWkf+5
1652 EIYcPzx+pX1h43SmwviCqXRjRtMANWbHLhWAptaK1ywS7gFgsD01qjyen3CP+m3D
1653 w6vKhaqledl0BYyrIzb4KkH04ahNyBVXbJwqv5pUaE4=</ds:SignatureValue>
1654 <ds:KeyInfo>
1655 <ds:X509Data>
1656 <ds:X509Certificate>
1657 MIIcYjCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgaxCzAJBgNVBAYTA1VT
1658 MRIwEAYDVQQIEwIeLXAuXNjb25zaW4xZDA0BGNVBAcTB01hZG1zb24xIDAeBgNVBAoT
1659 FlVuaXZlcnNpdHkgb2YgV2l2Y29uc2luMSswKQYDVQLEyJEaXZpc2lvbiBvZiBJ
1660 bmZvcmlhdG1vbiBUZWNobm9sb2d5MSUwIiwYDVQDExxIRVBLSSBTZXXJ2ZXIqQ0Eg
1661 LS0gMjAwMjA3MDFBMB4XDTAyMjY1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1
1662 MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1
1663 MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1
1664 MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1
1665 MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1
1666 MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1
1667 MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1
1668 MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1
1669 MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1
1670 MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1
1671 MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1
1672 MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1
1673 MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1
1674 MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1
1675 MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1MjM1
1676 <Status><StatusCode Value="samlp:Success" /></Status>
1677 <Assertion
1678   AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
1679   IssueInstant="2003-04-17T00:46:02Z"
1680   Issuer="www.opensaml.org"
1681   MajorVersion="1"
1682   MinorVersion="1"
1683   xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
1684   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1685   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
1686 <Conditions
1687   NotBefore="2003-04-17T00:46:02Z"
1688   NotOnOrAfter="2003-04-17T00:51:02Z">
1689 <AudienceRestrictionCondition><Audience>http://www.opensaml.org</Audience>
1690 </AudienceRestrictionCondition></Conditions>
1691 <AuthenticationStatement
1692   AuthenticationInstant="2003-04-17T00:46:00Z"
1693   AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
1694 <Subject>
1695 <NameIdentifier
1696   Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
1697 scott@example.org</NameIdentifier>
1698 <SubjectConfirmation>
1699 <ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:bearer</ConfirmationMethod>
1700 </SubjectConfirmation></Subject>
1701 <SubjectLocality

```

```

1702     IPAddress="127.0.0.1"/>
1703 </AuthenticationStatement>
1704 <ds:Signature
1705     xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
1706 <ds:SignedInfo>
1707 <ds:CanonicalizationMethod
1708     Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1709 <ds:SignatureMethod
1710     Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
1711 <ds:Reference
1712     URI="#_a75adf55-01d7-40cc-929f-dbd8372ebdfc">
1713 <ds:Transforms>
1714 <ds:Transform
1715     Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
1716 <ds:Transform
1717     Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1718 <InclusiveNamespaces
1719     PrefixList="#default saml samlp ds xsd xsi"
1720     xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />
1721 </ds:Transform>
1722 </ds:Transforms>
1723 <ds:DigestMethod
1724     Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
1725 <ds:DigestValue>Kclet6XcaOgOWXM4gty6/UNdviI=</ds:DigestValue>
1726 </ds:Reference>
1727 </ds:SignedInfo>
1728 <ds:SignatureValue>
1729 hq4zk+ZknjggCQgZm7ea8fI79gJEsRy3E8LHDpYXWQIgZpkJN9CMLG8ENR4Nrw+n
1730 7iyzixBvKXX8P53BTCT4VghPBWhFYSt9tHWu/AtJfOTh6qaAsNdeCyG86jmtP3TD
1731 MWuL/cBUj2OtBZOQMFN7jQ9YB7k1Iz3RqVL+wNmeWI4=</ds:SignatureValue>
1732 <ds:KeyInfo>
1733 <ds:X509Data>
1734 <ds:X509Certificate>
1735 MIIcYjCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgaxCzAJBgNVBAYTA1VT
1736 MRIwEAYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24xIDAeBgNVBAoT
1737 F1VuaXZlcnNpdHkkgb2YgV2l2Y29uc2luMSswKQYDVQQLEyJEaXZpc2lvbiBvZiBJ
1738 bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
1739 Ls0gMjAwMjA3MDFBMB4XDTEyMDcyNjA3Mjc1MVoXDTEyMDkwNDA3Mjc1MVoYYSx
1740 CzAJBgNVBAYTA1VTMREwDwYDVQQQIEwhNaWNoaWdhbWJESMBAGAlUEBxMjQW5uIEFy
1741 Ym9yMQ4wDAYDVQQKEwVvV2l2Y29uc2luMSswKQYDVQIExMjAwMjA3MDFBMB4X
1742 dTEyMDcyNjA3Mjc1MVoYDzEwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAw
1743 CSqGSIB3DQEBAQUAA4GNADCBiQKBgQDZSAb2sxvhAXnXVIVTxs8vuRay+x50z7GJj
1744 IHRYQgIv6IqaGG04eTcyVMhoeKE0b45QgvBIAoAPSZB113R6+KYiE7x4XAWIrcP+
1745 c2MZVeXeTgV3Yz+USLg2Ylon+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W7O27rhRjE
1746 pmqOIFGTWQIDAQABox0wGzAMBgNVHRMBAf8EAjAAMAsGAlUdDwQEAwIFoDANBgkq
1747 hkiG9w0BAQQFAA0BgQBfDqEW+OI3jqBQHIBzhuJN/PizdN7s/z4D5d3pptWDJf2n
1748 qgi7lFV6MDkHmTvTqBtjmNk3No7v/dnP6Hr7wHxvCCRwubnmIfz6QZAv2FU78pLX
1749 8I3bsbmRAUg4UP9hH6ABVq4KQKMknxulxQxLhpRlyLGPdiowMNTreEG8cCx3w/w==
1750 </ds:X509Certificate>
1751 </ds:X509Data>
1752 </ds:KeyInfo>
1753 </ds:Signature></Assertion></Response>

```

1754

---

## 6 SAML Extensions

1755 The SAML schemas support extensibility. An example of an application that extends SAML assertions is  
1756 the Liberty Protocols and Schema Specification [**LibertyProt**]. The following sections explain how to use  
1757 the extensibility features in SAML to create extension schemas.

1758 Note that elements in the SAML schemas are not blocked from substitution, so that all SAML elements  
1759 MAY serve as the head element of a substitution group. Also, types are not defined as *final*, so that all  
1760 SAML types MAY be extended and restricted. The following sections discuss only elements that have  
1761 been specifically designed to support extensibility.

### 6.1 Assertion Schema Extension

1763 The SAML assertion schema is designed to permit separate processing of the assertion package and the  
1764 statements it contains, if the extension mechanism is used for either part.

1765 The following elements are intended specifically for use as extension points in an extension schema; their  
1766 types are set to *abstract*, and are thus usable only as the base of a derived type:

- 1767 • <Condition>
- 1768 • <Statement>
- 1769 • <SubjectStatement>

1770 The following elements that are directly usable as part of SAML MAY be extended:

- 1771 • <AuthenticationStatement>
- 1772 • <AuthorizationDecisionStatement>
- 1773 • <AttributeStatement>
- 1774 • <AudienceRestrictionCondition>

1775 The following elements are defined to allow elements from arbitrary namespaces within them, which  
1776 serves as a built-in extension point without requiring an extension schema:

- 1777 • <AttributeValue>
- 1778 • <Advice>

### 6.2 Protocol Schema Extension

1780 The following SAML protocol elements are intended specifically for use as extension points in an  
1781 extension schema; their types are set to *abstract*, and are thus usable only as the base of a derived  
1782 type:

- 1783 • <Query>
- 1784 • <SubjectQuery>

1785 The following elements that are directly usable as part of SAML MAY be extended:

- 1786 • <Request>
- 1787 • <AuthenticationQuery>
- 1788 • <AuthorizationDecisionQuery>
- 1789 • <AttributeQuery>
- 1790 • <Response>

## 1791 6.3 Use of Type Derivation and Substitution Groups

1792 W3C XML Schema [Schema1] provides two principal mechanisms for specifying an element of an  
1793 extended type: type derivation and substitution groups.

1794 For example, a `<Statement>` element can be assigned the type **NewStatementType** by means of the  
1795 `xsi:type` attribute. For such an element to be schema-valid, **NewStatementType** needs to be derived  
1796 from **StatementType**. The following example of a SAML assertion assumes that the extension schema  
1797 (represented by the `new:` prefix) has defined this new type:

```
1798 <saml:Assertion ...>  
1799   <saml:Statement xsi:type="new:NewStatementType">  
1800     ...  
1801   </saml:Statement>  
1802 </saml:Assertion>
```

1803 Alternatively, the extension schema can define a `<NewStatement>` element that is a member of a  
1804 substitution group that has `<Statement>` as a head element. For the substituted element to be schema-  
1805 valid, it needs to have a type that matches or is derived from the head element's type. The following is an  
1806 example of an extension schema fragment that defines this new element:

```
1807 <xsd:element "NewStatement" type="new:NewStatementType"  
1808   substitutionGroup="saml:Statement"/>
```

1809 The substitution group declaration allows the `<NewStatement>` element to be used anywhere the SAML  
1810 `<Statement>` element can be used. The following is an example of a SAML assertion that uses the  
1811 extension element:

```
1812 <saml:Assertion ...>  
1813   <new:NewStatement>  
1814     ...  
1815   </new:NewStatement>  
1816 </saml:Assertion>
```

1817 The choice of extension method has no effect on the semantics of the XML document but does have  
1818 implications for interoperability.

1819 The advantages of type derivation are as follows:

- 1820 • A document can be more fully interpreted by a parser that does not have access to the extension  
1821 schema because a “native” SAML element is available.
- 1822 • At the time of this writing, some W3C XML Schema validators do not support substitution groups,  
1823 whereas the `xsi:type` attribute is widely supported.

1824 The advantage of substitution groups is that a document can be explained without the need to explain the  
1825 functioning of the `xsi:type` attribute.

---

## 1826 7 SAML-Defined Identifiers

1827 The following sections define URI-based identifiers for common authentication methods, resource access  
1828 actions, and subject name identifier formats.

1829 Where possible an existing URN is used to specify a protocol. In the case of IETF protocols the URN of  
1830 the most current RFC that specifies the protocol is used. URI references created specifically for SAML  
1831 have one of the following stems:

```
1832 urn:oasis:names:tc:SAML:1.0:  
1833 urn:oasis:names:tc:SAML:1.1:
```

### 1834 7.1 Authentication Method Identifiers

1835 The `AuthenticationMethod` attribute of an `<AuthenticationStatement>` and the  
1836 `<SubjectConfirmationMethod>` element of a SAML subject perform different functions, although  
1837 both can refer to the same underlying mechanisms. An authentication statement with an  
1838 `AuthenticationMethod` attribute describes an authentication act that occurred in the past. The  
1839 `AuthenticationMethod` attribute indicates how that authentication was done. Note that the  
1840 authentication statement does not provide the means to perform that authentication, such as a password,  
1841 key, or certificate.

1842 In contrast, `<SubjectConfirmationMethod>` is a part of the `<SubjectConfirmation>` element,  
1843 which is an optional part of a SAML subject. `<SubjectConfirmation>` is used to allow the SAML  
1844 relying party to confirm that the request or message came from a system entity that corresponds to the  
1845 subject in the statement or query. The `<SubjectConfirmationMethod>` element indicates the method  
1846 that the relying party can use to do this in the future. This may or may not have any relationship to an  
1847 authentication that was performed previously. Unlike the authentication method, the subject confirmation  
1848 method may be accompanied by some piece of information, such as a certificate or key, that will allow the  
1849 relying party to perform the necessary check.

1850 Subject confirmation methods are defined in the SAML profiles in which they are used; see the SAML  
1851 bindings and profiles specification [**SAMLBind**] for more information. Additional methods may be added  
1852 by defining new profiles or by private agreement.

1853 The following identifiers refer to SAML-specified authentication methods.

#### 1854 7.1.1 Password

1855 **URI:** urn:oasis:names:tc:SAML:1.0:am:password

1856 The authentication was performed by means of a password.

#### 1857 7.1.2 Kerberos

1858 **URI:** urn:ietf:rfc:1510

1859 The authentication was performed by means of the Kerberos protocol [**RFC 1510**], an instantiation of the  
1860 Needham-Schroeder symmetric key authentication mechanism [**Needham78**].

#### 1861 7.1.3 Secure Remote Password (SRP)

1862 **URI:** urn:ietf:rfc:2945

1863 The authentication was performed by means of Secure Remote Password protocol as specified in [**RFC**  
1864 **2945**].

1865 **7.1.4 Hardware Token**

1866 **URI:** urn:oasis:names:tc:SAML:1.0:am:HardwareToken

1867 The authentication was performed using some (unspecified) hardware token.

1868 **7.1.5 SSL/TLS Certificate Based Client Authentication:**

1869 **URI:** urn:ietf:rfc:2246

1870 The authentication was performed using either the SSL or TLS protocol with certificate-based client  
1871 authentication. TLS is described in **[RFC 2246]**.

1872 **7.1.6 X.509 Public Key**

1873 **URI:** urn:oasis:names:tc:SAML:1.0:am:X509-PKI

1874 The authentication was performed by some (unspecified) mechanism on a key authenticated by means of  
1875 an X.509 PKI **[X.500][PKIX]**. It may have been one of the mechanisms for which a more specific identifier  
1876 has been defined below.

1877 **7.1.7 PGP Public Key**

1878 **URI:** urn:oasis:names:tc:SAML:1.0:am:PGP

1879 The authentication was performed by some (unspecified) mechanism on a key authenticated by means of  
1880 a PGP web of trust **[PGP]**. It may have been one of the mechanisms for which a more specific identifier  
1881 has been defined below.

1882 **7.1.8 SPKI Public Key**

1883 **URI:** urn:oasis:names:tc:SAML:1.0:am:SPKI

1884 The authentication was performed by some (unspecified) mechanism on a key authenticated by means of  
1885 a SPKI PKI **[SPKI]**. It may have been one of the mechanisms for which a more specific identifier has  
1886 been defined below.

1887 **7.1.9 XKMS Public Key**

1888 **URI:** urn:oasis:names:tc:SAML:1.0:am:XKMS

1889 The authentication was performed by some (unspecified) mechanism on a key authenticated by means of  
1890 a XKMS trust service **[XKMS]**. It may have been one of the mechanisms for which a more specific  
1891 identifier has been defined below.

1892 **7.1.10 XML Digital Signature**

1893 **URI:** urn:ietf:rfc:3075

1894 The authentication was performed by means of an XML digital signature **[RFC 3075]**.

1895 **7.1.11 Unspecified**

1896 **URI:** urn:oasis:names:tc:SAML:1.0:am:unspecified

1897 The authentication was performed by an unspecified means.

1898 **7.2 Action Namespace Identifiers**

1899 The following identifiers MAY be used in the `Namespace` attribute of the `<Action>` element (see Section  
1900 2.4.5.1) to refer to common sets of actions to perform on resources.

## 1901 **7.2.1 Read/Write/Execute/Delete/Control**

1902 **URI:** urn:oasis:names:tc:SAML:1.0:action:rwedc

1903 Defined actions:

1904     Read Write Execute Delete Control

1905 These actions are interpreted as follows:

1906 Read

1907     The subject may read the resource.

1908 Write

1909     The subject may modify the resource.

1910 Execute

1911     The subject may execute the resource.

1912 Delete

1913     The subject may delete the resource.

1914 Control

1915     The subject may specify the access control policy for the resource.

## 1916 **7.2.2 Read/Write/Execute/Delete/Control with Negation**

1917 **URI:** urn:oasis:names:tc:SAML:1.0:action:rwedc-negation

1918 Defined actions:

1919     Read Write Execute Delete Control ~Read ~Write ~Execute ~Delete ~Control

1920 The actions specified in Section 7.2.1 are interpreted in the same manner described there. Actions  
1921 prefixed with a tilde (~) are negated permissions and are used to affirmatively specify that the stated  
1922 permission is denied. Thus a subject described as being authorized to perform the action ~Read is  
1923 affirmatively denied read permission.

1924 A SAML authority MUST NOT authorize both an action and its negated form.

## 1925 **7.2.3 Get/Head/Put/Post**

1926 **URI:** urn:oasis:names:tc:SAML:1.0:action:ghpp

1927 Defined actions:

1928     GET HEAD PUT POST

1929 These actions bind to the corresponding HTTP operations. For example a subject authorized to perform  
1930 the GET action on a resource is authorized to retrieve it.

1931 The GET and HEAD actions loosely correspond to the conventional read permission and the PUT and  
1932 POST actions to the write permission. The correspondence is not exact however since an HTTP GET  
1933 operation may cause data to be modified and a POST operation may cause modification to a resource  
1934 other than the one specified in the request. For this reason a separate Action URI reference specifier is  
1935 provided.

## 1936 **7.2.4 UNIX File Permissions**

1937 **URI:** urn:oasis:names:tc:SAML:1.0:action:unix

1938 The defined actions are the set of UNIX file access permissions expressed in the numeric (octal) notation.

1939 The action string is a four-digit numeric code:

1940     *extended user group world*

1941 Where the *extended* access permission has the value

- 1942 +2 if sgid is set  
1943 +4 if suid is set  
1944 The *user group* and *world* access permissions have the value  
1945 +1 if execute permission is granted  
1946 +2 if write permission is granted  
1947 +4 if read permission is granted  
1948 For example, 0754 denotes the UNIX file access permission: user read, write and execute; group read  
1949 and execute; and world read.

## 1950 7.3 NameIdentifier Format Identifiers

- 1951 The following identifiers MAY be used in the Format attribute of the <NameIdentifier> element (see  
1952 Section 2.4.2.2) to refer to common formats for the content of the <NameIdentifier> element. The  
1953 recommended identifiers shown below SHOULD be used in preference to the deprecated identifiers,  
1954 which are planned to be removed in the next major version of the SAML assertion specification.

### 1955 7.3.1 Unspecified

- 1956 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified  
1957 The interpretation of the content of the <NameQualifier> element is left to individual implementations.

### 1958 7.3.2 Email Address

- 1959 **Recommended URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress  
1960 **Deprecated URI:** urn:oasis:names:tc:SAML:1.0:assertion#emailAddress  
1961 Indicates that the content of the <NameIdentifier> element is in the form of an email address,  
1962 specifically "addr-spec" as defined in IETF RFC 2822 [RFC 2822] §3.4.1. An addr-spec has the form  
1963 local-part@domain. Note that an addr-spec has no phrase (such as a common name) before it, has no  
1964 comment (text surrounded in parentheses) after it, and is not surrounded by "<" and ">".

### 1965 7.3.3 X.509 Subject Name

- 1966 **Recommended URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName  
1967 **Deprecated URI:** urn:oasis:names:tc:SAML:1.0:assertion#X509SubjectName  
1968 Indicates that the content of the <NameIdentifier> element is in the form specified for the contents of  
1969 the <ds:X509SubjectName> element in the XML Signature Recommendation [XMLSig]. Implementors  
1970 should note that the XML Signature specification specifies encoding rules for X.509 subject names that  
1971 differ from the rules given in IETF RFC 2253 [RFC 2253].

### 1972 7.3.4 Windows Domain Qualified Name

- 1973 **Recommended URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName  
1974 **Deprecated URI:** urn:oasis:names:tc:SAML:1.0:assertion#WindowsDomainQualifiedName  
1975 Indicates that the content of the <NameIdentifier> element is a Windows domain qualified name. A  
1976 Windows domain qualified user name is a string of the form "DomainName\UserName". The domain  
1977 name and "\" separator MAY be omitted.

---

## 8 References

- 1978
- 1979 [Excl-C14N] J. Boyer et al. Exclusive XML Canonicalization Version 1.0. World Wide Web Consortium, July 2002. <http://www.w3.org/TR/xml-exc-c14n/>.
- 1980
- 1981 [LibertyProt] J. Beatty et al., *Liberty Protocols and Schema Specification* Version 1.1, Liberty Alliance Project, January 2003, [http://www.projectliberty.org/specs/archive/v1\\_1/liberty-architecture-protocols-schema-v1.1.pdf](http://www.projectliberty.org/specs/archive/v1_1/liberty-architecture-protocols-schema-v1.1.pdf).
- 1982
- 1983
- 1984
- 1985 [Needham78] R. Needham et al. *Using Encryption for Authentication in Large Networks of Computers*. Communications of the ACM, Vol. 21 (12), pp. 993-999. December 1978.
- 1986
- 1987
- 1988 [PGP] Atkins, D., Stallings, W. and P. Zimmermann..*PGP Message Exchange Formats*. IETF RFC 1991, August 1996. <http://www.ietf.org/rfc/rfc1991.txt>.
- 1989
- 1990 [PKIX] R. Housley, W. Ford, W. Polk, D. Solo. *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*. IETF RFC 2459, January 1999. <http://www.ietf.org/rfc/rfc2459.txt>.
- 1991
- 1992
- 1993 [RFC 1510] J. Kohl, C. Neuman. *The Kerberos Network Authentication Requestor (V5)*. IETF RFC 1510, September 1993. <http://www.ietf.org/rfc/rfc1510.txt>.
- 1994
- 1995 [RFC 2119] S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF RFC 2119, March 1997. <http://www.ietf.org/rfc/rfc2119.txt>.
- 1996
- 1997 [RFC 2246] T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. IETF RFC 2246, January 1999. <http://www.ietf.org/rfc/rfc2246.txt>.
- 1998
- 1999 [RFC 2253] M. Wahl et al. *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names*. IETF RFC 2253, December 1997. <http://www.ietf.org/rfc/rfc2253.txt>.
- 2000
- 2001
- 2002 [RFC 2396] T. Berners-Lee et al. *Uniform Resource Identifiers (URI): Generic Syntax*. IETF RFC 2396, August, 1998. <http://www.ietf.org/rfc/rfc2396.txt>.
- 2003
- 2004 [RFC 2630] R. Housley. *Cryptographic Message Syntax*. IETF RFC 2630, June 1999. <http://www.ietf.org/rfc/rfc2630.txt>.
- 2005
- 2006 [RFC 2822] P. Resnick. *Internet Message Format*. IETF RFC 2822, April 2001. <http://www.ietf.org/rfc/rfc2822.txt>.
- 2007
- 2008 [RFC 2945] T. Wu. *The SRP Authentication and Key Exchange System*. IETF RFC 2945, September 2000. <http://www.ietf.org/rfc/rfc2945.txt>.
- 2009
- 2010 [RFC 3075] D. Eastlake, J. Reagle, D. Solo. *XML-Signature Syntax and Processing*. IETF RFC 3075, March 2001. <http://www.ietf.org/rfc/rfc3075.txt>.
- 2011
- 2012 [SAMLBind] E. Maler et al. *Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML)*. OASIS, May 2003. <http://www.oasis-open.org/committees/security/>.
- 2013
- 2014
- 2015 [SAMLConform] E. Maler et al. *Conformance Program Specification for the OASIS Security Assertion Markup Language (SAML)*. OASIS, May 2003. <http://www.oasis-open.org/committees/security/>.
- 2016
- 2017
- 2018 [SAMLCore1.0] E. Maler et al. *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)*. OASIS, November 2002. <http://www.oasis-open.org/committees/download.php/1371/oasis-sstc-saml-core-1.0.pdf>.
- 2019
- 2020
- 2021 [SAMLGloss] E. Maler et al. *Glossary for the OASIS Security Assertion Markup Language (SAML)*. OASIS, May 2003. <http://www.oasis-open.org/committees/security/>.
- 2022
- 2023 [SAMLPSchema] E. Maler et al. *SAML protocol schema*. OASIS, May 2003. <http://www.oasis-open.org/committees/security/>.
- 2024

2025	<b>[SAMLSecure]</b>	E. Maler et al. <i>Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML)</i> . OASIS, May 2003. <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
2026		
2027		
2028	<b>[SAML-XSD]</b>	E. Maler et al. <i>SAML assertion schema</i> . OASIS, May 2003. <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
2029		
2030	<b>[Schema1]</b>	H. S. Thompson et al. <i>XML Schema Part 1: Structures</i> . World Wide Web Consortium Recommendation, May 2001. <a href="http://www.w3.org/TR/xmlschema-1/">http://www.w3.org/TR/xmlschema-1/</a> .
2031		
2032	<b>[Schema2]</b>	P. V. Biron et al. <i>XML Schema Part 2: Datatypes</i> . World Wide Web Consortium Recommendation, May 2001. <a href="http://www.w3.org/TR/xmlschema-2/">http://www.w3.org/TR/xmlschema-2/</a> .
2033		
2034	<b>[SPKI]</b>	C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, T. Ylonen. <i>SPKI Certificate Theory</i> . IETF RFC 2693, September 1999.
2035		
2036		<a href="http://www.ietf.org/rfc/rfc2693.txt">http://www.ietf.org/rfc/rfc2693.txt</a> .
2037	<b>[UNICODE-C]</b>	M. Davis, M. J. Dürst. <i>Unicode Normalization Forms</i> . UNICODE Consortium, March 2001. <a href="http://www.unicode.org/unicode/reports/tr15/tr15-21.html">http://www.unicode.org/unicode/reports/tr15/tr15-21.html</a> .
2038		
2039	<b>[W3C-CHAR]</b>	M. J. Dürst. <i>Requirements for String Identity Matching and String Indexing</i> . World Wide Web Consortium, July 1998. <a href="http://www.w3.org/TR/WD-charreq">http://www.w3.org/TR/WD-charreq</a> .
2040		
2041	<b>[W3C-CharMod]</b>	M. J. Dürst. <i>Character Model for the World Wide Web 1.0</i> . World Wide Web Consortium, April, 2002. <a href="http://www.w3.org/TR/charmod/">http://www.w3.org/TR/charmod/</a> .
2042		
2043	<b>[X.500]</b>	ITU-T Recommendation X.501: Information Technology - Open Systems Interconnection - The Directory: Models. 1993.
2044		
2045	<b>[XKMS]</b>	W. Ford, P. Hallam-Baker, B. Fox, B. Dillaway, B. LaMacchia, J. Epstein, J. Lapp. XML Key Management Specification (XKMS). W3C Note 30 March 2001. <a href="http://www.w3.org/TR/xkms/">http://www.w3.org/TR/xkms/</a> .
2046		
2047		
2048	<b>[XML]</b>	T. Bray, et al. <i>Extensible Markup Language (XML) 1.0 (Second Edition)</i> . World Wide Web Consortium, October 2000. <a href="http://www.w3.org/TR/REC-xml">http://www.w3.org/TR/REC-xml</a> .
2049		
2050	<b>[XMLSig]</b>	D. Eastlake et al., <i>XML-Signature Syntax and Processing</i> , World Wide Web Consortium, February 2002. <a href="http://www.w3.org/TR/xmlsig-core/">http://www.w3.org/TR/xmlsig-core/</a> .
2051		
2052	<b>[XMLSig-XSD]</b>	XML Signature Schema. World Wide Web Consortium.
2053		<a href="http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd">http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd</a> .
2054		

---

## 2055 **Appendix A. Acknowledgments**

2056 The editors would like to acknowledge the contributions of the OASIS Security Services Technical  
2057 Committee, whose voting members at the time of publication were:

- 2058 • Frank Siebenlist, Argonne National Laboratory
- 2059 • Irving Reid, Baltimore Technologies
- 2060 • Hal Lockhart, BEA Systems
- 2061 • Steven Lewis, Booz Allen Hamilton
- 2062 • John Hughes, Entegriy Solutions
- 2063 • Carlisle Adams, Entrust
- 2064 • Jason Rouault, Hewlett-Packard
- 2065 • Maryann Hondo, IBM
- 2066 • Anthony Nadalin, IBM
- 2067 • Scott Cantor, individual
- 2068 • RL "Bob" Morgan, individual
- 2069 • Trevor Perrin, individual
- 2070 • Pdraig Moloney, NASA
- 2071 • Prateek Mishra, Netegrity (co-chair)
- 2072 • Frederick Hirsch, Nokia
- 2073 • Senthil Sengodan, Nokia
- 2074 • Timo Skytta, Nokia
- 2075 • Charles Knouse, Oblix
- 2076 • Steve Anderson, OpenNetwork
- 2077 • Simon Godik, Overxeer
- 2078 • Rob Philpott, RSA Security (co-chair)
- 2079 • Dipak Chopra, SAP
- 2080 • Jahan Moreh, Sigaba
- 2081 • Bhavna Bhatnagar, Sun Microsystems
- 2082 • Jeff Hodges, Sun Microsystems
- 2083 • Eve Maler, Sun Microsystems (coordinating editor)
- 2084 • Emily Xu, Sun Microsystems
- 2085 • Phillip Hallam-Baker, VeriSign

2086

---

## Appendix B. Notices

2087 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that  
2088 might be claimed to pertain to the implementation or use of the technology described in this document or  
2089 the extent to which any license under such rights might or might not be available; neither does it  
2090 represent that it has made any effort to identify any such rights. Information on OASIS's procedures with  
2091 respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights  
2092 made available for publication and any assurances of licenses to be made available, or the result of an  
2093 attempt made to obtain a general license or permission for the use of such proprietary rights by  
2094 implementors or users of this specification, can be obtained from the OASIS Executive Director.

2095 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications,  
2096 or other proprietary rights which may cover technology that may be required to implement this  
2097 specification. Please address the information to the OASIS Executive Director.

2098 Copyright © OASIS Open 2003. *All Rights Reserved.*

2099 This document and translations of it may be copied and furnished to others, and derivative works that  
2100 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published  
2101 and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice  
2102 and this paragraph are included on all such copies and derivative works. However, this document itself  
2103 may not be modified in any way, such as by removing the copyright notice or references to OASIS,  
2104 except as needed for the purpose of developing OASIS specifications, in which case the procedures for  
2105 copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required to  
2106 translate it into languages other than English.

2107 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors  
2108 or assigns.

2109 This document and the information contained herein is provided on an "AS IS" basis and OASIS  
2110 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY  
2111 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR  
2112 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.