



# Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0

Working Draft 110, ~~12 April~~ 11 May 2004

**Document identifier:**

sstc-saml-core-2.0-draft-110

**Location:**

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

**Editors:**

Scott Cantor, individual ([cantor.2@osu.edu](mailto:cantor.2@osu.edu))  
John Kemp, Nokia ([john.kemp@nokia.com](mailto:john.kemp@nokia.com))  
Eve Maler, Sun Microsystems ([eve.maler@sun.com](mailto:eve.maler@sun.com))

**Contributors:**

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

38 **Abstract:**

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

41 **Status:**

42 This is a working draft produced by the Security Services Technical Committee. Publication of  
43 this draft does not imply TC endorsement. This is an active working draft that may be updated,  
44 replaced, or obsoleted at any time. **See the Revision History for details of changes made in  
45 this revision.**

46 Committee members should submit comments and potential errata to the [security-  
47 services@lists.oasis-open.org](mailto:security-services@lists.oasis-open.org) list. Others should submit them to the [security-services-  
49 comment@lists.oasis-open.org](mailto:security-services-<br/>48 comment@lists.oasis-open.org) list (to post, you must subscribe; to subscribe, send a message  
50 to [security-services-comment-request@lists.oasis-open.org](mailto:security-services-comment-request@lists.oasis-open.org) with "subscribe" in the body) or use  
51 other OASIS-supported means of submitting comments. The committee will publish vetted errata  
on the Security Services TC web page (<http://www.oasis-open.org/committees/security/>).

52 For information on whether any patents have been disclosed that may be essential to  
53 implementing this specification, and any offers of patent licensing terms, please refer to the  
54 Intellectual Property Rights web page for the Security Services TC ([http://www.oasis-  
open.org/committees/security/ipr.php](http://www.oasis-<br/>55 open.org/committees/security/ipr.php)).

# Table of Contents

57	1 Introduction.....	6
58	1.1 Notation.....	6
59	1.2 Schema Organization and Namespaces.....	7
60	1.2.1 String and URI Values.....	7
61	1.2.2 Time Values.....	7
62	1.2.3 ID and ID Reference Values.....	7
63	1.2.4 Comparing SAML Values.....	8
64	2 SAML Assertions.....	9
65	2.1 Schema Header and Namespace Declarations.....	9
66	2.2 Simple Types.....	9
67	2.2.1 Simple Type DecisionType.....	10
68	2.3 Name Identifiers.....	10
69	2.3.1 Element <BaseIdentifier>.....	10
70	2.3.2 Element <NameIdentifier>.....	11
71	2.3.3 Element <EncryptedIdentifier>.....	11
72	2.3.4 Element <Issuer>.....	12
73	2.4 Assertions.....	12
74	2.4.1 Element <AssertionIDReference>.....	12
75	2.4.2 Element <AssertionURIReference>.....	12
76	2.4.3 Element <Assertion>.....	13
77	2.4.3.1 Element <Subject>.....	14
78	2.4.3.2 Elements <SubjectConfirmation>, <ConfirmationMethod>, and	
79	<SubjectConfirmationData>.....	15
80	2.4.3.3 Element <Conditions>.....	15
81	2.4.3.3.1 Attributes NotBefore and NotOnOrAfter.....	17
82	2.4.3.3.2 Element <Condition>.....	17
83	2.4.3.3.3 Elements <AudienceRestrictionCondition> and <Audience>.....	17
84	2.4.3.3.4 Element <DoNotCacheCondition>.....	18
85	2.4.3.3.5 Element <ProxyRestrictionCondition>.....	18
86	2.4.3.4 Element <Advice>.....	19
87	2.5 Statements.....	20
88	2.5.1 Element <Statement>.....	20
89	2.5.2 Element <AuthenticationStatement>.....	20
90	2.5.2.1 Element <SubjectLocality>.....	21
91	2.5.2.2 Element <AuthnContext>.....	22
92	2.5.3 Element <AttributeStatement>.....	22
93	2.5.3.1 Elements <AttributeDesignator> and <Attribute>.....	23
94	2.5.3.1.1 Element <AttributeValue>.....	24
95	2.5.4 Element <AuthorizationDecisionStatement>.....	24
96	2.5.4.1 Element <Action>.....	26
97	2.5.4.2 Element <Evidence>.....	26
98	3 SAML Protocols.....	28
99	3.1 Schema Header and Namespace Declarations.....	28
100	3.2 Requests and Responses.....	29
101	3.2.1 Complex Type RequestAbstractType.....	29
102	3.2.2 Complex Type StatusResponseType.....	30
103	3.2.2.1 Element <Status>.....	31
104	3.2.2.2 Element <StatusCode>.....	31

105	3.2.2.3 Element <StatusMessage>.....	33
106	3.2.2.4 Element <StatusDetail>.....	33
107	3.3 Assertion Query and Request Protocol.....	34
108	3.3.1 Element <AssertionIDRequest>.....	34
109	3.3.2 Queries.....	34
110	3.3.2.1 Element <SubjectQuery>.....	34
111	3.3.2.2 Element <AuthenticationQuery>.....	35
112	3.3.2.3 Element <AttributeQuery>.....	36
113	3.3.2.4 Element <AuthorizationDecisionQuery>.....	36
114	3.3.3 Element <Response>.....	37
115	3.3.3.1 Processing Rules.....	38
116	3.4 Authentication Request Protocol.....	38
117	3.4.1 Element <AuthnRequest>.....	39
118	3.4.1.1 Element <NameIDPolicy>.....	41
119	3.4.1.2 Element <RequestAuthnContext>.....	41
120	3.4.1.3 Element <Scoping>.....	43
121	3.4.1.4 Element <IDPList>.....	43
122	3.4.1.5 Element <IDPEntry>.....	44
123	3.4.1.6 Processing Rules.....	44
124	3.4.1.7 Proxying.....	45
125	3.4.1.7.1 Processing Rules.....	45
126	3.5 Artifact Protocol.....	46
127	3.5.1 Element <ArtifactRequest>.....	47
128	3.5.2 Element <ArtifactResponse>.....	47
129	3.5.3 Processing Rules.....	48
130	3.6 Name Identifier Management Protocol.....	48
131	3.6.1 Element <ManageNameIdentifierRequest>.....	48
132	3.6.2 Element <ManageNameIdentifierResponse>.....	49
133	3.6.3 Processing Rules.....	50
134	3.8 Single Logout Protocol.....	50
135	3.8.1 Element <LogoutRequest>.....	51
136	3.8.2 Element <LogoutResponse>.....	52
137	3.8.3 Processing Rules.....	52
138	3.8.3.1 Session Participant Rules.....	52
139	3.8.3.2 Session Authority Rules.....	53
140	3.9 Name Identifier Mapping Protocol.....	53
141	3.9.1 Element <NameIdentifierMappingRequest>.....	54
142	3.9.2 Element <NameIdentifierMappingResponse>.....	54
143	3.9.3 Processing Rules.....	55
144	4 SAML Versioning.....	56
145	4.1 SAML Specification Set Version.....	56
146	4.1.1 Schema Version.....	56
147	4.1.2 SAML Assertion Version.....	56
148	4.1.3 SAML Protocol Version.....	57
149	4.1.3.1 Request Version.....	57
150	4.1.4 Response Version.....	57
151	4.1.5 Permissible Version Combinations.....	58
152	4.2 SAML Namespace Version.....	58
153	4.2.1 Schema Evolution.....	58
154	5 SAML and XML Signature Syntax and Processing.....	59
155	5.1 Signing Assertions.....	60

156	5.2 Request/Response Signing.....	60
157	5.3 Signature Inheritance.....	60
158	5.4 XML Signature Profile.....	60
159	5.4.1 Signing Formats and Algorithms.....	60
160	5.4.2 References.....	60
161	5.4.3 Canonicalization Method.....	61
162	5.4.4 Transforms.....	61
163	5.4.5 KeyInfo.....	61
164	5.4.6 Binding Between Statements in a Multi-Statement Assertion.....	61
165	5.4.7 Example.....	61
166	6 SAML Extensions.....	64
167	6.1 Assertion Schema Extension.....	64
168	6.2 Protocol Schema Extension.....	64
169	7 SAML-Defined Identifiers.....	66
170	7.1 Authentication Method Identifiers.....	66
171	7.1.1 Password.....	66
172	7.1.2 Kerberos .....	66
173	7.1.3 Secure Remote Password (SRP).....	66
174	7.1.4 Hardware Token.....	67
175	7.1.5 SSL/TLS Certificate Based Client Authentication:.....	67
176	7.1.6 X.509 Public Key .....	67
177	7.1.7 PGP Public Key .....	67
178	7.1.8 SPKI Public Key .....	67
179	7.1.9 XKMS Public Key .....	67
180	7.1.10 XML Digital Signature .....	67
181	7.1.11 Authentication Context.....	68
182	7.1.12 Unspecified .....	68
183	7.2 Action Namespace Identifiers.....	68
184	7.2.1 Read/Write/Execute/Delete/Control.....	68
185	7.2.2 Read/Write/Execute/Delete/Control with Negation.....	68
186	7.2.3 Get/Head/Put/Post.....	69
187	7.2.4 UNIX File Permissions.....	69
188	7.3 NameIdentifier Format Identifiers.....	69
189	7.3.1 Unspecified.....	70
190	7.3.2 Email Address.....	70
191	7.3.3 X.509 Subject Name.....	70
192	7.3.4 Windows Domain Qualified Name.....	70
193	7.3.5 Kerberos Principal Name.....	70
194	7.3.6 Provider Identifier.....	70
195	7.3.7 Federated Identifier.....	70
196	7.3.8 Transient Identifier.....	71
197	7.4 Attribute NameFormat Identifiers.....	71
198	7.4.1 Unspecified.....	71
199	7.4.2 URI Reference.....	72
200	7.5 Attribute ValueType Identifiers.....	72
201	7.5.1 Unspecified Value Type.....	72
202	8 References.....	73
203	8.1 Normative References.....	73
204	8.2 Non-Normative References.....	73
205		

206

# 1 Introduction

207 This specification defines the syntax and semantics for Security Assertion Markup Language (SAML)  
208 assertions and the protocols for requesting and returning them. SAML assertions, requests, and  
209 responses are encoded in XML [XML]and use XML namespaces [XMLNS]. They are typically embedded  
210 in other structures for transport, such as HTTP form POSTs and XML-encoded SOAP messages. The  
211 SAML specification for bindings [SAMLBind] provides frameworks for this embedding and transport. Files  
212 containing just the SAML assertion schema [SAML-XSD] and protocol schema [SAMLX-XSD] are  
213 available.

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

## 1.1 Notation

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

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

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

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

226         Listings of SAML schemas appear like this.

227         Example code listings appear like this.

229 In cases of disagreement between the SAML schema documents [SAML-XSD] [SAMLX-XSD] and this  
230 specification, the schema documents take precedence.

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

- 234 • The prefix `saml:` stands for the SAML assertion namespace,  
235     `urn:oasis:names:tc:SAML:2.0:assertion`.
- 236 • The prefix `samlp:` stands for the SAML request-response protocol namespace,  
237     `urn:oasis:names:tc:SAML:2.0:protocol`.
- 238 • The prefix `ds:` stands for the W3C XML Signature namespace,  
239     <http://www.w3.org/2000/09/xmldsig#> [XMLSig-XSD].
- 240 • The prefix `xenc:` stands for the W3C XML Encryption namespace,  
241     <http://www.w3.org/2001/04/xmlenc#> [XMLEnc-XSD].
- 242 • The prefix `xsd:` stands for the W3C XML Schema namespace,  
243     <http://www.w3.org/2001/XMLSchema> [Schema1], in example listings. In schema listings, this is  
244     the default namespace and no prefix is shown.

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

## 247 1.2 Schema Organization and Namespaces

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

```
250 urn:oasis:names:tc:SAML:2.0:assertion
```

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

```
253 urn:oasis:names:tc:SAML:2.0:protocol
```

254 The assertion schema is imported into the protocol schema. Also imported into both schemas is the  
255 schema for XML Signature[XMLSig], which is associated with the following XML namespace:

```
256 http://www.w3.org/2000/09/xmldsig#
```

257 See Section SAML Namespace Version for information on SAML namespace versioning.

### 258 1.2.1 String and URI Values

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

### 265 1.2.2 Time Values

266 All SAML time values have the type **xsd:dateTime**, which is built in to the W3C XML Schema Datatypes  
267 specification [Schema1], and MUST be expressed in UTC form.

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

### 270 1.2.3 ID and ID Reference Values

271 The **xsd:ID** simple type is used to declare SAML identifiers for assertions, requests, and responses.  
272 Values declared to be of type **xsd:ID** in this specification MUST satisfy the following properties in  
273 addition to those imposed by the definition of the **xsd:ID** type itself:

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

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

283 The **xsd:NCName** simple type is used in SAML to reference identifiers of type **xsd:ID**. Note that  
284 **xsd>IDREF** cannot be used for this purpose since, in SAML, the element referred to by a SAML  
285 reference identifier might actually be defined in a document separate from that in which the identifier

286 reference is used, which violates the **xsd:IDREF** requirement that its value match the value of an ID  
287 attribute on some element in the same XML document.

## 288 **1.2.4 Comparing SAML Values**

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

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

300 Applications that compare data received in SAML documents to data from external sources **MUST** take  
301 into account the normalization rules specified for XML. Text contained within elements is normalized so  
302 that line endings are represented using linefeed characters (ASCII code 10<sub>Decimal</sub>), as described in the  
303 XML Recommendation [XML]§2.11. Attribute values defined as strings (or types derived from strings) are  
304 normalized as described in [XML] §3.3.3. All white space characters are replaced with blanks (ASCII code  
305 32<sub>Decimal</sub>).

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

---

## 2 SAML Assertions

309

310 An assertion is a package of information that supplies one or more statements made by a SAML  
311 authority. This SAML specification defines three different kinds of assertion statement that can be  
312 created by a SAML authority. As mentioned above and described in Section SAML Extensions,  
313 extensions are permitted by the SAML assertion schema, allowing user-defined extensions to assertions  
314 and statements, as well as allowing the definition of new kinds of assertion and statement. The three  
315 kinds of statement defined in this specification are:

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

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

### 2.1 Schema Header and Namespace Declarations

323

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

```
326 <schema  
327     targetNamespace="urn:oasis:names:tc:SAML:2.0:assertion"  
328     xmlns="http://www.w3.org/2001/XMLSchema"  
329     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
330     xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
331     xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"  
332     elementFormDefault="unqualified"  
333     attributeFormDefault="unqualified"  
334     blockDefault="substitution"  
335     version="2.0">  
336     <import namespace="http://www.w3.org/2000/09/xmldsig#"  
337           schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-  
338 schema.xsd"/>  
339     <import namespace="http://www.w3.org/2001/04/xmlenc#"  
340           schemaLocation="http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-  
341 schema.xsd"/>  
342     <annotation>  
343         <documentation>  
344             Document identifier: sstc-saml-schema-assertion-2.0  
345             Location: http://www.oasis-  
346 open.org/committees/documents.php?wg_abbrev=security  
347         </documentation>  
348     </annotation>  
349     ...  
350 </schema>
```

### 2.2 Simple Types

351

352 The following section defines the SAML assertion-related simple types.

## 353 2.2.1 Simple Type DecisionType

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

356 Permit

357 The specified action is permitted.

358 Deny

359 The specified action is denied.

360 Indeterminate

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

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

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

```
366 <simpleType name="DecisionType">  
367   <restriction base="string">  
368     <enumeration value="Permit"/>  
369     <enumeration value="Deny"/>  
370     <enumeration value="Indeterminate"/>  
371   </restriction>  
372 </simpleType>
```

## 373 2.3 Name Identifiers

374 The following sections define the SAML constructs that contain descriptive identifiers of subjects and  
375 assertion and message issuers.

### 376 2.3.1 Element `<BaseIdentifier>`

377 The `<BaseIdentifier>` element is an extension point that allows applications to add new kinds of  
378 identifiers. Its **BaseIdentifierAbstractType** complex type is abstract and is thus usable only as the base  
379 of a derived type. It defines the following common attributes for all identifier representations:

380 NameQualifier [Optional]

381 The security or administrative domain that qualifies the identifier of the subject. This attribute  
382 provides a means to federate identifiers from disparate user stores without collision.

383 SPNameQualifier [Optional]

384 Further qualifies an identifier with the name of the service provider or affiliation of providers  
385 which has federated the principal's identity.

386 The following schema fragment defines the `<BaseIdentifier>` element and its **BaseIdentifierType**  
387 complex type:

```
388 <element name="BaseIdentifier" type="saml:BaseIdentifierAbstractType"/>  
389 <complexType name="BaseIdentifierAbstractType" abstract="true">  
390   <complexContent>  
391     <extension base="anyType">  
392       <attribute name="NameQualifier" type="string" use="optional"/>  
393       <attribute name="SPNameQualifier" type="string" use="optional"/>  
394     </extension>
```

```
395     </complexContent>
396 </complexType>
```

## 397 2.3.2 Element <NameIdentifier>

398 The <NameIdentifier> element is of type **NameIdentifierType**, which restricts  
399 **BaseIdentifierAbstractType** to simple string content and provides additional attributes as follows:

400 Format [Optional]

401 A URI reference representing the classification of string-based identifier information. See Section  
402 NameIdentifier Format Identifiers for some URI references that MAY be used as the value of the  
403 Format attribute and their associated descriptions and processing rules. If no Format value is  
404 provided, the identifier urn:oasis:names:tc:SAML:1.0:nameid-format:unspecified (see Section  
405 Unspecified) is in effect.

406 When a Format value other than those specified in Section NameIdentifier Format Identifiers is  
407 used, the content of the <NameIdentifier> element is to be interpreted according to the  
408 specification of that format as defined outside of this specification. If not otherwise indicated by  
409 the specification of the format, issues of anonymity, pseudonymity, and the persistence of the  
410 identifier with respect to the asserting and relying parties are implementation-specific.

411 SPProvidedIdentifier [Optional]

412 The name identifier established by the service provider or affiliation of providers for the principal,  
413 if different from the primary name identifier given in the content of the <NameIdentifier>  
414 element.

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

```
417 <element name="NameIdentifier" type="saml:NameIdentifierType"/>
418 <complexType name="NameIdentifierType" mixed="false">
419   <simpleContent>
420     <restriction base="saml:BaseIdentifierAbstractType">
421       <simpleType>
422         <restriction base="string"/>
423       </simpleType>
424       <attribute name="Format" type="anyURI" use="optional"/>
425       <attribute name="SPProvidedIdentifier" type="string"
426 use="optional"/>
427     </restriction>
428   </simpleContent>
429 </complexType>
```

## 430 2.3.3 Element <EncryptedIdentifier>

431 The <EncryptedIdentifier> element extends **BaseIdentifierAbstractType** to carry the content of  
432 the element in encrypted fashion, as defined by the XML Encryption Syntax and Processing specification  
433 [XMLEnc]. The <EncryptedIdentifier> element contains the following additional elements and  
434 attributes:

435 <xenc:EncryptedData> [Required]

436 The encrypted content and associated encryption details, as defined by the XML Encryption  
437 Syntax and Processing specification [XMLEnc]. The encrypted content MUST contain an element  
438 that has a type that is derived from **BaseIdentifierAbstractType** or from **AssertionType**.

439 <xenc:EncryptedKey> [Zero or more]  
440       Wrapped decryption keys, as defined by [XMLEnc]. Each wrapped key SHOULD include a  
441       Recipient attribute that specifies the entity for whom the key has been encrypted.

442 Encrypted identifiers are intended as a privacy protection when the plain-text value passes through an  
443 intermediary; as such, the ciphertext MUST be unique to any given encryption operation. For more on  
444 such issues, see [XMLEnc]§6.3.

445 The following schema fragment defines the <EncryptedIdentifier> element and its  
446 **EncryptedIdentifierType** complex type:

```
447     <element name="EncryptedIdentifier" type="saml:EncryptedIdentifierType"/>  
448     <complexType name="EncryptedIdentifierType" mixed="false">  
449       <complexContent>  
450         <restriction base="saml:BaseIdentifierType">  
451           <sequence>  
452             <element ref="xenc:EncryptedData"/>  
453             <element ref="xenc:EncryptedKey" minOccurs="0"  
454 maxOccurs="unbounded"/>  
455           </sequence>  
456         </restriction>  
457       </complexContent>  
458     </complexType>
```

### 459 2.3.4 Element <Issuer>

460 The <Issuer> element, with complex type **NameIdentifierType**, provides information about the issuer  
461 of a SAML assertion or protocol message. The element requires the use of a string to carry the issuer's  
462 name, but permits various attributes of descriptive metadata.

463 The following schema fragment defines the <Issuer> element:

```
464 <element name="Issuer" type="saml:NameIdentifierType"/>
```

## 465 2.4 Assertions

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

### 467 2.4.1 Element <AssertionIDReference>

468 The <AssertionIDReference> element makes a reference to a SAML assertion by its unique  
469 identifier. The specific authority who issued the assertion or from whom the assertion can be obtained is  
470 not specified as part of the reference.

471 The following schema fragment defines the <AssertionIDReference> element:

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

### 473 2.4.2 Element <AssertionURIReference>

474 The <AssertionURIReference> element makes a reference to a SAML assertion by its uniform  
475 resource identifier (URI). Dereferencing the URI (in a fashion dictated by the URI) is intended to produce  
476 the assertion. See the Bindings specification [SAMLBind] for information on how this element is used in a  
477 protocol binding.

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

479 `<element name="AssertionURIReference" type="anyURI"/>`

### 480 2.4.3 Element <Assertion>

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

483 **MajorVersion** [Required]

484 The major version of this assertion. The identifier for the version of SAML defined in this  
485 specification is 2. SAML versioning is discussed in Section SAML Versioning.

486 **MinorVersion** [Required]

487 The minor version of this assertion. The identifier for the version of SAML defined in this  
488 specification is 0. SAML versioning is discussed in Section SAML Versioning.

489 **AssertionID** [Required]

490 The identifier for this assertion. It is of type **xsd:ID**, and **MUST** follow the requirements specified in  
491 Section 1.2.3 for identifier uniqueness.

492 **IssueInstant** [Required]

493 The time instant of issue in UTC, as described in Section Time Values.

494 **<Issuer>** [Required]

495 The SAML authority that is making the claim(s) in the assertion. The issuer identity **SHOULD** be  
496 unambiguous to the intended relying parties. If the **Format** attribute is omitted, the identifier  
497 `urn:oasis:names:tc:SAML:1.0:nameid-format:unspecified` (see section 7.3.1) is  
498 assumed.

499 This specification defines no relationship between the entity represented by this element and the  
500 signer of the assertion (if any). Any such requirements imposed by a relying party that consumes the  
501 assertion or to specific profiles are application-specific.

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

503 An XML Signature that authenticates the assertion, as described in Section SAML and XML  
504 Signature Syntax and Processing.

505 **<Subject>** [Optional]

506 The subject of the statement(s) in the assertion.

507 **<Conditions>** [Optional]

508 Conditions that **MUST** be taken into account in assessing the validity of and/or using the assertion.

509 **<Advice>** [Optional]

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

512 Zero or more of the following statement elements:

513 **<Statement>**

514 A statement defined in an extension schema.

515 **<AuthenticationStatement>**

516 An authentication statement.

517 <AuthorizationDecisionStatement>

518 An authorization decision statement.

519 <AttributeStatement>

520 An attribute statement.

521 An assertion with no statements **MUST** contain a <Subject> element; a assertion containing no  
522 statements binds the name identifier in the subject to a principal. Otherwise <Subject>, if present,  
523 identifies the subject of all of the statements in the assertion. If omitted, then the statements in the  
524 assertion are assumed to identify (implicitly or explicitly) the subject or subjects to which they apply in an  
525 application- or profile-specific manner.

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

```
528 <element name="Assertion" type="saml:AssertionType"/>
529 <complexType name="AssertionType">
530   <sequence>
531     <element ref="saml:Issuer"/>
532     <element ref="ds:Signature" minOccurs="0"/>
533     <element ref="saml:Subject" minOccurs="0"/>
534     <element ref="saml:Conditions" minOccurs="0"/>
535     <element ref="saml:Advice" minOccurs="0"/>
536     <choice minOccurs="0" maxOccurs="unbounded">
537       <element ref="saml:Statement"/>
538       <element ref="saml:AuthenticationStatement"/>
539       <element ref="saml:AuthorizationDecisionStatement"/>
540       <element ref="saml:AttributeStatement"/>
541     </choice>
542   </sequence>
543   <attribute name="MajorVersion" type="integer" use="required"/>
544   <attribute name="MinorVersion" type="integer" use="required"/>
545   <attribute name="AssertionID" type="ID" use="required"/>
546   <attribute name="IssueInstant" type="dateTime" use="required"/>
547 </complexType>
```

### 548 2.4.3.1 Element <Subject>

549 The optional <Subject> element specifies the principal that is the subject of all of the (zero or more)  
550 statements in the assertion. It contains a name identifier, a series of one or more subject confirmations,  
551 or both:

552 <NameIdentifier>, <EncryptedIdentifier>, or <BaseIdentifier>

553 Identifies the subject.

554 <SubjectConfirmation>

555 Information that allows the subject to be authenticated. If more than one subject confirmation is  
556 provided, then usage of any one of them is sufficient to confirm the subject for the purpose of  
557 applying the assertion.

558 If the <Subject> element contains both an identifier and one or more subject confirmations, the SAML  
559 authority is asserting that if the SAML relying party performs the specified <SubjectConfirmation>, it  
560 can treat the entity presenting the assertion to the relying party as the entity that the SAML authority  
561 associates with the name identifier for the purposes of processing the assertion. A <Subject> element  
562 **SHOULD NOT** identify more than one principal. The following schema fragment defines the <Subject>  
563 element and its **SubjectType** complex type:

```
564 <element name="Subject" type="saml:SubjectType"/>
565 <complexType name="SubjectType">
566   <choice>
```

```

567         <sequence>
568             <choice>
569                 <element ref="saml:BaseIdentifier"/>
570                 <element ref="saml:NameIdentifier"/>
571                 <element ref="saml:EncryptedIdentifier"/>
572             </choice>
573             <element ref="saml:SubjectConfirmation" minOccurs="0"
574 maxOccurs="unbounded"/>
575         </sequence>
576         <element ref="saml:SubjectConfirmation" maxOccurs="unbounded"/>
577     </choice>
578 </complexType>

```

### 579 2.4.3.2 Elements <SubjectConfirmation>, <ConfirmationMethod>, and 580 <SubjectConfirmationData>

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

583 <ConfirmationMethod> [Required]

584 A URI reference that identifies a protocol to be used to authenticate the subject. URI references  
585 identifying SAML-defined confirmation methods are currently defined with the SAML profiles in the  
586 SAML profiles specification [SAMLProf]. Additional methods may be added by defining new URIs and  
587 profiles or by private agreement.

588 <SubjectConfirmationData> [Optional]

589 Additional authentication information to be used by a specific authentication protocol. For example,  
590 typical content of this element might be a <ds:KeyInfo> element as defined in the XML Signature  
591 Syntax and Processing specification [XMLSig], which identifies a cryptographic key.

592 The following schema fragment defines the <SubjectConfirmation> element and its  
593 **SubjectConfirmationType** complex type, along with the <SubjectConfirmationData> element and  
594 the <ConfirmationMethod> element:

```

595 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>
596 <complexType name="SubjectConfirmationType">
597     <sequence>
598         <element ref="saml:ConfirmationMethod"/>
599         <element ref="saml:SubjectConfirmationData" minOccurs="0"/>
600     </sequence>
601 </complexType>
602 <element name="SubjectConfirmationData" type="anyType"/>
603 <element name="ConfirmationMethod" type="anyURI"/>

```

### 604 2.4.3.3 Element <Conditions>

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

606 NotBefore [Optional]

607 Specifies the earliest time instant at which the assertion is valid. The time value is encoded in UTC  
608 as described in Section Time Values.

609 NotOnOrAfter [Optional]

610 Specifies the time instant at which the assertion has expired. The time value is encoded in UTC as  
611 described in Section Time Values.

612 <Condition> [Any Number]

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

614 <AudienceRestrictionCondition> [Any Number]

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

616 <DoNotCacheCondition> [Any Number]

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

619 <ProxyRestrictionCondition> [Any Number]

620 Specifies limitations that the asserting party imposes on relying parties that wish to issue subsequent  
621 assertions of their own on the basis of the information contained in the original assertion.

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

```
624 <element name="Conditions" type="saml:ConditionsType"/>
625 <complexType name="ConditionsType">
626   <choice minOccurs="0" maxOccurs="unbounded">
627     <element ref="saml:AudienceRestrictionCondition"/>
628     <element ref="saml:DoNotCacheCondition">
629     <element ref="saml:ProxyRestrictionCondition"/>
630     <element ref="saml:Condition"/>
631   </choice>
632   <attribute name="NotBefore" type="dateTime" use="optional"/>
633   <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
634 </complexType>
```

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

- 639 1. If no sub-elements or attributes are supplied in the <Conditions> element, then the assertion is  
640 considered to be **Valid**.
- 641 2. If any sub-element or attribute of the <Conditions> element is determined to be invalid, then the  
642 assertion is **Invalid**.
- 643 3. If any sub-element or attribute of the <Conditions> element cannot be evaluated, then the validity  
644 of the assertion cannot be determined and is deemed to be **Indeterminate**.
- 645 4. If all sub-elements and attributes of the <Conditions> element are determined to be **Valid**, then  
646 the assertion is considered to be **Valid**.

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

651 Note that an assertion that has validity status **Valid** may not be trustworthy for reasons such as not being  
652 issued by a trustworthy SAML authority or not being authenticated by a trustworthy means.

653 Also note that some conditions may not directly impact the validity of the containing assertion (they  
654 always evaluate to **Valid**), but may restrict the behavior of relying parties with respect to the use of the  
655 assertion.

### 656 **2.4.3.3.1 Attributes NotBefore and NotOnOrAfter**

657 The `NotBefore` and `NotOnOrAfter` attributes specify time limits on the validity of the assertion within  
658 the context of its profile of use. They do not guarantee that the statements in the assertion will be valid  
659 throughout the validity period.

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

662 If the value for either `NotBefore` or `NotOnOrAfter` is omitted it is considered unspecified. If the  
663 `NotBefore` attribute is unspecified (and if any other conditions that are supplied evaluate to **Valid**), the  
664 assertion is valid at any time before the time instant specified by the `NotOnOrAfter` attribute. If the  
665 `NotOnOrAfter` attribute is unspecified (and if any other conditions that are supplied evaluate to **Valid**),  
666 the assertion is valid from the time instant specified by the `NotBefore` attribute with no expiry. If neither  
667 attribute is specified (and if any other conditions that are supplied evaluate to **Valid**), the assertion is  
668 valid at any time.

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

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

### 673 **2.4.3.3.2 Element <Condition>**

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

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

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

### 680 **2.4.3.3.3 Elements <AudienceRestrictionCondition> and <Audience>**

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

686 `<Audience>`

687 A URI reference that identifies an intended audience. The URI reference MAY identify a document  
688 that describes the terms and conditions of audience membership. It MAY also contain the unique  
689 identifier of a SAML system entity, as described by the `<NameIdentifier>` Format URI of  
690 `urn:oasis:names:tc:SAML:2.0:nameid-format:provider`.

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

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

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

```
701 <element name="AudienceRestrictionCondition"  
702 type="saml:AudienceRestrictionConditionType"/>  
703 <complexType name="AudienceRestrictionConditionType">  
704 <complexContent>  
705 <extension base="saml:ConditionAbstractType">  
706 <sequence>  
707 <element ref="saml:Audience" maxOccurs="unbounded"/>  
708 </sequence>  
709 </extension>  
710 </complexContent>  
711 </complexType>  
712 <element name="Audience" type="anyURI"/>
```

#### 713 2.4.3.3.4 Element `<DoNotCacheCondition>`

714 Indicates that the assertion SHOULD be used immediately by the relying party and MUST NOT be  
715 retained for future use. Note that no relying party is required to perform caching. However, any that do so  
716 MUST observe this condition. This condition conveys one-time-use semantics, and is independent from  
717 the `NotBefore` and `NotOnOrAfter` condition information.

718 A SAML authority SHOULD NOT include more than one `<DoNotCacheCondition>` element within a  
719 `<Conditions>` element of an assertion. If multiple `<DoNotCacheCondition>` elements appear within  
720 a `<Conditions>` element, a Relying Party MUST treat the multiple elements as though a single  
721 `<DoNotCacheCondition>` element was specified.

722 For the purposes of determining the validity of the `<Conditions>` element, the  
723 `<DoNotCacheCondition>` is considered to always be valid.

724 The following schema fragment defines the `<DoNotCacheCondition>` element and its  
725 **DoNotCacheConditionType** complex type:

```
726 <element name="DoNotCacheCondition" type="saml:DoNotCacheConditionType"/>  
727 <complexType name="DoNotCacheConditionType">  
728 <complexContent>  
729 <extension base="saml:ConditionAbstractType"/> </complexContent>  
730 </complexType>
```

#### 731 2.4.3.3.5 Element `<ProxyRestrictionCondition>`

732 Specifies limitations that the asserting party imposes on relying parties that wish to issue subsequent  
733 assertions of their own on the basis of the information contained in the original assertion. A relying party  
734 MUST NOT issue an assertion that itself violates the restrictions specified in this condition on the basis  
735 of an assertion containing such a condition.

736 The `<ProxyRestrictionCondition>` element contains the following elements and attributes:

737 Count [Optional]

738 Specifies the number of indirections that MAY exist between this assertion and an assertion which  
739 has ultimately been issued on the basis of it.

740 <Audience> [Zero or More]

741 Specifies the set of audiences to whom new assertions MAY be issued on the basis of this assertion.

742 A Count value of zero indicates that a relying party MUST NOT issue an assertion to another relying  
743 party on the basis of this assertion. If greater than zero, any assertions so issued MUST themselves  
744 contain a <ProxyRestrictionCondition> element with a Count value of at most one less than this  
745 value.

746 If no <Audience> elements are specified, then no restrictions are made upon the relying parties to  
747 whom subsequent assertions can be issued. Otherwise, any assertions so issued MUST themselves  
748 contain an <AudienceRestrictionCondition> element with at least one of the <Audience>  
749 elements present in the previous <ProxyRestrictionCondition> element, and no <Audience>  
750 elements present that were not in the previous <ProxyRestrictionCondition> element.

751 A SAML authority SHOULD NOT include more than one <ProxyRestrictionCondition> element  
752 within a <Conditions> element of an assertion. If multiple <ProxyRestrictionCondition>  
753 elements appear within a <Conditions> element, a relying party MUST treat the multiple elements as  
754 though a single <ProxyRestrictionCondition> element was specified, with a Count value equal to  
755 the lowest of any specified, and the set of <Audience> elements consisting of the union of the elements  
756 specified.

757 For the purposes of determining the validity of the <Conditions> element, the  
758 <ProxyRestrictionCondition> is considered to always be valid.

759 The following schema fragment defines the <ProxyRestrictionCondition> element and its  
760 **ProxyRestrictionConditionType** complex type:

```
761 <element name="ProxyRestrictionCondition"  
762 type="saml:ProxyRestrictionConditionType"/>  
763 <complexType name="ProxyRestrictionConditionType">  
764   <complexContent>  
765     <extension base="saml:ConditionAbstractType">  
766       <sequence>  
767         <element ref="saml:Audience" minOccurs="0"  
768 maxOccurs="unbounded"/>  
769       </sequence>  
770       <attribute name="Count" type="nonNegativeInteger"  
771 use="optional"/>  
772     </extension>  
773   </complexContent>  
774 </complexType>
```

#### 775 2.4.3.4 Element <Advice>

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

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

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

- 783 • Include evidence supporting the assertion claims to be cited, either directly (through incorporating  
784 the claims) or indirectly (by reference to the supporting assertions).
- 785 • State a proof of the assertion claims.

- 786 • Specify the timing and distribution points for updates to the assertion.

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

```
788 <element name="Advice" type="saml:AdviceType"/>
789 <complexType name="AdviceType">
790   <choice minOccurs="0" maxOccurs="unbounded">
791     <element ref="saml:AssertionIDReference"/>
792     <element ref="saml:AssertionURIReference"/>
793     <element ref="saml:Assertion"/>
794     <any namespace="##other" processContents="lax"/>
795   </choice>
796 </complexType>
```

## 797 2.5 Statements

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

### 799 2.5.1 Element <Statement>

800 The <Statement> element is an extension point that allows other assertion-based applications to reuse  
801 the SAML assertion framework. Its **StatementAbstractType** complex type is abstract and is thus usable  
802 only as the base of a derived type. This element has an optional attribute:

803 **SessionIndex** [Optional]

804     Indexes a particular session between the subject and the authority issuing this statement. The value  
805     of the attribute SHOULD be a small, positive integer, but may be any string of text. This value MUST  
806     NOT be a unique value identifying a principal's session at the authority.

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

```
809 <element name="Statement" type="saml:StatementAbstractType"/>
810 <complexType name="StatementAbstractType" abstract="true">
811   <attribute name="SessionIndex" type="string" use="optional"/>
812 </complexType>
```

#### 813 2.5.1.1

### 814 2.5.2 Element <AuthenticationStatement>

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

819 **AuthenticationMethod** [Required]

820     A URI reference that specifies the type of authentication that took place. URI references identifying  
821     common authentication protocols are listed in Section Authentication Method Identifiers. A value of  
822     urn:oasis:names:tc:SAML:2.0:am:authncontext indicates that an <AuthnContext>  
823     element is included in the statement that describes further details of the authentication.

824 **AuthenticationInstant** [Required]

825     Specifies the time at which the authentication took place. The time value is encoded in UTC as  
826     described in Section Time Values.

827 <SubjectLocality> [Optional]

828 Specifies the DNS domain name and IP address for the system from which the subject was  
829 apparently authenticated.

830 <AuthnContext> [Optional]

831 The context used by the identity provider in the authentication event that yielded this statement.  
832 Contains a reference to an authentication context class, an authentication context statement or  
833 statement reference, or both. See the Authentication Context specification [SAMLAuthnCxt] for a full  
834 description of authentication context information.

835 **Note:** The <AuthorityBinding> element and its corresponding type were removed  
836 from <AuthenticationStatement> for V2.0 of SAML.

837 <AuthenticationStatement> elements MUST contain a SessionIndex value, conforming to the  
838 rules specified in section 2.5.1.

839 Assertions containing <AuthenticationStatement> elements MUST contain a <Subject> element.

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

```
842 <element name="AuthenticationStatement"  
843         type="saml:AuthenticationStatementType"/>  
844 <complexType name="AuthenticationStatementType">  
845   <complexContent>  
846     <extension base="saml:StatementAbstractType">  
847       <sequence>  
848         <element ref="saml:SubjectLocality" minOccurs="0"/>  
849         <element ref="saml:AuthnContext" minOccurs="0"/>  
850       </sequence>  
851       <attribute name="AuthenticationMethod" type="anyURI"  
852 use="required"/>  
853       <attribute name="AuthenticationInstant" type="dateTime"  
854 use="required"/>  
855     </extension>  
856   </complexContent>  
857 </complexType>
```

### 858 2.5.2.1 Element <SubjectLocality>

859 The <SubjectLocality> element specifies the DNS domain name and IP address for the system  
860 from which the subject was authenticated. It has the following attributes:

861 IPAddress [Optional]

862 The IP address of the system from which the subject was authenticated.

863 DNSAddress [Optional]

864 The DNS address of the system from which the subject was authenticated.

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

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

```
869 <element name="SubjectLocality"  
870         type="saml:SubjectLocalityType"/>  
871 <complexType name="SubjectLocalityType">  
872   <attribute name="IPAddress" type="string" use="optional"/>
```

```
873     <attribute name="DNSAddress" type="string" use="optional"/>
874 </complexType>
```

### 875 2.5.2.2 Element <AuthnContext>

876 The <AuthnContext> element specifies the context of an authentication event with a context class  
877 reference, a context statement or statement reference, or both. It's complex **AuthnContextType** has the  
878 following elements:

879 <AuthnContextClassRef> [Optional]

880 A URI identifying an authentication context class that describes the authentication context statement  
881 that follows.

882 <AuthnContextStatement> or <AuthnContextStatementRef> [Optional]

883 Either an authentication context statement, or a URI that identifies such a statement. The URI MAY  
884 directly resolve into an XML document containing the referenced statement.

885 The following schema fragment defines the <AuthnContext> element and its **AuthnContextType**  
886 complex type:

```
887 <element name="AuthnContext" type="saml:AuthnContextType"/>
888 <complexType name="AuthnContextType">
889   <sequence>
890     <element ref="saml:AuthnContextClassRef" minOccurs="0"/>
891     <choice minOccurs="0">
892       <element ref="saml:AuthnContextStatement"/>
893       <element ref="saml:AuthnContextStatementRef"/>
894     </choice>
895   </sequence>
896 </complexType>
897 <element name="AuthnContextClassRef" type="anyURI"/>
898 <element name="AuthnContextStatementRef" type="anyURI"/>
899 <element name="AuthnContextStatement" type="anyType"/>
```

### 900 2.5.3 Element <AttributeStatement>

901 The <AttributeStatement> element describes a statement by the SAML authority asserting that the  
902 statement's subject is associated with the specified attributes. It is of type **AttributeStatementType**,  
903 which extends **StatementAbstractType** with the addition of the following element:

904 <Attribute> [One or More]

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

906 Assertions containing <AttributeStatement> elements MUST contain a <Subject> element.

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

```
909 <element name="AttributeStatement" type="saml:AttributeStatementType"/>
910 <complexType name="AttributeStatementType">
911   <complexContent>
912     <extension base="saml:StatementAbstractType">
913       <sequence>
914         <element ref="saml:Attribute"
915 maxOccurs="unbounded"/>
916       </sequence>
917     </extension>
918   </complexContent>
919 </complexType>
```

### 920 2.5.3.1 Elements <AttributeDesignator> and <Attribute>

921 The <AttributeDesignator> element identifies an attribute name within an attribute namespace. It  
922 has the **AttributeDesignatorType** complex type. It is used in an attribute query to request that attribute  
923 values within a specific namespace be returned (see Section Element <AttributeQuery> for more  
924 information). The <AttributeDesignator> element contains the following XML attributes:

925 Name [Required]

926 The name of the attribute.

927 NameFormat [Optional]

928 A URI reference representing the classification of the attribute name for purposes of interpreting  
929 the name. See Section 7.x for some URI references that MAY be used as the value of the  
930 NameFormat attribute and their associated descriptions and processing rules. If no  
931 NameFormat value is provided, the identifier urn:oasis:names:tc:SAML:2.0:atname-  
932 format:unspecified (see Section 7.x) is in effect.

933 ValueType [Optional]

934 A URI reference representing the datatype of the desired or supplied attribute. If no ValueType  
935 value is provided, the identifier urn:oasis:names:tc:saml:2.0:valuetype:format:unspecified (see  
936 Section 7.x) is in effect. Note that datatypes specified on the <AttributeValue> element  
937 using xsi:type have no SAML defined relationship with ValueType. The ValueType setting  
938 (default or explicit) in an attribute query using the <AttributeDesignator> element MUST be  
939 exactly matched (in addition to other exact matches as described in Section x) in order for an  
940 attribute to be returned.

941 Arbitrary attributes

942 This complex type uses an <xsd:anyAttribute> extension point to allow for arbitrary XML  
943 attributes to be added to <AttributeDesignator> constructs without the need for an explicit  
944 schema extension. This allows additional fields to be added as needed to supply additional  
945 parameters to be used in an attribute query. SAML extensions MUST NOT add local (non-  
946 namespace-qualified) XML attributes to the AttributeType complex type or to any element bound  
947 to this type or a derivation of it; such attributes are reserved for future maintenance and  
948 enhancement of SAML itself.

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

```
951 <element name="AttributeDesignator" type="saml:AttributeDesignatorType"/>  
952 <complexType name="AttributeDesignatorType">  
953   <attribute name="Name" type="string" use="required"/>  
954   <attribute name="NameFormat" type="anyURI" use="optional"/>  
955   <attribute name="ValueType" type="anyURI" use="optional"/>  
956   <anyAttribute/>  
957 </complexType>
```

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

961 <AttributeValue> [Any Number]

962 The value of the attribute. If an attribute contains more than one discrete value, it is  
963 RECOMMENDED that each value appear in its own <AttributeValue> element. If the attribute  
964 exists but has no value, then the <AttributeValue> element MUST be omitted. If more than one  
965 <AttributeValue> element is supplied for an attribute, and any of the elements have a datatype  
966 assigned through xsi:type, then all of the <AttributeValue> elements must have the identical

967 datatype assigned.

#### 968 Arbitrary attributes

969 This complex type inherits from **AttributeDesignatorType** the ability to add arbitrary XML  
970 attributes to <Attribute> constructs without the need for an explicit schema extension. This  
971 allows additional fields to be added as needed to supply the context in which the attribute should  
972 be understood. SAML extensions MUST NOT add local (non-namespace-qualified) XML  
973 attributes to the AttributeType complex type or to any element bound to this type or a derivation  
974 of it; such attributes are reserved for future maintenance and enhancement of SAML itself.

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

```
976 <element name="Attribute" type="saml:AttributeType"/>
977 <complexType name="AttributeType">
978   <complexContent>
979     <extension base="saml:AttributeDesignatorType">
980       <sequence>
981         <element ref="saml:AttributeValue" minOccurs="0"
982 maxOccurs="unbounded"/>
983       </sequence>
984     </extension>
985   </complexContent>
986 </complexType>
```

#### 987 2.5.3.1.1 Element <AttributeValue>

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

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

994 **Note:** Specifying a datatype on <AttributeValue> using **xsi:type** will require the  
995 presence of the extension schema that defines the datatype in order for schema  
996 processing to proceed.

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

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

#### 999 2.5.4 Element <AuthorizationDecisionStatement>

1000 **Note:** The <AuthorizationDecisionStatement> feature has been frozen as of  
1001 SAML V2.0, with no future enhancements planned. Users who require additional  
1002 functionality may want to consider the eXtensible Access Control Markup Language  
1003 [XACML], which offers enhanced authorization decision features.

1004 The <AuthorizationDecisionStatement> element describes a statement by the SAML authority  
1005 asserting that a request for access by the statement's subject to the specified resource has resulted in  
1006 the specified authorization decision on the basis of some optionally specified evidence.

1007 The resource is identified by means of a URI reference. In order for the assertion to be interpreted  
1008 correctly and securely, the SAML authority and SAML relying party MUST interpret each URI reference  
1009 in a consistent manner. Failure to achieve a consistent URI reference interpretation can result in different  
1010 authorization decisions depending on the encoding of the resource URI reference. Rules for normalizing  
1011 URI references are to be found in IETF RFC 2396 [RFC 2396] §6:

1012 In general, the rules for equivalence and definition of a normal form, if any, are scheme  
1013 dependent. When a scheme uses elements of the common syntax, it will also use the common  
1014 syntax equivalence rules, namely that the scheme and hostname are case insensitive and a URL  
1015 with an explicit ":port", where the port is the default for the scheme, is equivalent to one where  
1016 the port is elided.

1017 To avoid ambiguity resulting from variations in URI encoding SAML system entities SHOULD employ the  
1018 URI normalized form wherever possible as follows:

- 1019 • SAML authorities SHOULD encode all resource URI references in normalized form.
- 1020 • Relying parties SHOULD convert resource URI references to normalized form prior to processing.

1021 Inconsistent URI reference interpretation can also result from differences between the URI reference  
1022 syntax and the semantics of an underlying file system. Particular care is required if URI references are  
1023 employed to specify an access control policy language. The following security conditions should be  
1024 satisfied by the system which employs SAML assertions:

- 1025 • Parts of the URI reference syntax are case sensitive. If the underlying file system is case insensitive,  
1026 a requester SHOULD NOT be able to gain access to a denied resource by changing the case of a  
1027 part of the resource URI reference.
- 1028 • Many file systems support mechanisms such as logical paths and symbolic links, which allow users  
1029 to establish logical equivalences between file system entries. A requester SHOULD NOT be able to  
1030 gain access to a denied resource by creating such an equivalence.

1031 The `<AuthorizationDecisionStatement>` element is of type  
1032 **AuthorizationDecisionStatementType**, which extends **StatementAbstractType** with the addition of the  
1033 following elements (in order) and attributes:

1034 **Resource** [Required]

1035 A URI reference identifying the resource to which access authorization is sought. It is permitted for  
1036 this attribute to have the value of the empty URI reference (""), and the meaning is defined to be "the  
1037 start of the current document", as specified by IETF RFC 2396 [RFC 2396] §4.2.

1038 **Decision** [Required]

1039 The decision rendered by the SAML authority with respect to the specified resource. The value is of  
1040 the **DecisionType** simple type.

1041 **<Action>** [One or more]

1042 The set of actions authorized to be performed on the specified resource.

1043 **<Evidence>** [Optional]

1044 A set of assertions that the SAML authority relied on in making the decision.

1045 Assertions containing `<AuthorizationDecisionStatement>` elements MUST contain a `<Subject>`  
1046 element.

1047 The following schema fragment defines the `<AuthorizationDecisionStatement>` element and its  
1048 **AuthorizationDecisionStatementType** complex type:

```
1049 <element name="AuthorizationDecisionStatement"  
1050 type="saml:AuthorizationDecisionStatementType"/>  
1051 <complexType name="AuthorizationDecisionStatementType">  
1052 <complexContent>  
1053 <extension base="saml:StatementAbstractType">  
1054 <sequence>  
1055 <element ref="saml:Action" maxOccurs="unbounded"/>
```

```

1056         <element ref="saml:Evidence" minOccurs="0"/>
1057     </sequence>
1058     <attribute name="Resource" type="anyURI" use="required"/>
1059     <attribute name="Decision" type="saml:DecisionType"
1060 use="required"/>
1061     </extension>
1062 </complexContent>
1063 </complexType>

```

#### 1064 2.5.4.1 Element <Action>

1065 The <Action> element specifies an action on the specified resource for which permission is sought. It  
 1066 has the following attribute and string-data content:

1067 Namespace [Optional]

1068 A URI reference representing the namespace in which the name of the specified action is to be  
 1069 interpreted. If this element is absent, the namespace urn:oasis:names:tc:SAML:1.0:action:rwdc-  
 1070 negotiation specified in Section Read/Write/Execute/Delete/Control with Negation is in effect.

1071 *string data* [Required]

1072 An action sought to be performed on the specified resource.

1073 The following schema fragment defines the <Action> element and its **ActionType** complex type:

```

1074 <element name="Action" type="saml:ActionType"/>
1075 <complexType name="ActionType">
1076     <simpleContent>
1077         <extension base="string">
1078             <attribute name="Namespace" type="anyURI"/>
1079         </extension>
1080     </simpleContent>
1081 </complexType>

```

#### 1082 2.5.4.2 Element <Evidence>

1083 The <Evidence> element contains an assertion or assertion reference that the SAML authority relied on  
 1084 in issuing the authorization decision. It has the **EvidenceType** complex type. It contains a mixture of one  
 1085 or more of the following elements:

1086 <AssertionIDReference> [Any number]

1087 Specifies an assertion by reference to the value of the assertion's `AssertionID` attribute.

1088 <AssertionURIReference> [Any number]

1089 Specifies an assertion by reference to a URI.

1090 <Assertion> [Any number]

1091 Specifies an assertion by value.

1092 Providing an assertion as evidence MAY affect the reliance agreement between the SAML relying party  
 1093 and the SAML authority making the authorization decision. For example, in the case that the SAML  
 1094 relying party presented an assertion to the SAML authority in a request, the SAML authority MAY use  
 1095 that assertion as evidence in making its authorization decision without endorsing the <Evidence>  
 1096 element's assertion as valid either to the relying party or any other third party.

1097 The following schema fragment defines the <Evidence> element and its **EvidenceType** complex type:

```

1098 <element name="Evidence" type="saml:EvidenceType"/>
1099 <complexType name="EvidenceType">

```

```
1100     <choice maxOccurs="unbounded">
1101         <element ref="saml:AssertionIDReference"/>
1102         <element ref="saml:AssertionURIReference"/>
1103         <element ref="saml:Assertion"/>
1104     </choice>
1105 </complexType>
```

## 3 SAML Protocols

1106

1107 SAML assertions and related/supporting messages MAY be generated and exchanged using a variety of  
1108 protocols. The bindings specification for SAML [SAMLBind] describes specific means of transporting  
1109 queries, assertions, and other messages using existing widely deployed transport protocols.

1110 Specific SAML request and response messages derive from common types. The requester sends an  
1111 element derived from **RequestAbstractType** to a SAML responder, and the responder generates an  
1112 element adhering to or deriving from **StatusResponseType**, as shown in Figure 1.

1113



1115

Figure 1: SAML Request-Response Protocol

1116 The protocols defined by SAML achieve the following actions:

- 1117 • Returning one or more requested assertions (includes a direct request of the desired assertions, as  
1118 well as querying for assertions that meet particular criteria)
- 1119 • Performing authentication on request and returning the corresponding assertion
- 1120 • Registering a name identifier or terminating a name registration on request
- 1121 • Retrieve a protocol message that has been requested by means of an artifact
- 1122 • Performing a near-simultaneous logout of a collection of related sessions ("single logout") on  
1123 request
- 1124 • Providing a name identifier mapping on request

### 3.1 Schema Header and Namespace Declarations

1125

1126 The following schema fragment defines the XML namespaces and other header information for the  
1127 protocol schema:

1128

1129

1130

1131

1132

1133

1134

1135

1136

1137

1138

1139

1140

1141

1142

1143

1144

1145

```
<schema
  targetNamespace="urn:oasis:names:tc:SAML:2.0:protocol"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  elementFormDefault="unqualified"
  attributeFormDefault="unqualified"
  blockDefault="substitution"
  version="2.0">
  <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
    schemaLocation="sstc-saml-schema-assertion-2.0.xsd"/>
  <import namespace="http://www.w3.org/2000/09/xmldsig#"
    schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-
schema.xsd"/>
  <annotation>
    <documentation>
      Document identifier: sstc-saml-schema-protocol-2.0
```

```
1146         Location: http://www.oasis-
1147 open.org/committees/documents.php?wg_abbrev=security
1148         </documentation>
1149     </annotation>
1150     ...
1151 </schema>
```

## 1152 3.2 Requests and Responses

1153 The following sections define the SAML constructs that underlie request and response messages.

### 1154 3.2.1 Complex Type RequestAbstractType

1155 All SAML requests are of types that are derived from the abstract **RequestAbstractType** complex type.  
1156 This type defines common attributes and elements that are associated with all SAML requests:

1157 **RequestID** [Required]

1158 An identifier for the request. It is of type **xsd:ID** and MUST follow the requirements specified in  
1159 Section 1.2.3 for identifier uniqueness. The values of the **RequestID** attribute in a request and the  
1160 **InResponseTo** attribute in the corresponding response MUST match.

1161 **MajorVersion** [Required]

1162 The major version of this request. The identifier for the version of SAML defined in this specification  
1163 is 2. SAML versioning is discussed in Section SAML Versioning.

1164 **MinorVersion** [Required]

1165 The minor version of this request. The identifier for the version of SAML defined in this specification  
1166 is 0. SAML versioning is discussed in Section SAML Versioning.

1167 **IssueInstant** [Required]

1168 The time instant of issue of the request. The time value is encoded in UTC as described in Section  
1169 Time Values.

1170 **Consent** [Optional]

1171 Indicates whether or not consent has been obtained from a user in the sending this request.

1172 **<Issuer>** [Optional]

1173 Identifies the entity that generated the request message.

1174 **<ds:Signature>** [Optional]

1175 An XML Signature that authenticates the request, as described in Section SAML and XML Signature  
1176 Syntax and Processing.

1177 **<RelayState>** [Optional]

1178 This contains state information that MUST be relayed back in the associated response.

1179 **<Extensions>** [Optional]

1180 This contains optional protocol message extension elements that are agreed upon between the  
1181 communicating parties.

1182 **Note:** The **<RespondWith>** element has been removed from **<Request>** for V2.0 of  
1183 SAML.

1184 The following schema fragment defines the **RequestAbstractType** complex type:

```

1185 <complexType name="RequestAbstractType" abstract="true">
1186   <sequence>
1187     <element ref="saml:Issuer" minOccurs="0"/>
1188     <element ref="ds:Signature" minOccurs="0"/>
1189
1190     <element ref="samlp:Extensions" minOccurs="0"/>
1191   </sequence>
1192   <attribute name="RequestID" type="ID" use="required"/>
1193   <attribute name="MajorVersion" type="integer" use="required"/>
1194   <attribute name="MinorVersion" type="integer" use="required"/>
1195   <attribute name="IssueInstant" type="dateTime" use="required"/>
1196   <attribute name="Consent" type="anyURI" use="optional"/>
1197 </complexType>
1198 <element name="Extensions" type="samlp:ExtensionsType"/>
1199 <complexType name="ExtensionsType">
1200   <sequence>
1201     <any namespace="##other" processContents="lax"
1202     maxOccurs="unbounded"/>
1203   </sequence>
1204 </complexType>

```

### 1205 3.2.1.1

## 1206 3.2.2 Complex Type StatusResponseType

1207 All SAML responses are of types that are derived from the **StatusResponseType** complex type. This  
1208 type defines common attributes and elements that are associated with all SAML responses:

#### 1209 ResponseID [Required]

1210 An identifier for the response. It is of type **xsd:ID**, and MUST follow the requirements specified in  
1211 Section 1.2.3 for identifier uniqueness.

#### 1212 InResponseTo [Optional]

1213 A reference to the identifier of the request to which the response corresponds, if any. If the response  
1214 is not generated in response to a request, or if the RequestID attribute value of a request cannot be  
1215 determined (because the request is malformed), then this attribute MUST NOT be present.  
1216 Otherwise, it MUST be present and its value MUST match the value of the corresponding  
1217 RequestID attribute value.

#### 1218 MajorVersion [Required]

1219 The major version of this response. The identifier for the version of SAML defined in this  
1220 specification is 2. SAML versioning is discussed in Section SAML Versioning.

#### 1221 MinorVersion [Required]

1222 The minor version of this response. The identifier for the version of SAML defined in this  
1223 specification is 0. SAML versioning is discussed in Section SAML Versioning.

#### 1224 IssueInstant [Required]

1225 The time instant of issue of the response. The time value is encoded in UTC as described in Section  
1226 Time Values.

#### 1227 Recipient [Optional]

1228 The intended recipient of this response. This is useful to prevent malicious forwarding of responses  
1229 to unintended recipients, a protection that is required by some use profiles. It is set by the generator  
1230 of the response to a URI reference that identifies the intended recipient. If present, the actual  
1231 recipient MUST check that the URI reference identifies the recipient or a resource managed by the  
1232 recipient. If it does not, the response MUST be discarded.

- 1233 <Issuer> [Optional]  
 1234 Identifies the entity that generated the response message.
- 1235 <ds:Signature> [Optional]  
 1236 An XML Signature that authenticates the response, as described in Section SAML and XML  
 1237 Signature Syntax and Processing.
- 1238 <Extensions> [Optional]  
 1239 This contains optional protocol message extension elements that are agreed upon between the  
 1240 communicating parties.
- 1241 <Status> [Required]  
 1242 A code representing the status of the corresponding request.
- 1243 The following schema fragment defines the **StatusResponseType** complex type:

```

1244 <complexType name="StatusResponseType">
1245   <sequence>
1246     <element ref="saml:Issuer" minOccurs="0"/>
1247     <element ref="ds:Signature" minOccurs="0"/>
1248     <element ref="samlp:Extensions" minOccurs="0"/>
1249     <element ref="samlp:Status"/>
1250   </sequence>
1251   <attribute name="ResponseID" type="ID" use="required"/>
1252   <attribute name="InResponseTo" type="NCName" use="optional"/>
1253   <attribute name="MajorVersion" type="integer" use="required"/>
1254   <attribute name="MinorVersion" type="integer" use="required"/>
1255   <attribute name="IssueInstant" type="dateTime" use="required"/>
1256   <attribute name="Recipient" type="anyURI" use="optional"/>
1257 </complexType>

```

### 1258 3.2.2.1 Element <Status>

1259 The <Status> element contains the following elements:

- 1260 <StatusCode> [Required]  
 1261 A code representing the status of the corresponding request.
- 1262 <StatusMessage> [Optional]  
 1263 A message which MAY be returned to an operator.
- 1264 <StatusDetail> [Optional]  
 1265 Additional information concerning an error condition.

1266 The following schema fragment defines the <Status> element and its **StatusType** complex type:

```

1267 <element name="Status" type="samlp:StatusType"/>
1268 <complexType name="StatusType">
1269   <sequence>
1270     <element ref="samlp:StatusCode"/>
1271     <element ref="samlp:StatusMessage" minOccurs="0"/>
1272     <element ref="samlp:StatusDetail" minOccurs="0"/>
1273   </sequence>
1274 </complexType>

```

### 1275 3.2.2.2 Element <StatusCode>

1276 The <StatusCode> element specifies one or more possibly nested, codes representing the status of the  
 1277 corresponding request. The <StatusCode> element has the following element and attribute:

1278 Value [Required]  
1279 The status code value. This attribute contains an XML Schema QName; a namespace prefix MUST  
1280 be provided. The value of the topmost <StatusCode> element MUST be from the top-level list  
1281 provided in this section.

1282 <StatusCode> [Optional]  
1283 A subordinate status code that provides more specific information on an error condition.

1284 The top-level <StatusCode> values are QNames associated with the SAML protocol namespace. The  
1285 local parts of these QNames are as follows:

1286 Success  
1287 The request succeeded.

1288 Requester  
1289 The request could not be performed due to an error on the part of the requester.

1290 Responder  
1291 The request could not be performed due to an error on the part of the SAML responder or SAML  
1292 authority.

1293 VersionMismatch  
1294 The SAML responder could not process the request because the version of the request message  
1295 was incorrect.

1296 The following second-level status codes are referenced at various places in the specification. Additional  
1297 second-level status codes MAY be defined in future versions of the SAML specification.

1298 InvalidNameIDPolicy  
1299 The responding provider does not support the specified name identifier format for the requested  
1300 subject.

1301 NoAuthnContext  
1302 The specified authentication context requirements cannot be met by the responder.

1303 NoAvailableIDP  
1304 Used by an intermediary to indicate that none of the supported identity provider <Loc> elements in  
1305 an <IDPList> can be resolved or that none of the supported identity providers are available.

1306 NoPassive  
1307 Indicates the identity provider cannot authenticate the principal passively, as has been requested.

1308 NoSupportedIDP  
1309 Used by an intermediary to indicate that none of the identity providers in an <IDPList> are  
1310 supported by the intermediary.

1311 ProxyCountExceeded  
1312 Indicates that an identity provider cannot authenticate the principal directly and is not permitted to  
1313 proxy the request further.

1314 RequestDenied  
1315 The SAML responder or SAML authority is able to process the request but has chosen not to  
1316 respond. This status code MAY be used when there is concern about the security context of the  
1317 request message or the sequence of request messages received from a particular requester.

1318 RequestUnsupported

1319     The SAML responder or SAML authority does not support the request.

1320 RequestVersionDeprecated

1321     The SAML responder can not process any requests with the protocol version specified in the  
1322     request.

1323 RequestVersionTooHigh

1324     The SAML responder cannot process the request because the protocol version specified in the  
1325     request message is a major upgrade from the highest protocol version supported by the responder.

1326 RequestVersionTooLow

1327     The SAML responder cannot process the request because the protocol version specified in the  
1328     request message is too low.

1329 ResourceNotRecognized

1330     The SAML authority does not wish to support resource-specific attribute queries, or the resource  
1331     value provided in the request message is invalid or unrecognized.

1332 TooManyResponses

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

1334 UnknownPrincipal

1335     The responding provider does not recognize the principal specified or implied by the request.

1336 SAML system entities are free to define more specific status codes in other namespaces, but MUST NOT  
1337 define additional codes in the SAML assertion or protocol namespace.

1338 The QNames defined as status codes SHOULD be used only in the <StatusCode> element's Value  
1339 attribute and have the above semantics only in that context.

1340 The following schema fragment defines the <StatusCode> element and its **StatusCodeType** complex  
1341 type:

```
1342 <element name="StatusCode" type="samlp:StatusCodeType"/>
1343 <complexType name="StatusCodeType">
1344   <sequence>
1345     <element ref="samlp:StatusCode" minOccurs="0"/>
1346   </sequence>
1347   <attribute name="Value" type="QName" use="required"/>
1348 </complexType>
```

### 1349 **3.2.2.3 Element <StatusMessage>**

1350 The <StatusMessage> element specifies a message that MAY be returned to an operator:

1351 The following schema fragment defines the <StatusMessage> element:

```
1352 <element name="StatusMessage" type="string"/>
```

### 1353 **3.2.2.4 Element <StatusDetail>**

1354 The <StatusDetail> element MAY be used to specify additional information concerning an error  
1355 condition.

1356 The following schema fragment defines the <StatusDetail> element and its **StatusDetailType**  
1357 complex type:

```

1358 <element name="StatusDetail" type="samlp:StatusDetailType"/>
1359 <complexType name="StatusDetailType">
1360   <sequence>
1361     <any namespace="##any" processContents="lax" minOccurs="0"
1362     maxOccurs="unbounded"/>
1363   </sequence>
1364 </complexType>

```

### 1365 3.3 Assertion Query and Request Protocol

1366 This section defines messages and processing rules for requesting existing assertions by reference or  
 1367 querying for assertions by subject and statement type.

#### 1368 3.3.1 Element <AssertionIDRequest>

1369 If the requester knows the unique identifier of one or more assertions, the <AssertionIDRequest>  
 1370 message can be used to request that the assertion(s) be returned in a <Response> message. The  
 1371 <saml:AssertionIDReference> element is used to specify the assertion(s) to return. See Section  
 1372 Element <AssertionIDReference> for more information on this element.

1373 The following schema fragment defines the <AssertionIDRequest> element:

```

1374 <element name="AssertionIDRequest" type="samlp:AssertionIDRequestType"/>
1375 <complexType name="AssertionIDRequestType">
1376   <complexContent>
1377     <extension base="samlp:RequestAbstractType">
1378       <sequence>
1379         <element ref="saml:AssertionIDReference"
1380         maxOccurs="unbounded"/>
1381       </sequence>
1382     </extension>
1383   </complexContent>
1384 </complexType>

```

#### 1385 3.3.2 Queries

1386 The following sections define the SAML query request messages.

##### 1387 3.3.2.1 Element <SubjectQuery>

1388 The <SubjectQuery> message element is an extension point that allows new SAML queries to be  
 1389 defined that specify a single SAML subject. Its **SubjectQueryAbstractType** complex type is abstract and  
 1390 is thus usable only as the base of a derived type. **SubjectQueryAbstractType** adds the <Subject>  
 1391 element and an optional *SessionIndex* attribute to **RequestAbstractType**.

1392 *SessionIndex* [Optional]

1393 If present, specifies a filter for possible responses. Such a query asks the question "What assertions  
 1394 containing subject statements do you have for this subject within the context of the supplied session  
 1395 information?"

1396 If the *SessionIndex* attribute is present in any defined query, at least one element that extends  
 1397 **StatementAbstractType** in the set of returned assertions MUST contain an *SessionIndex* attribute  
 1398 that matches the *SessionIndex* attribute in the query. It is OPTIONAL for the complete set of all such  
 1399 matching assertions to be returned in the response.

1400 The following schema fragment defines the <SubjectQuery> element and its  
 1401 **SubjectQueryAbstractType** complex type:

```

1402 <element name="SubjectQuery" type="saml:SubjectQueryAbstractType"/>
1403 <complexType name="SubjectQueryAbstractType" abstract="true">
1404   <complexContent>
1405     <extension base="saml:RequestAbstractType">
1406       <sequence>
1407         <element ref="saml:Subject"/>
1408       </sequence>
1409       <attribute name="SessionIndex" type="string"
1410 use="optional"/>
1411     </extension>
1412   </complexContent>
1413 </complexType>

```

### 1414 3.3.2.2 Element <AuthenticationQuery>

1415 The <AuthenticationQuery> message element is used to make the query “What assertions  
1416 containing authentication statements are available for this subject?” A successful <Response> will  
1417 contain one or more assertions containing authentication statements.

1418 The <AuthenticationQuery> message MUST NOT be used as a request for a new authentication  
1419 using credentials provided in the request. <AuthenticationQuery> is a request for statements about  
1420 authentication acts that have occurred in a previous interaction between the indicated subject and the  
1421 Authentication Authority.

1422 This element is of type **AuthenticationQueryType**, which extends **SubjectQueryAbstractType** with the  
1423 addition of the following attribute:

1424 AuthenticationMethod [Optional]

1425 If present, specifies a filter for possible responses. Such a query asks the question “What assertions  
1426 containing authentication statements do you have for this subject with the supplied authentication  
1427 method?”

1428 In response to an authentication query, a SAML authority returns assertions with authentication  
1429 statements as follows:

- 1430 • Rules given in Section for matching against the <Subject> element of the query identify the  
1431 assertions that may be returned.
- 1432 • If the AuthenticationMethod attribute is present in the query, at least one  
1433 <AuthenticationStatement> element in the set of returned assertions MUST contain an  
1434 AuthenticationMethod attribute that matches the AuthenticationMethod attribute in  
1435 the query. It is OPTIONAL for the complete set of all such matching assertions to be returned in  
1436 the response.

1437 TODO: add <AuthnContext> into message in some fashion. Maybe reuse <RequestAuthnContext> from  
1438 <AuthnRequest>?

1439 The following schema fragment defines the <AuthenticationQuery> element and its  
1440 **AuthenticationQueryType** complex type:

```

1441 <element name="AuthenticationQuery" type="saml:AuthenticationQueryType"/>
1442 <complexType name="AuthenticationQueryType">
1443   <complexContent>
1444     <extension base="saml:SubjectQueryAbstractType">
1445       <attribute name="AuthenticationMethod" type="anyURI"/>
1446     </extension>
1447   </complexContent>
1448 </complexType>

```

### 1449 3.3.2.3 Element <AttributeQuery>

1450 The <AttributeQuery> element is used to make the query "Return the requested attributes for this  
1451 subject." A successful response will be in the form of assertions containing attribute statements. This  
1452 element is of type **AttributeQueryType**, which extends **SubjectQueryAbstractType** with the addition of  
1453 the following element and attribute:

1454 Resource [Optional]

1455 If present, specifies that the attribute query is being made in order to evaluate a specific access  
1456 request relating to the resource. The SAML authority MAY use the resource attribute to establish the  
1457 scope of the request. It is permitted for this attribute to have the value of the empty URI reference  
1458 (""), and the meaning is defined to be "the start of the current document", as specified by [RFC 2396]  
1459 §4.2.

1460 If the resource attribute is specified and the SAML authority does not wish to support resource-  
1461 specific attribute queries, or if the resource value provided is invalid or unrecognized, then the  
1462 Attribute Authority SHOULD respond with a top-level <StatusCode> value of Responder and a  
1463 second-level <StatusCode> value of ResourceNotRecognized.

1464 <AttributeDesignator> [Any Number]

1465 Each <AttributeDesignator> element specifies an attribute whose value is to be returned. If no  
1466 attributes are specified, it indicates that all attributes allowed by policy are requested.

1467 In response to an attribute query, a SAML authority returns assertions with attribute statements as  
1468 follows:

- 1469 • Rules given in Section for matching against the <Subject> element of the query identify the  
1470 assertions that may be returned.
- 1471 • If any <AttributeDesignator> elements are present in the query, they constrain the attribute  
1472 values returned, as noted above.
- 1473 • The SAML authority MAY take the Resource attribute into account in further constraining the values  
1474 returned, as noted above.
- 1475 • The attribute values returned MAY be constrained by application-specific policy considerations.

1476 The following schema fragment defines the <AttributeQuery> element and its **AttributeQueryType**  
1477 complex type:

```
1478 <element name="AttributeQuery" type="saml:AttributeQueryType"/>  
1479 <complexType name="AttributeQueryType">  
1480   <complexContent>  
1481     <extension base="saml:SubjectQueryAbstractType">  
1482       <sequence>  
1483         <element ref="saml:AttributeDesignator"  
1484           minOccurs="0" maxOccurs="unbounded"/>  
1485       </sequence>  
1486       <attribute name="Resource" type="anyURI" use="optional"/>  
1487     </extension>  
1488   </complexContent>  
1489 </complexType>
```

### 1490 3.3.2.4 Element <AuthorizationDecisionQuery>

1491 The <AuthorizationDecisionQuery> element is used to make the query "Should these actions on  
1492 this resource be allowed for this subject, given this evidence?" A successful response will be in the form  
1493 of assertions containing authorization decision statements.

1494 **Note:** The <AuthorizationDecisionQuery> feature has been frozen as of SAML  
1495 V2.0, with no future enhancements planned. Users who require additional functionality  
1496 may want to consider the eXtensible Access Control Markup Language [XACML], which  
1497 offers enhanced authorization decision features.

1498 This element is of type **AuthorizationDecisionQueryType**, which extends **SubjectQueryAbstractType**  
1499 with the addition of the following elements and attribute:

1500 Resource [Required]

1501 A URI reference indicating the resource for which authorization is requested.

1502 <Action> [One or More]

1503 The actions for which authorization is requested.

1504 <Evidence> [Optional]

1505 A set of assertions that the SAML authority MAY rely on in making its authorization decision.

1506 In response to an authorization decision query, a SAML authority returns assertions with authorization  
1507 decision statements as follows:

- 1508 • Rules given in Section 3.3.4.1 for matching against the <Subject> element of the query identify the  
1509 assertions that may be returned.

1510 The following schema fragment defines the <AuthorizationDecisionQuery> element and its  
1511 **AuthorizationDecisionQueryType** complex type:

```
1512 <element name="AuthorizationDecisionQuery"  
1513 type="saml:AuthorizationDecisionQueryType"/>  
1514 <complexType name="AuthorizationDecisionQueryType">  
1515 <complexContent>  
1516 <extension base="saml:SubjectQueryAbstractType">  
1517 <sequence>  
1518 <element ref="saml:Action" maxOccurs="unbounded"/>  
1519 <element ref="saml:Evidence" minOccurs="0"/>  
1520 </sequence>  
1521 <attribute name="Resource" type="anyURI" use="required"/>  
1522 </extension>  
1523 </complexContent>  
1524 </complexType>
```

### 1525 3.3.3 Element <Response>

1526 The <Response> message element is used when a response consists of a list of zero or more  
1527 assertions that answer the request. It has the complex type **ResponseType**, which extends  
1528 **StatusResponseType** by adding the following element:

1529 <Assertion> [Any Number]

1530 Specifies an assertion by value. (See Section Element <Assertion> for more information.)

1531 The following schema fragment defines the <Response> element and its **ResponseType** complex type:

```
1532 <element name="Response" type="saml:ResponseType"/>  
1533 <complexType name="ResponseType">  
1534 <complexContent>  
1535 <extension base="saml:StatusResponseType">  
1536 <sequence>  
1537 <element ref="saml:Assertion" minOccurs="0"  
1538 maxOccurs="unbounded"/>  
1539 </sequence>
```

1540  
1541  
1542

```
</extension>  
</complexContent>  
</complexType>
```

### 1543 3.3.3.1 Processing Rules

1544 In response to a query message, every assertion returned by a SAML authority MUST contain a  
1545 `<Subject>` element that **strongly matches** the `<Subject>` element found in the query.

1546 A `<Subject>` element S1 strongly matches S2 if and only if the following two conditions both apply:

- 1547 • If S2 includes an identifier element (any element whose type is derived from  
1548 **BaseIdentifierAbstractType**), then S1 must include an identical identifier element.
- 1549 • If S2 includes one or more `<SubjectConfirmation>` elements, then S1 must include at least one  
1550 `<SubjectConfirmation>` element such that the assertion's subject can be confirmed in the  
1551 manner described by at least one element in the requested set.

1552 If the SAML authority cannot provide an assertion with any statements satisfying the constraints  
1553 expressed by a query, the `<Response>` element MUST NOT contain an `<Assertion>` element and  
1554 MUST include a `<StatusCode>` element with value `Success`. It MAY return a `<StatusMessage>`  
1555 element with additional information.

## 1556 3.4 Authentication Request Protocol

1557 When a principal (or an agent acting on the principal's behalf) wishes to obtain assertions containing  
1558 authentication statements to establish a security context at one or more relying parties, it can use the  
1559 authentication request protocol to send an `<AuthnRequest>` message to a SAML authority and request  
1560 that it return a `<Response>` message containing one or more such assertions. Such assertions MAY  
1561 contain additional statements of any type, but at least one assertion MUST contain at least one  
1562 authentication statement. A SAML authority that supports this protocol is also termed an identity  
1563 provider.

1564 Apart from this requirement, the specific contents of the returned assertions depend on the profile or  
1565 context of use. Also, the exact means by which the principal or agent authenticates to the identity  
1566 provider are not specified, though the means of authentication MAY impact the content of the response.  
1567 Other issues related to the validation of authentication credentials by the identity provider or any  
1568 communication between the identity provider and any other entities involved in the authentication  
1569 process are also out of scope of this protocol.

1570 The descriptions and processing rules in the following sections reference the following actors, many of  
1571 whom might be the same entity in a particular profile of use:

1572 Request Issuer

1573 The entity who creates the authentication request and to whom the response is to be returned.

1574 Presenter

1575 The entity who presents the request to the authority and either authenticates itself during the  
1576 sending of the message, or relies on an existing security context to establish its identity. If not  
1577 the request issuer, the sender acts as an intermediary between the request issuer and the  
1578 responding identity provider.

1579 Requested Subject

1580 The entity about whom one or more assertions are being requested.

- 1581 Confirming Subject  
1582           The entity or entities expected to be able to satisfy one of the <SubjectConfirmation>  
1583           elements of the resulting assertion(s).
- 1584 Relying Party  
1585           The entity or entities expected to consume the assertion(s) to accomplish a purpose defined by  
1586           the profile or context of use, generally to establish a security context.

### 1587 **3.4.1 Element <AuthnRequest>**

1588 To request that an identity provider issue an authentication assertion, an entity authenticates to it (or  
1589 relies on an existing security context) and sends it an <AuthnRequest> message that describes the  
1590 properties that the resulting assertion needs to have to satisfy its purpose. Among these properties may  
1591 be information that relates to the content of the assertion and/or information that relates to how the  
1592 resulting <Response> message should be delivered to the request issuer.

1593 The request issuer might not be the same as the presenter of the request, if for example the request  
1594 issuer is a relying party that intends to use the resulting assertion to authenticate or authorize the  
1595 requested subject to provide a service.

1596 The <AuthnRequest> message SHOULD be signed or otherwise authenticated and integrity protected  
1597 by the protocol binding used to deliver the message.

1598 This message has the complex type **AuthnRequestType**, which extends **RequestAbstractType** and  
1599 adds the following elements and attributes, all of which are optional in general, but may be required by  
1600 specific profiles:

1601 <Subject> [Optional]

1602           Specifies the requested subject of the resulting assertion(s). This may include one or more  
1603           <SubjectConfirmation> elements to indicate how and/or by whom the resulting assertions' can  
1604           be confirmed.

1605           If entirely omitted or if no identifier is included, the presenter of the message is presumed to be the  
1606           requested subject. If no <SubjectConfirmation> elements are included, then the presenter is  
1607           presumed to be the only confirming entity required and the method is implied by the profile of use  
1608           and/or the policies of the identity provider.

1609 <NameIDPolicy> [Optional]

1610           Specifies constraints on the name identifier to be used to represent the requested subject. If omitted,  
1611           then any type of identifier supported by the identity provider for the requested subject can be used,  
1612           constrained by any relevant deployment-specific policies, with respect to privacy, for example.

1613 <Conditions> [Optional]

1614           Specifies the SAML conditions the request issuer expects to govern the validity and/or use of the  
1615           resulting assertion(s). The responder MAY modify or supplement this set as it deems necessary.

1616 <RequestAuthnContext> [Optional]

1617           Specifies the requirements, if any, that the request issuer places on the authentication context that  
1618           applies to the responding provider's authentication of the presenter.

1619 <Scoping> [Optional]

1620           Specifies the identity providers trusted by the request issuer to authenticate the presenter, as well as  
1621           limitations and context related to proxying of the <AuthnRequest> message to subsequent identity  
1622           providers by the responder.

- 1623 `IsPassive` [Optional]
- 1624 A Boolean value. If "true", the identity provider and the user agent itself MUST NOT take control of  
 1625 the user interface from the request issuer and interact with the presenter in a noticeable fashion. If a  
 1626 value is not provided, the default is "true".
- 1627 `ForceAuthn` [Optional]
- 1628 A Boolean value. If "true", the identity provider MUST authenticate the presenter directly rather than  
 1629 rely on a previous security context. If a value is not provided, the default is "false". However, if both  
 1630 `ForceAuthn` and `IsPassive` are "true", the identity provider MUST NOT freshly authenticate the  
 1631 presenter unless the constraints of `IsPassive` can be met.
- 1632 `ProtocolBinding` [Optional]
- 1633 A URI that identifies a SAML protocol binding to be used when returning the `<Response>` message.
- 1634 `AssertionConsumerServiceID` [Optional]
- 1635 References one of a set of `<AssertionConsumerService>` elements in the request issuer's  
 1636 metadata as the one to which the `<Response>` should be returned. It applies only to profiles that  
 1637 specify use of this metadata element, in which the request issuer is different than the presenter. If  
 1638 omitted, the metadata element labeled with the `isDefault` attribute MUST be used with such  
 1639 profiles.
- 1640 `AssertionConsumerServiceURL` [Optional]
- 1641 Specifies by value the location to which the `<Response>` message MUST be returned. The  
 1642 responder MUST insure by some means (such as metadata) that the value specified is in fact  
 1643 associated with the request issuer.
- 1644 `ProviderName` [Optional]
- 1645 Specifies the human-readable name of the request issuer for use by the presenter's user agent or  
 1646 the identity provider.
- 1647 See Section 3.4.1.8 for general processing rules regarding this message.
- 1648 The following schema fragment defines the `<AuthnRequest>` element and its **AuthnRequestType**  
 1649 complex type:

```

1650 <element name="AuthnRequest" type="saml:AuthnRequestType"/>
1651 <complexType name="AuthnRequestType">
1652   <complexContent>
1653     <extension base="saml:RequestAbstractType">
1654       <sequence>
1655         <element ref="saml:Subject" minOccurs="0"/>
1656         <element ref="saml:NameIDPolicy" minOccurs="0"/>
1657         <element
1658           ref="saml:Conditions" minOccurs="0"/>
1659         <element ref="saml:RequestAuthnContext"
1660           minOccurs="0"/>
1661         <element ref="saml:Scoping" minOccurs="0"/>
1662       </sequence>
1663       <attribute name="IsPassive" type="boolean"
1664         use="optional"/>
1665       <attribute name="ForceAuthn" type="boolean"
1666         use="optional"/>
1667       <attribute name="ProtocolBinding" type="anyURI"
1668         use="optional"/>
1669       <attribute name="AssertionConsumerServiceID" type="string"
1670         use="optional"/>
1671       <attribute name="AssertionConsumerServiceURL"
1672         type="anyURI" use="optional"/>

```

```

1673         <attribute name="ProviderName" type="string"
1674 use="optional"/>
1675         </extension>
1676     </complexContent>
1677 </complexType>

```

### 1678 3.4.1.1 Element <NameIDPolicy>

1679 The <NameIDPolicy> element tailors the name identifier in the subjects of assertions resulting from an  
 1680 <AuthnRequest>. Its **NameIDPolicyType** complex type defines the following attributes:

1681 Format [Required]

1682 Specifies the URI of a name identifier format defined in this or another specification (see Section 7.3  
 1683 for examples).

1684 SPNameQualifier [Optional]

1685 Used with a Format of urn:oasis:names:tc:SAML:2.0:nameid-format:federated or  
 1686 urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted, it optionally specifies that a  
 1687 federated identifier be returned (or created) in the namespace of a service provider other than the  
 1688 issuing service provider, or an affiliation group.

1689 AllowCreate [Optional]

1690 Used to indicate whether the identity provider is allowed, in the course of fulfilling the request, to  
 1691 create a new identifier to represent the principal. Defaults to "true". When "false", the request issuer  
 1692 constrains the identity provider to only issue an assertion to it if an acceptable identifier for the  
 1693 principal has already been established between them.

1694 When this element is used, if the content is not understood by or acceptable to the identity provider,  
 1695 then a <Response> MUST be returned with a <Status> containing a second-level <StatusCode> of  
 1696 samlp:InvalidNameIDPolicy.

1697 A Format of urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted indicates that the  
 1698 resulting assertion(s) MUST contain <EncryptedIdentifier> elements instead of plaintext. The  
 1699 underlying name identifier's unencrypted form can be of any type supported by the identity provider for  
 1700 the requested subject.

1701 Any Format value (or the omission of this element) MAY result in an <EncryptedIdentifier> in the  
 1702 resulting assertion(s), if the identity provider's (or the subject's) policies regarding privacy dictate this.

1703 The following schema fragment defines the <NameIDPolicy> element and its **NameIDPolicyType**  
 1704 complex type:

```

1705     <element name="NameIDPolicy" type="samlp:NameIDPolicyType"/>
1706     <complexType name="NameIDPolicyType">
1707         <sequence/>
1708         <attribute name="Format" type="anyURI" use="required"/>
1709         <attribute name="SPNameQualifier" type="string" use="optional"/>
1710         <attribute name="AllowCreate" type="boolean" use="optional"/>
1711     </complexType>

```

### 1712 3.4.1.2 Element <RequestAuthnContext>

1713 The <RequestAuthnContext> element specifies the authentication context requirements of the  
 1714 request issuer with respect to the authentication of the presenter. Its **RequestAuthnContextType**  
 1715 complex type defines the following elements and attributes:

1716 <AuthnContextClassRef> or <AuthnContextStatementRef> [One or More]  
 1717 Specifies one or more URIs identifying authentication context classes or statements.

1718 Comparison [Optional]  
 1719 Specifies the comparison method used to evaluate the requested context classes or statements, one  
 1720 of "exact", "minimum", "maximum", or "better". The default is "exact".

1721 If <RequestAuthnContext> is specified in an <AuthnRequest> message, the authentication  
 1722 statement in the resulting assertion MUST contain an authentication context that conforms to the  
 1723 requested context as described below.

1724 Either a set of class references or statement references can be used. Additionally, the set of supplied  
 1725 references MUST be evaluated as an ordered set, where the first element is the most preferred  
 1726 authentication context class or statement. If none of the specified classes or statements can be satisfied  
 1727 in accordance with the rules below, then the identity provider MUST return a <Response> message with  
 1728 a second-level <StatusCode> of samlp:NoAuthnContext.

1729 If Comparison is set to "exact" or omitted, then the resulting authentication context in the authentication  
 1730 statement MUST be the exact match of at least one of the authentication contexts specified.

1731 If Comparison is set to "minimum", then the resulting authentication context in the authentication  
 1732 statement MUST be at least as strong (as deemed by the identity provider) as one of the authentication  
 1733 contexts specified.

1734 If Comparison is set to "better", then the resulting authentication context in the authentication statement  
 1735 MUST be stronger (as deemed by the identity provider) than any one of the authentication contexts  
 1736 specified.

1737 If Comparison is set to "maximum", then the resulting authentication context in the authentication  
 1738 statement MUST be as strong as possible (as deemed by the identity provider) without exceeding the  
 1739 strength of at least one of the authentication contexts specified.

1740 The following schema fragment defines the <RequestAuthnContext> element and its  
 1741 **RequestAuthnContextType** complex type:

```

1742 <element name="RequestAuthnContext" type="samlp:RequestAuthnContextType"/>
1743 <complexType name="RequestAuthnContextType">
1744   <choice>
1745     <element ref="saml:AuthnContextClassRef" maxOccurs="unbounded"/>
1746     <element ref="saml:AuthnContextStatementRef"
1747 maxOccurs="unbounded"/>
1748   </choice>
1749
1750   <attribute name="Comparison" type="samlp:AuthnContextComparisonType"
1751 use="optional"/>
1752 </complexType>
1753 <simpleType name="AuthnContextComparisonType">
1754   <restriction base="string">
1755     <enumeration value="exact"/>
1756     <enumeration value="minimum"/>
1757     <enumeration value="maximum"/>
1758     <enumeration value="better"/>
1759   </restriction>
1760 </simpleType>

```

### 1761 3.4.1.3 Element <Scoping>

1762 The <Scoping> element specifies the identity providers trusted by the request issuer to authenticate the  
1763 presenter, as well as limitations and context related to proxying of the <AuthnRequest> message to  
1764 subsequent identity providers by the responder. Its **ScopingType** complex type defines the following  
1765 elements and attribute:

1766 <IDPList> [Optional]

1767 An advisory list of identity providers and associated information that the request issuer deems  
1768 acceptable to respond to the request.

1769 <RequesterID> [Zero or More]

1770 Identifies the set requesting entities on whose behalf the request issuer is acting. Used to  
1771 communicate the chain of request issuers when proxying occurs, as described in section 3.4.1.9.

1772 ProxyCount [Optional]

1773 Specifies the number of proxying indirections permissible between the identity provider that receives  
1774 this <AuthnRequest> and the identity provider who ultimately authenticates the principal. A count  
1775 of zero permits no proxying, while omitting this attribute expresses no such restriction.

1776 In profiles specifying an active intermediary, the intermediary MAY examine the list and return a  
1777 <Response> message with a second-level <StatusCode> of samlp:NoAvailableIDP or  
1778 samlp:NoSupportedIDP if it cannot contact or does not support any of the specified identity providers.

1779 The following schema fragment defines the <Scoping> element and its **ScopingType** complex type:

```
1780 <element name="Scoping" type="samlp:ScopingType"/>
1781 <complexType name="ScopingType">
1782   <sequence>
1783     <element ref="samlp:IDPList" minOccurs="0"/>
1784     <element ref="samlp:RequesterID" minOccurs="0"
1785     maxOccurs="unbounded"/>
1786   </sequence>
1787   <attribute name="ProxyCount" type="nonNegativeInteger" use="optional"/>
1788 </complexType>
1789 <element name="RequesterID" type="anyURI"/>
```

### 1790 3.4.1.4 Element <IDPList>

1791 The <IDPList> element specifies the identity providers trusted by the request issuer to authenticate the  
1792 presenter. Its **IDPListType** complex type defines the following elements:

1793 <IDPEntry> [One or More]

1794 Information about a single identity provider

1795 <GetComplete> [Optional]

1796 If the <IDPList> is not complete, this element may specify a URI that resolves to the complete list.

1797 The following schema fragment defines the <IDPList> element and its **IDPListType** complex type:

```
1798 <element name="IDPList" type="samlp:IDPListType"/>
1799 <complexType name="IDPListType">
1800   <sequence>
1801     <element ref="samlp:IDPEntry" maxOccurs="unbounded"/>
1802     <element ref="samlp:GetComplete" minOccurs="0"/>
1803   </sequence>
1804 </complexType>
1805 <element name="GetComplete" type="anyURI"/>
```

### 1806 3.4.1.5 Element <IDPEntry>

1807 The <IDPEntry> element specifies a single identity provider trusted by the request issuer to  
1808 authenticate the presenter. Its **IDPEntryType** complex type defines the following elements:

1809 <ID> [Required]

1810 The unique identifier of the identity provider

1811 <Name> [Optional]

1812 A human readable name for the identity provider

1813 <Loc> [Optional]

1814 The location of a profile-specific endpoint supporting the authentication request protocol. The  
1815 binding to be used must be understood from the profile of use.

1816 The following schema fragment defines the <IDPEntry> element and its **IDPEntryType** complex type:

```
1817 <element name="IDPEntry" type="samlp:IDPEntryType"/>  
1818 <complexType name="IDPEntryType">  
1819   <sequence/>  
1820   <attribute name="ID" type="anyURI" use="required"/>  
1821   <attribute name="Name" type="string" use="optional"/>  
1822   <attribute name="Loc" type="anyURI" use="optional"/>  
1823 </complexType>
```

### 1824 3.4.1.6 Processing Rules

1825 The <AuthnRequest> and <Response> exchange supports a variety of usage scenarios and is  
1826 therefore typically profiled for use in a specific context in which this optionality is constrained and specific  
1827 kinds of input and output are required or prohibited. The following processing rules apply as invariant  
1828 behavior across any profile of this protocol exchange.

1829 The recipient **MUST** validate any signature present on the request or response message.

1830 The responder **MUST** ultimately reply to an <AuthnRequest> with a <Response> message containing  
1831 one or more assertions that meet the specifications defined by the request, or a <Status> describing  
1832 the error that occurred. The responder **MAY** conduct additional message exchanges with the request  
1833 sender as needed to initiate or complete the authentication process, subject to the nature of the protocol  
1834 binding and the authentication mechanism. As described in the next section, this includes proxying the  
1835 request by directing the presenter to another identity provider by issuing its own <AuthnRequest>  
1836 message, so that the resulting assertion can be used to authenticate the presenter to the original  
1837 responder.

1838 If the responder is unable to authenticate the presenter or does not recognize the requested subject, it  
1839 **MUST** return a <Response> with a <Status> containing a second-level <StatusCode> of  
1840 `samlp:UnknownPrincipal`.

1841 If the <Subject> element in the request is present, then the resulting assertions' <Subject> **MUST**  
1842 **strongly match** the request <Subject>, as described in section 3.3.4.1, except that the identifier **MAY**  
1843 be in a different form if specified by <NameIDPolicy>.

1844 All of the content defined specifically within <AuthnRequest> is optional, although some may be  
1845 required by certain profiles. In the absence of any specific content at all, the following behavior is  
1846 assumed:

- 1847 • The assertion(s) returned **MUST** contain a <Subject> element that represents the presenter.  
1848 The identifier type and format are determined by the identity provider. At least one statement

- 1849 MUST be an `<AuthenticationStatement>` that describes the authentication performed by the  
1850 responder or authentication service associated with it.
- 1851 • The request presenter should, to the extent possible, be the only entity able to satisfy the  
1852 `<SubjectConfirmation>` of the assertion(s). In the case of weaker confirmation methods,  
1853 binding-specific or other mechanisms will be used to help satisfy this requirement.
  - 1854 • The resulting assertion(s) MUST contain an `<AudienceRestrictionCondition>` element  
1855 referencing the request issuer as an acceptable relying party. Other audiences MAY be included  
1856 as deemed appropriate by the identity provider.

### 1857 **3.4.1.7 Proxying**

1858 If an identity provider that receives an `<AuthnRequest>` has not yet authenticated the presenter or  
1859 cannot directly authenticate him/her, but believes that the presenter has already authenticated to another  
1860 identity provider, it may respond to the request by issuing a new `<AuthnRequest>` on its own behalf to  
1861 be presented to the other identity provider. The original identity provider is termed the proxying identity  
1862 provider.

1863 Upon the successful return of a `<Response>` to the proxying provider, the enclosed assertion MAY be  
1864 used to authenticate the presenter so that the proxying provider can issue an assertion of its own in  
1865 response to the original `<AuthnRequest>`, completing the overall message exchange. Both the  
1866 proxying and authenticating identity providers MAY include constraints on proxying activity in the  
1867 messages and assertions they issue, as described in previous sections, and below.

1868 The request issuer can influence proxy behavior by including a `<Scoping>` element where the provider  
1869 sets a desired `ProxyCount` value and/or indicates a list of preferred identity providers which may be  
1870 proxied by including an ordered `<IDPList>` of preferred providers.

1871 An identity provider can control secondary use of its assertions by proxying identity providers using a  
1872 `<ProxyRestrictionCondition>` element in the assertions it issues.

#### 1873 **3.4.1.7.1 Processing Rules**

1874 An identity provider MAY proxy an `<AuthnRequest>` if the `<ProxyCount>` attribute is omitted or is  
1875 greater than zero. Whether it chooses to proxy or not is a matter of local policy. An identity provider MAY  
1876 choose to proxy for a provider specified in the `<IDPList>`, if provided, but is not required to do so.

1877 An identity provider MUST NOT proxy a request where `<ProxyCount>` is set to zero. The identity  
1878 provider MUST return an error containing a second-level `<samlp:StatusCode>` value of  
1879 `samlp:ProxyCountExceeded`, unless it can directly authenticate the presenter.

1880 If it chooses to proxy, when creating the new `<AuthnRequest>`, an identity provider MUST include  
1881 equivalent or stricter forms of all the information included in the original request (such as authentication  
1882 context policy). Note however that the proxying provider is free to specify whatever `<NameIDPolicy>` it  
1883 wishes to maximize the chances of a successful response.

1884 If the authenticating identity provider is not a SAML identity provider, then the proxying provider MUST  
1885 have some other way to ensure that the elements governing user agent interaction (`<IsPassive>`, for  
1886 example) will be honored by the authenticating provider.

1887 The new `<AuthnRequest>` MUST contain a `<ProxyCount>` attribute with a value of at most one less  
1888 than the original value. If the original request does not contain a `<ProxyCount>` attribute, then the new  
1889 request SHOULD contain a `<ProxyCount>` attribute.

1890 If an <IDPList> was specified in the original request, the new request MUST also contain an  
1891 <IDPList>. The proxying identity provider MAY add additional identity providers to the end of the  
1892 <IDPList>, but MUST NOT remove any from the list.

1893 The authentication request and response are processed in normal fashion, in accordance with the rules  
1894 given in Section 3.4.1.8 and the profile of use. Once the presenter has authenticated to the proxying  
1895 identity provider (by delivering a <Response>), the following steps are followed:

- 1896 • The proxying identity provider prepares a new assertion on its own behalf by copying in the  
1897 relevant information from the original assertion. The original assertion will be restricted by  
1898 <AudienceRestrictionCondition> to (at least) the proxying identity provider, while the new  
1899 assertion's condition will reference (at least) the original request issuer.
- 1900 • The new assertion's <Subject> should contain an identifier that satisfies the original request  
1901 issuer's preferences, as defined by its <NameIDPolicy> element.
- 1902 • The <AuthenticationStatement> in the new assertion MUST include an <AuthnContext>  
1903 element containing an <ac:AuthenticatingAuthority> element referencing the identity  
1904 provider to which the proxying identity provider referred the presenter. If the original assertion  
1905 contains <AuthnContext> information that includes one or more  
1906 <ac:AuthenticatingAuthority> elements, those elements SHOULD be included in the new  
1907 assertion, with the new element placed after them.
- 1908 • If the authenticating identity provider is not a SAML provider, then the proxying identity provider  
1909 MUST generate a unique identifier value for the authenticating provider. This value SHOULD be  
1910 consistent over time across different requests. The value MUST not conflict with values used or  
1911 generated by other SAML providers.
- 1912 • Any other <AuthnContext> information MAY be copied, translated, or omitted in accordance  
1913 with the policies of the proxying identity provider, provided that the original requirements dictated  
1914 by the request issuer are met.

1915 If, in the future, the identity provider is asked to authenticate the same presenter for a second request  
1916 issuer, and this request is equally or less strict than the original request, the identity provider MAY skip  
1917 the creation of a new <AuthnRequest> to the authenticating identity provider and immediately issue  
1918 another assertion (assuming the original assertion it received is still valid). The concrete definition of  
1919 "equally or less strict" is up to the proxying identity provider.

## 1920 **3.5 Artifact Protocol**

1921 The artifact protocol provides a mechanism by which SAML protocol messages can be transported in a  
1922 SAML binding by reference instead of by value. Both requests and responses can be obtained by  
1923 reference using this specialized protocol. A message sender, instead of binding a message to a transport  
1924 protocol, sends a small piece of data called an artifact using the binding. An artifact can take a variety of  
1925 forms, but must support a means by which the receiver can determine who sent it. If the receiver wishes,  
1926 it can then use this protocol in conjunction with a different (generally synchronous) SAML binding  
1927 protocol to dereference the artifact into the original protocol message. The most common use for this  
1928 mechanism is with bindings that cannot easily carry a message because of size constraints.

1929 Depending on the characteristics of the underlying message being passed by reference, the artifact  
1930 protocol MAY require protections such as mutual authentication, integrity protection, confidentiality, etc.  
1931 from the protocol binding used to dereference the artifact. In all cases, the artifact MUST exhibit a single-  
1932 use semantic such that once it has been successfully dereferenced, it can no longer be used by any  
1933 party.

1934 Regardless of the protocol message obtained, the result of dereferencing an artifact MUST be treated  
1935 exactly as if the message so obtained had been sent originally in place of the artifact.

### 1936 3.5.1 Element <ArtifactRequest>

1937 The <ArtifactRequest> message is used to request that a protocol message be returned in an  
1938 <ArtifactResponse> message by specifying an artifact that represents the protocol message. The  
1939 original transmission of the artifact is governed by the specific binding or profile of SAML that is being  
1940 used; see the SAML specifications for bindings [SAMLBind] and profiles [SAMLProf] for more information  
1941 on the use of artifacts in bindings and profiles.

1942 The <ArtifactRequest> message SHOULD be signed or otherwise authenticated and integrity  
1943 protected by the protocol binding used to deliver the message.

1944 The <Issuer> of the request MUST contain the unique identifier of the requesting provider, with a  
1945 Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:provider.

1946 This message has the complex type **ArtifactRequestType**, which extends **RequestAbstractType** and  
1947 adds the following element:

1948 <Artifact> [Required]

1949 The artifact value that the requester received and now wishes to translate into the protocol message  
1950 it represents. See [SAMLBind] for specific artifact format information.

1951 The following schema fragment defines the <ArtifactRequest> element and its  
1952 **ArtifactRequestType** complex type:

```
1953 <element name="ArtifactRequest" type="samlp:ArtifactRequestType"/>  
1954 <complexType name="ArtifactRequestType">  
1955   <complexContent>  
1956     <extension base="samlp:RequestAbstractType">  
1957       <sequence>  
1958         <element ref="samlp:Artifact"/>  
1959       </sequence>  
1960     </extension>  
1961   </complexContent>  
1962 </complexType>  
1963 <element name="Artifact" type="string"/>
```

### 1964 3.5.2 Element <ArtifactResponse>

1965 The recipient of an <ArtifactRequest> message MUST respond with an <ArtifactResponse>  
1966 message, which is of complex type **ArtifactResponseType**, which extends **StatusResponseType** with a  
1967 single optional wildcard element corresponding to the protocol message being returned. This wrapped  
1968 message element can be a request or a response.

1969 The <ArtifactResponse> message SHOULD be signed or otherwise authenticated and integrity  
1970 protected by the protocol binding used to deliver the message.

1971 The <Issuer> of the response MUST contain the unique identifier of the responding provider, with a  
1972 Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:provider.

1973 The following schema fragment defines the <ArtifactResponse> element and its  
1974 **ArtifactResponseType** complex type:

```
1975 <element name="ArtifactResponse" type="samlp:ArtifactResponseType"/>  
1976 <complexType name="ArtifactResponseType">  
1977   <complexContent>  
1978     <extension base="samlp:StatusResponseType">  
1979       <sequence>  
1980         <any namespace="##any" processContents="lax"  
1981         minOccurs="0"/>  
1982       </sequence>
```

1983  
1984  
1985

```
</extension>  
</complexContent>  
</complexType>
```

### 1986 **3.5.3 Processing Rules**

1987 The recipient **MUST** validate any signature present on the request or response message.

1988 If the responder recognizes the artifact as valid, then it responds with the associated protocol message  
1989 in an `<ArtifactResponse>` message. Otherwise, it responds with an `<ArtifactResponse>`  
1990 message with no embedded message. In both cases, the `<Status>` element **MUST** include a  
1991 `<StatusCode>` element with the code value `Success`. A response message with no embedded  
1992 message inside it is termed an empty response in the remainder of this section.

1993 The responder **MUST** enforce a one-time-use property on the artifact by insuring that any subsequent  
1994 request with the same artifact by any requester results in an empty response as described above.

1995 Some SAML protocol messages, most particularly the `<AuthnRequest>` message in some profiles,  
1996 **MAY** be intended for consumption by any party that receives it and can respond appropriately. In most  
1997 other cases, however, a message is intended for a specific entity. In such cases, the artifact when issued  
1998 **MUST** be associated with the intended recipient of the message that the artifact represents. If the artifact  
1999 issuer receives an `<ArtifactRequest>` from a requester that cannot authenticate itself as the original  
2000 intended recipient, then the artifact issuer **MUST** return an empty response.

2001 The artifact issuer **SHOULD** enforce the shortest practical time limit on the usability of an artifact, such  
2002 that an acceptable window of time (but no more) exists for the artifact receiver to obtain the artifact and  
2003 return it in an `<ArtifactRequest>` to the issuer.

2004 Note that the `<ArtifactResponse>`'s `InResponseTo` attribute **MUST** contain the value of the  
2005 corresponding `<AssertionRequest>`'s `RequestID` attribute, but the embedded protocol message will  
2006 contain its own message identifier, and in the case of an embedded response, may contain a different  
2007 `InResponseTo` value that corresponds to the original request message to which the embedded  
2008 message is responding.

### 2009 **3.6 Name Identifier Management Protocol**

2010 After establishing a persistent name identifier for a principal, an identity provider wishing to change the  
2011 value and/or format of the identifier that it will use when referring to the principal, or to indicate that a  
2012 name identifier will no longer be used to refer to the principal, informs service providers of the change by  
2013 sending them a `<ManageNameIdentifierRequest>` message.

2014 The same message **MAY** also be used by a service provider to register or change the  
2015 `SPProvidedIdentifier` value to be included when the underlying name identifier is used to  
2016 communicate with it, or to terminate the use of a name identifier between itself and the identity provider.

2017

#### 2018 **3.6.1 Element `<ManageNameIdentifierRequest>`**

2019 A provider sends a `<ManageNameIdentifierRequest>` message to inform the recipient of a changed  
2020 name identifier or to indicate the termination of the use of a name identifier.

2021 The `<ManageNameIdentifierRequest>` message **SHOULD** be signed or otherwise authenticated  
2022 and integrity protected by the protocol binding used to deliver the message

2023 The <Issuer> of the request MUST contain the unique identifier of the requesting provider, with a  
2024 Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:provider.

2025 This message has the complex type **RegisterNameIdentifierRequestType**, which extends  
2026 **RequestAbstractType** and adds the following elements:

2027 <NameIdentifier> [Required]

2028 The name identifier and associated attributes that specify the principal as currently recognized by  
2029 the identity and service providers prior to this request.

2030 <NewIdentifier> or <TerminateIdentifier> [Required]

2031 The new identifier value to be used when communicating with the requesting provider concerning  
2032 this principal or an indication that the use of the old identifier has been terminated. In the former  
2033 case, if the requester is the service provider, the new identifier MUST appear in subsequent  
2034 <NameIdentifier> elements in the SPProvidedIdentifier attribute. If the requester is the  
2035 identity provider, the new value will appear in subsequent <NameIdentifier> elements as the  
2036 element's value.

2037 The following schema fragment defines the <RegisterNameIdentifierRequest> element and its  
2038 **RegisterNameIdentifierRequestType** complex type:

```
2039 <element name="ManageNameIdentifierRequest"
2040 type="samlp:ManageNameIdentifierRequestType"/>
2041 <complexType name="ManageNameIdentifierRequestType">
2042 <complexContent>
2043 <extension base="samlp:RequestAbstractType">
2044 <sequence>
2045 <element ref="saml:NameIdentifier"/>
2046 <choice>
2047 <element ref="samlp:NewIdentifier"/>
2048 <element ref="samlp:TerminateIdentifier"/>
2049 </choice>
2050 </sequence>
2051 </extension>
2052 </complexContent>
2053 </complexType>
2054 <element name="NewIdentifier" type="string"/>
2055 <element name="TerminateIdentifier" type="samlp:TerminateIdentifierType"/>
2056 <complexType name="TerminateIdentifierType">
2057 <sequence/>
2058 </complexType>
```

### 2059 3.6.2 Element <ManageNameIdentifierResponse>

2060 The recipient of a <ManageNameIdentifierRequest> message MUST respond with a  
2061 <ManageNameIdentifierResponse> message, which is of type **StatusResponseType** with no  
2062 additional content.

2063 The <ManageNameIdentifierResponse> message SHOULD be signed or otherwise authenticated  
2064 and integrity protected by the protocol binding used to deliver the message.

2065 The <Issuer> of the response MUST contain the unique identifier of the responding provider, with a  
2066 Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:provider.

2067 The following schema fragment defines the <RegisterNameIdentifierResponse> element:

```
2068 <element name="ManageNameIdentifierResponse" type="samlp:StatusResponseType"/>
```

### 2069 **3.6.3 Processing Rules**

2070 The recipient **MUST** validate any signature present on the request or response message.

2071 If the request includes a `<NameIdentifier>` that the recipient does not recognize, the responding  
2072 provider **MAY** respond with a `<Status>` containing a second-level `<StatusCode>` of  
2073 `samlp:UnknownPrincipal`.

2074 If the `<TerminateIdentifier>` element is included in the request, the requesting provider is  
2075 indicating that (in the case of a service provider) it will no longer accept assertions from the identifier  
2076 provider or (in the case of an identity provider) it will no longer issue assertions to the service provider  
2077 about the principal. The receiving provider **MAY** perform any maintenance with the knowledge that the  
2078 relationship represented by the name identifier has been terminated. It **MAY** choose to invalidate the  
2079 active session(s) of a principal for whom a relationship has been terminated.

2080 If the service provider requests that its identifier be changed by including a `<NewIdentifier>` element,  
2081 the identity provider **MUST** include the `<NewIdentifier>` element's value as the  
2082 `SPProvidedIdentifier` when subsequently communicating to the service provider regarding this  
2083 principal.

2084 If the identity provider requests that its identifier be changed by including a `<NewIdentifier>` element,  
2085 the service provider **MUST** use the `<NewIdentifier>` element's value as the `<NameIdentifier>`  
2086 element value when subsequently communicating with the identity provider regarding this principal.

2087 In any case, the `<NameIdentifier>` value in the request and its associated `SPProvidedIdentifier`  
2088 attribute **MUST** contain the most recent name identifier information established between the providers for  
2089 the principal.

2090 In the case of a federated name identifier, the `NameQualifier` attribute **MUST** contain the unique  
2091 identifier of the identity provider., If the principal's identity federation is between the identity provider and  
2092 an affiliation group of which the service provider is a member, then the `SPNameQualifier` attribute  
2093 **MUST** contain the unique identifier of the affiliation group. Otherwise, it **MUST** contain the unique  
2094 identifier of the service provider.

2095 Changes to these identifiers may take a potentially significant amount of time to propagate through the  
2096 systems at both the requester and the responder. Implementations might wish to allow each party to  
2097 accept either identifier for some period of time following the successful completion of a name identifier  
2098 change. Not doing so could result in the inability of the principal to access resources.

2099 All other processing rules associated with the underlying request and response messages **MUST** be  
2100 observed.

### 2101 **3.7**

2102

### 2103 **3.8 Single Logout Protocol**

2104 The single logout protocol provides a message exchange protocol by which all sessions provided by a  
2105 particular session authority are near-simultaneously terminated. The single logout protocol is used either  
2106 when a principal logs out at a session participant or when the principal logs out directly at the  
2107 session authority. This protocol may also be used to logout a principal due to a timeout. The reason for  
2108 the logout event may be indicated through the `reason` attribute.

2109

2110 The principal may have established authenticated sessions both with the session authority, and  
2111 individual session participants, based on authentication assertions supplied by the session authority.

2112  
2113 When the principal invokes the single logout process at a session participant, the session participant  
2114 MUST send a <LogoutRequest> message to the session authority that provided the authentication  
2115 service related to that session at the session participant.  
2116  
2117 When either the principal invokes a logout at the session authority, or a session participant sends a  
2118 logout request to the session authority specifying that principal, the session authority MUST send a  
2119 <LogoutRequest> message to each session participant to which it provided authentication assertions  
2120 under its current session with the principal, with the exception of the session participant that sent the  
2121 <LogoutRequest> message to the session authority.

### 2123 3.8.1 Element <LogoutRequest>

2124 A session participant or session authority sends a <LogoutRequest> message to indicate that a  
2125 session has been terminated.

2126 The <LogoutRequest> message SHOULD be signed or otherwise authenticated and integrity  
2127 protected by the protocol binding used to deliver the message.

2128 This message has the complex type **LogoutRequestType**, which extends **RequestAbstractType**, and  
2129 adds the following elements and attributes:

2130 <NameIdentifier> [Required]

2131 The name identifier and associated attributes that specify the principal as currently recognized by  
2132 the identity and service providers prior to this request.

2133 <SessionIndex> [Optional]

2134 The identifier that indexes this session at the message recipient.

2135 NotOnOrAfter [Optional]

2136 The time at which the request expires.

2137 Reason [Optional]

2138 An indication of the reason for the logout, in the form of a URI reference.

2139 The following schema fragment defines the <LogoutRequest> element and associated  
2140 **LogoutRequestType** complex type:

```
2141 <element name="LogoutRequest" type="saml:LogoutRequestType"/>
2142 <complexType name="LogoutRequestType">
2143   <complexContent>
2144     <extension base="saml:RequestAbstractType">
2145       <sequence>
2146         <element ref="saml:NameIdentifier"/>
2147         <element ref="SessionIndex"/>
2148       </sequence>
2149       <attribute name="Reason" type="anyURI" minOccurs="0"/>
2150       <attribute name="NotOnOrAfter" type="dateTime" minOccurs="0"/>
2151     </extension>
2152   </complexContent>
2153 </complexType>
2154 <element name="SessionIndex" type="string" minOccurs="0"
2155 maxOccurs="unbounded"/>
```

## 2156 **3.8.2 Element <LogoutResponse>**

2157 The recipient of a <LogoutRequest> message MUST respond with a <LogoutResponse> message,  
2158 of type **StatusResponseType**, with no additional content specified.

2159 The <LogoutResponse> message SHOULD be signed or otherwise authenticated and integrity  
2160 protected by the protocol binding used to deliver the message.

2161 The following schema fragment defines the <LogoutResponse> element:

```
2162 <element name="LogoutResponse" type="samlp:StatusResponseType"/>
```

## 2163 **3.8.3 Processing Rules**

2164 The <Issuer> of either message in this protocol MUST contain the unique identifier of the requesting or  
2165 responding provider, with a `Format` value of `urn:oasis:names:tc:SAML:2.0:nameid-`  
2166 `format:provider`.

2167 Message recipients MUST validate any signature present on the messages specified in this protocol.

2168 The message sender MAY use the `Reason` attribute to indicate the reason for sending the  
2169 <LogoutRequest>. Other values MAY be agreed upon between participants, but the following values  
2170 are defined directly by this specification for use by all message senders:

2171 `urn:oasis:names:tc:SAML:2.0:logout:user`

2172 Specifies that the message is being sent because the principal wishes to terminate the indicated  
2173 session.

2174 `urn:oasis:names:tc:SAML:2.0:logout:admin`

2175 Specifies that the message is being sent because an administrator wishes to terminate the indicated  
2176 session for that principal.

2177 All other processing rules associated with the underlying request and response messages MUST be  
2178 observed.

### 2179 **3.8.3.1 Session Participant Rules**

2180 When a session participant receives a <LogoutRequest>, the session participant MUST authenticate  
2181 the message.. If the sender is the authority that provided an assertion linked to the principal's current  
2182 session, the session participant MUST invalidate the principal's session(s) referred to by the  
2183 <NameIdentifier> element, and any <SessionIndex> elements supplied in the message.  
2184

2185 The session participant MUST apply the logout request message to any assertion that meets the  
2186 following conditions, even if the assertion arrives after the logout request:

- 2187 • The <SessionIndex> of the assertion's statements matches one specified in the logout request.
- 2188 • The assertion would otherwise be valid
- 2189 • The logout request has not yet expired (determined by examining the `NotOnOrAfter` attribute on  
2190 the message).

### 2191 3.8.3.2 Session Authority Rules

2192 When a session authority receives a `<LogoutRequest>`, the session authority MUST authenticate the  
2193 sender. If the sender is a session participant to which the session authority provided an assertion for the  
2194 current session, then the session authority SHOULD do the following:

- 2195 • Send a `<LogoutRequest>` message to each session participant for which the session authority  
2196 provided assertions in the current session, *other than the originator of a current*  
2197 `<LogoutRequest>`.
- 2198 • Send a `<LogoutRequest>` message to any session authority on behalf of whom the session  
2199 authority proxied the user's authentication, unless the second authority is the originator of the  
2200 `<LogoutRequest>`.
- 2201 • Terminate the principal's current session as specified by the `<NameIdentifier>` element, and  
2202 any `<SessionIndex>` elements present in the logout request message.

2203 It should be noted that a session authority MAY initiate a logout for reasons other than having received a  
2204 `<LogoutRequest>` from a session participant – these include, but are not limited to:

- 2205 • If some timeout period was agreed out-of-band with an individual session participant, the session  
2206 authority MAY send a `<LogoutRequest>` to that individual participant alone.
- 2207 • An agreed global timeout period has been exceeded.
- 2208 • The principal, or some other trusted entity has requested logout of the principal, directly at the  
2209 session authority.
- 2210 • The session authority has determined that the principal's credentials may have been compromised.

2211 When constructing a logout request message, the session authority MUST set the value of the  
2212 `NotOnOrAfter` attribute of the message to a time value, indicating an expiration time for the message.

2213 In addition to the values specified in section 3.6.3 for the `Reason` attribute, the following values are also  
2214 available for use by the session authority only:

2215 `urn:oasis:names:tc:SAML:2.0:logout:global-timeout`

2216 Specifies that the message is being sent because of the global session timeout interval period  
2217 being exceeded.

2218 `urn:oasis:names:tc:SAML:2.0:logout:sp-timeout`

2219 Specifies that the message is being sent because a timeout interval period agreed between a  
2220 participant and the authority has been exceeded.

2221 If an error occurs during this further processing of the logout (for example, relying session participants  
2222 may not all implement the particular single logout protocol binding used by the requesting session  
2223 participant), then the session authority MUST respond to the original requester with a  
2224 `<LogoutResponse>` message, indicating the status of the logout request. The value  
2225 `samlp:UnsupportedBinding` is provided for a second-level `<samlp:StatusCode>`, indicating that a  
2226 session participant should retry the `<LogoutRequest>` using a different protocol binding.

## 2227 3.9 Name Identifier Mapping Protocol

2228 When an entity that shares an identifier for a principal with an identity provider wishes to obtain a name  
2229 identifier for the same principal in a particular format or federation namespace, it can send a request to  
2230 the identity provider using this protocol.

2231 For example, a service provider that wishes to communicate with another service provider with whom it  
2232 does not share an identity federation for the principal can use an identity provider that shares an identity  
2233 federation for the principal with both service providers to map from its own federated identifier to a new  
2234 identifier, generally encrypted, with which it can communicate with the second service provider.

2235 Regardless of the type of identifier involved, the mapped identifier SHOULD be encrypted into an  
2236 <EncryptedIdentifier> element unless a specific deployment dictates such protection is  
2237 unnecessary.

### 2238 **3.9.1 Element <NameIdentifierMappingRequest>**

2239 To request an alternate name identifier for a principal from an identity provider, a requester sends an  
2240 <NameIdentifierMappingRequest> message. This message has the complex type  
2241 **NameIdentifierMappingRequestType**, which extends **RequestAbstractType** and adds the following  
2242 element:

2243 <BaseIdentifier> or <NameIdentifier> or <EncryptedIdentifier> [Required]

2244 The identifier and associated attributes that specify the principal as currently recognized by the  
2245 requester and the responder.

2246 <NameIDPolicy>

2247 The format and optional name qualifier that describes the requirements for the identifier to be  
2248 returned.

2249 The message SHOULD be signed or otherwise authenticated and integrity protected by the protocol  
2250 binding used to deliver the message.

2251 The following schema fragment defines the <NameIdentifierMappingRequest> element and its  
2252 **NameIdentifierMappingRequestType** complex type:

```
2253 <element name="NameIdentifierMappingRequest"  
2254 type="samlp:NameIdentifierMappingRequestType"/>  
2255 <complexType name="NameIdentifierMappingRequestType">  
2256 <complexContent>  
2257 <extension base="samlp:RequestAbstractType">  
2258 <sequence>  
2259 <choice>  
2260 <element ref="saml:BaseIdentifier"/>  
2261 <element ref="saml:NameIdentifier"/>  
2262 <element ref="saml:EncryptedIdentifier"/>  
2263 </choice>  
2264 <element ref="samlp:NameIDPolicy"/>  
2265 </sequence>  
2266 </extension>  
2267 </complexContent>  
2268 </complexType>
```

### 2269 **3.9.2 Element <NameIdentifierMappingResponse>**

2270 The recipient of a <NameIdentifierMappingRequest> message MUST respond with a  
2271 <NameIdentifierMappingResponse> message. This message has the complex type  
2272 **NameIdentifierMappingRequestType**, which extends **RequestAbstractType** and adds the following  
2273 element:

2274 <NameIdentifier> or <EncryptedIdentifier> [Required]

2275 The identifier and associated attributes that specify the principal in the manner requested, usually in  
2276 encrypted form.

2277 The message SHOULD be signed or otherwise authenticated and integrity protected by the protocol  
2278 binding used to deliver the message.

2279 The <Issuer> of the response MUST contain the unique identifier of the responding provider, with a  
2280 Format value of urn:oasis:names:tc:SAML:2.0:nameid-format:provider.

2281 The following schema fragment defines the <NameIdentifierMappingResponse> element and its  
2282 **NameIdentifierMappingResponseType** complex type:

```
2283 <element name="NameIdentifierMappingResponse"  
2284 type="samlp:NameIdentifierMappingResponseType"/>  
2285 <complexType name="NameIdentifierMappingResponseType">  
2286   <complexContent>  
2287     <extension base="samlp:StatusResponseType">  
2288       <choice>  
2289         <element ref="saml:NameIdentifier">  
2290         <element ref="saml:EncryptedIdentifier">  
2291       </choice>  
2292     </extension>  
2293   </complexContent>  
2294 </complexType>
```

### 2295 3.9.3 Processing Rules

2296 The recipient MUST validate any signature present on the request or response message.

2297 If the responder does not recognize the principal identified in the request, it MAY respond with a  
2298 <Status> containing a second-level <StatusCode> of samlp:UnknownPrincipal.

2299 At the responder's discretion, the samlp:InvalidNameIDPolicy status code MAY be returned to  
2300 indicate an inability or unwillingness to supply an identifier in the requested format.

2301 All other processing rules associated with the underlying request and response messages MUST be  
2302 observed.

---

## 2303 4 SAML Versioning

2304 The SAML specification set is versioned in two independent ways. Each is discussed in the following  
2305 sections, along with processing rules for detecting and handling version differences, when applicable.  
2306 Also included are guidelines on when and why specific version information is expected to change in  
2307 future revisions of the specification.

2308 When version information is expressed as both a Major and Minor version, it may be expressed  
2309 discretely, or in the form *Major.Minor*. The version number *Major<sub>B</sub>.Minor<sub>B</sub>* is higher than the version  
2310 number *Major<sub>A</sub>.Minor<sub>A</sub>* if and only if:

2311  $Major_B > Major_A \vee ( ( Major_B = Major_A ) \wedge Minor_B > Minor_A )$

### 2312 4.1 SAML Specification Set Version

2313 Each release of the SAML specification set will contain a major and minor version designation describing  
2314 its relationship to earlier and later versions of the specification set. The version will be expressed in the  
2315 content and filenames of published materials, including the specification set document(s), and XML  
2316 schema instance(s). There are no normative processing rules surrounding specification set versioning,  
2317 since it merely encompasses the collective release of normative specification documents which  
2318 themselves contain processing rules.

2319 The overall size and scope of changes to the specification set document(s) will informally dictate whether  
2320 a set of changes constitutes a major or minor revision. In general, if the specification set is backwards  
2321 compatible with an earlier specification set (that is, valid older messages, protocols, and semantics  
2322 remain valid), then the new version will be a minor revision. Otherwise, the changes will constitute a  
2323 major revision. Note that SAML V1.1 has made one backwards-incompatible change to SAML V1.0,  
2324 described in Section .

#### 2325 4.1.1 Schema Version

2326 As a non-normative documentation mechanism, any XML schema instances published as part of the  
2327 specification set will contain a schema "version" attribute in the form *Major.Minor*, reflecting the  
2328 specification set version in which it has been published. Validating implementations MAY use the  
2329 attribute as a means of distinguishing which version of a schema is being used to validate messages, or  
2330 to support a multiplicity of versions of the same logical schema.

#### 2331 4.1.2 SAML Assertion Version

2332 The SAML <Assertion> element contains attributes for expressing the major and minor version of the  
2333 assertion using a pair of integers. Each version of the SAML specification set will be construed so as to  
2334 document the syntax, semantics, and processing rules of the assertions of the same version. That is,  
2335 specification set version 1.0 describes assertion version 1.0, and so on.

2336 There is explicitly NO relationship between the assertion version and the SAML assertion XML  
2337 namespace that contains the schema definitions for that assertion version.

2338 The following processing rules apply:

- 2339 • A SAML authority MUST NOT issue any assertion with an assertion version number not supported  
2340 by the authority.
- 2341 • A SAML relying party MUST NOT process any assertion with a major assertion version number not  
2342 supported by the relying party.

- 2343 • A SAML relying party MAY process or MAY reject an assertion whose minor assertion version  
2344 number is higher than the minor assertion version number supported by the relying party. However,  
2345 all assertions that share a major assertion version number MUST share the same general processing  
2346 rules and semantics, and MAY be treated in a uniform way by an implementation. That is, if a V1.1  
2347 assertion shares the syntax of a V1.0 assertion, an implementation MAY treat the assertion as a V1.0  
2348 assertion without ill effect.

### 2349 **4.1.3 SAML Protocol Version**

2350 The SAML protocol <Request> and <Response> elements contain attributes for expressing the major  
2351 and minor version of the request or response message using a pair of integers. Each version of the  
2352 SAML specification set will be construed so as to document the syntax, semantics, and processing rules  
2353 of the protocol messages of the same version. That is, specification set version 1.0 describes request  
2354 and response version V1.0, and so on.

2355 There is explicitly NO relationship between the protocol version and the SAML protocol XML namespace  
2356 that contains the schema definitions for protocol messages for that protocol version.

2357 The version numbers used in SAML protocol <Request> and <Response> elements will be the same  
2358 for any particular revision of the SAML specification set.

#### 2359 **4.1.3.1 Request Version**

2360 The following processing rules apply to requests:

- 2361 • A SAML requester SHOULD issue requests with the highest request version supported by both the  
2362 SAML requester and the SAML responder.
- 2363 • If the SAML requester does not know the capabilities of the SAML responder, then it should assume  
2364 that it supports requests with the highest request version supported by the requester.
- 2365 • A SAML requester MUST NOT issue a request message with a request version number matching a  
2366 response version number that the requester does not support.
- 2367 • A SAML responder MUST reject any request with a major request version number not supported by  
2368 the responder.
- 2369 • A SAML responder MAY process or MAY reject any request whose minor request version number is  
2370 higher than the highest supported request version that it supports. However, all requests that share a  
2371 major request version number MUST share the same general processing rules and semantics, and  
2372 MAY be treated in a uniform way by an implementation. That is, if a V1.1 request shares the syntax  
2373 of a V1.0 request, a responder MAY treat the request message as a V1.0 request without ill effect.

### 2374 **4.1.4 Response Version**

2375 The following processing rules apply to responses:

- 2376 • A SAML responder MUST NOT issue a response message with a response version number higher  
2377 than the request version number of the corresponding request message.
- 2378 • A SAML responder MUST NOT issue a response message with a major response version number  
2379 lower than the major request version number of the corresponding request message except to report  
2380 the error `RequestVersionTooHigh`.

2381 An error response resulting from incompatible SAML protocol versions MUST result in reporting a top-  
2382 level <StatusCode> value of `VersionMismatch`, and MAY result in reporting one of the following

2383 second-level values: RequestVersionTooHigh, RequestVersionTooLow, or  
2384 RequestVersionDeprecated.

## 2385 4.1.5 Permissible Version Combinations

2386 ~~In general, a~~ assertions of a particular major version ~~may~~ appear only in response messages of the same  
2387 major version, as permitted by the importation of the SAML assertion namespace into the SAML protocol  
2388 schema. ~~Future versions of this specification are expected to explicitly describe the permitted~~  
2389 ~~combinations across major versions.~~

2390 ~~Specifically, this permits. For example,~~ a V1.1 assertion MAY ~~to~~ appear in a V1.0 response message,  
2391 and a V1.0 assertion ~~to appear~~ in a V1.1 response message, if the appropriate assertion schema is  
2392 referenced during namespace importation. But a V1.0 assertion MUST NOT appear in a V2.0 response  
2393 message because they are of different major versions.

## 2394 4.2 SAML Namespace Version

2395 XML schema instances and "qualified names" (QNames) published as part of the specification set  
2396 contain one or more target namespaces into which the type, element, and attribute definitions are  
2397 placed. Each namespace is distinct from the others, and represents, in shorthand, the structural and  
2398 syntactical definitions that make up that part of the specification.

2399 The namespace URIs defined by the specification set will generally contain version information of the  
2400 form *Major.Minor* somewhere in the URI. The major and minor version in the URI MUST correspond to  
2401 the major and minor version of the specification set in which the namespace is first introduced and  
2402 defined. This information is not typically consumed by an XML processor, which treats the namespace  
2403 opaquely, but is intended to communicate the relationship between the specification set and the  
2404 namespaces it defines.

2405 As a general rule, implementers can expect the namespaces (and the associated schema definitions)  
2406 defined by a major revision of the specification set to remain valid and stable across minor revisions of  
2407 the specification. New namespaces may be introduced, and when necessary, old namespaces replaced,  
2408 but this is expected to be rare. In such cases, the older namespaces and their associated definitions  
2409 should be expected to remain valid until a major specification set revision.

### 2410 4.2.1 Schema Evolution

2411 In general, maintaining namespace stability while adding or changing the content of a schema are  
2412 competing goals. While certain design strategies can facilitate such changes, it is complex to predict how  
2413 older implementations will react to any given change, making forward compatibility difficult to achieve.  
2414 Nevertheless, the right to make such changes in minor revisions is reserved, in the interest of  
2415 namespace stability. Except in special circumstances (for example to correct major deficiencies or fix  
2416 errors), implementations should expect forward compatible schema changes in minor revisions, allowing  
2417 new messages to validate against older schemas.

2418 Implementations SHOULD expect and be prepared to deal with new extensions and message types in  
2419 accordance with the processing rules laid out for those types. Minor revisions MAY introduce new types  
2420 that leverage the extension facilities described in Section SAML Extensions. Older implementations  
2421 SHOULD reject such extensions gracefully when they are encountered in contexts that dictate mandatory  
2422 semantics. Examples include new query, statement, or condition types.

2423

## 5 SAML and XML Signature Syntax and Processing

2424 SAML assertions and SAML protocol request and response messages may be signed, with the following  
2425 benefits:

- 2426 • An assertion signed by the SAML authority supports:
  - 2427 – Assertion integrity.
  - 2428 – Authentication of the SAML authority to a SAML relying party.
  - 2429 – If the signature is based on the SAML authority's public-private key pair, then it also provides for  
2430 non-repudiation of origin.
- 2431 • A SAML protocol request or response message signed by the message originator supports:
  - 2432 – Message integrity.
  - 2433 – Authentication of message origin to a destination.
  - 2434 – If the signature is based on the originator's public-private key pair, then it also provides for non-  
2435 repudiation of origin.

2436 A digital signature is not always required in SAML. For example, it may not be required in the following  
2437 situations:

- 2438 • In some circumstances signatures may be "inherited," such as when an unsigned assertion gains  
2439 protection from a signature on the containing protocol response message. "Inherited" signatures  
2440 should be used with care when the contained object (such as the assertion) is intended to have a  
2441 non-transitory lifetime. The reason is that the entire context must be retained to allow validation,  
2442 exposing the XML content and adding potentially unnecessary overhead.
- 2443 • The SAML relying party or SAML requester may have obtained an assertion or protocol message  
2444 from the SAML authority or SAML responder directly (with no intermediaries) through a secure  
2445 channel, with the SAML authority or SAML responder having authenticated to the relying party or  
2446 SAML responder by some means other than a digital signature.

2447 Many different techniques are available for "direct" authentication and secure channel establishment  
2448 between two parties. The list includes TLS/SSL, HMAC, password-based mechanisms, etc. In addition,  
2449 the applicable security requirements depend on the communicating applications and the nature of the  
2450 assertion or message transported.

2451 It is recommended that, in all other contexts, digital signatures be used for assertions and request and  
2452 response messages. Specifically:

- 2453 • A SAML assertion obtained by a SAML relying party from an entity other than the SAML authority  
2454 SHOULD be signed by the SAML authority.
- 2455 • A SAML protocol message arriving at a destination from an entity other than the originating site  
2456 SHOULD be signed by the origin site.

2457 Profiles may specify alternative signature mechanisms such as S/MIME or signed Java objects that  
2458 contain SAML documents. Caveats about retaining context and interoperability apply. XML Signatures  
2459 are intended to be the primary SAML signature mechanism, but the specification attempts to ensure  
2460 compatibility with profiles that may require other mechanisms.

2461 Unless a profile specifies an alternative signature mechanism, enveloped XML Digital Signatures MUST  
2462 be used if signing.

## 2463 **5.1 Signing Assertions**

2464 All SAML assertions MAY be signed using the XML Signature. This is reflected in the assertion schema  
2465 as described in Section Assertions.

## 2466 **5.2 Request/Response Signing**

2467 All SAML protocol request and response messages MAY be signed using the XML Signature. This is  
2468 reflected in the schema as described in Sections Requests and Responses and .

## 2469 **5.3 Signature Inheritance**

2470 A SAML assertion may be embedded within another SAML element, such as an enclosing `<Assertion>`  
2471 or a `<Request>` or `<Response>`, which may be signed. When a SAML assertion does not contain a  
2472 `<ds:Signature>` element, but is contained in an enclosing SAML element that contains a  
2473 `<ds:Signature>` element, and the signature applies to the `<Assertion>` element and all its children,  
2474 then the assertion can be considered to inherit the signature from the enclosing element. The resulting  
2475 interpretation should be equivalent to the case where the assertion itself was signed with the same key  
2476 and signature options.

2477 Many SAML use cases involve SAML XML data enclosed within other protected data structures such as  
2478 signed SOAP messages, S/MIME packages, and authenticated SSL connections. SAML profiles may  
2479 define additional rules for interpreting SAML elements as inheriting signatures or other authentication  
2480 information from the surrounding context, but no such inheritance should be inferred unless specifically  
2481 identified by the profile.

## 2482 **5.4 XML Signature Profile**

2483 The XML Signature specification [XMLSig] calls out a general XML syntax for signing data with flexibility  
2484 and many choices. This section details the constraints on these facilities so that SAML processors do not  
2485 have to deal with the full generality of XML Signature processing. This usage makes specific use of the  
2486 **xsd:ID**-typed attributes optionally present on the root elements to which signatures can apply: the  
2487 `AssertionID` attribute on `<Assertion>`, the `RequestID` attribute on `<Request>`, and the  
2488 `ResponseID` attribute on `<Response>`. These three attributes are collectively referred to in this section  
2489 as the identifier attributes.

### 2490 **5.4.1 Signing Formats and Algorithms**

2491 XML Signature has three ways of relating a signature to a document: enveloping, enveloped, and  
2492 detached.

2493 SAML assertions and protocols MUST use enveloped signatures when signing assertions and protocol  
2494 messages. SAML processors SHOULD support the use of RSA signing and verification for public key  
2495 operations in accordance with the algorithm identified by <http://www.w3.org/2000/09/xmldsig#rsa-sha1>.

### 2496 **5.4.2 References**

2497 Signed SAML assertions and protocol messages MUST supply a value for the identifier attribute on the  
2498 root element (`<Assertion>`, `<Request>`, or `<Response>`). The assertion's or message's root element  
2499 may or may not be the root element of the actual XML document containing the signed assertion or  
2500 message.

2501 Signatures MUST contain a single `<ds:Reference>` containing a URI reference to the identifier  
2502 attribute value of the root element of the message being signed. For example, if the attribute value is  
2503 "foo", then the URI attribute in the `<ds:Reference>` element MUST be "#foo".

### 2504 **5.4.3 Canonicalization Method**

2505 SAML implementations SHOULD use Exclusive Canonicalization, with or without comments, both in the  
2506 `<ds:CanonicalizationMethod>` element of `<ds:SignedInfo>`, and as a `<ds:Transform>`  
2507 algorithm. Use of Exclusive Canonicalization ensures that signatures created over SAML messages  
2508 embedded in an XML context can be verified independent of that context.

### 2509 **5.4.4 Transforms**

2510 Signatures in SAML messages SHOULD NOT contain transforms other than the enveloped signature  
2511 transform (with the identifier <http://www.w3.org/2000/09/xmlsig#enveloped-signature>) or the exclusive  
2512 canonicalization transforms (with the identifier <http://www.w3.org/2001/10/xml-exc-c14n#> or  
2513 <http://www.w3.org/2001/10/xml-exc-c14n#WithComments>).

2514 Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid. If they do  
2515 not, verifiers MUST ensure that no content of the SAML message is excluded from the signature. This  
2516 can be accomplished by establishing out-of-band agreement as to what transforms are acceptable, or by  
2517 applying the transforms manually to the content and reverifying the result as consisting of the same  
2518 SAML message.

### 2519 **5.4.5 KeyInfo**

2520 XML Signature [XMLSig] defines usage of the `<ds:KeyInfo>` element. SAML does not require the  
2521 use of `<ds:KeyInfo>` nor does it impose any restrictions on its use. Therefore, `<ds:KeyInfo>` MAY  
2522 be absent.

### 2523 **5.4.6 Binding Between Statements in a Multi-Statement Assertion**

2524 Use of signing does not affect semantics of statements within assertions in any way, as stated in Section  
2525 SAML Assertions.

### 2526 **5.4.7 Example**

2527 Following is an example of a signed response containing a signed assertion. Line breaks have been  
2528 added for readability; the signatures are not valid and cannot be successfully verified.

```
2529 <Response  
2530   IssueInstant="2003-04-17T00:46:02Z"  
2531   MajorVersion="1"  
2532   MinorVersion="1"  
2533   Recipient="www.opensaml.org"  
2534   ResponseID="_c7055387-af61-4fce-8b98-e2927324b306"  
2535   xmlns="urn:oasis:names:tc:SAML:1.0:protocol"  
2536   xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"  
2537   xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
2538   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">  
2539   <ds:Signature  
2540     xmlns:ds="http://www.w3.org/2000/09/xmlsig#">  
2541     <ds:SignedInfo>  
2542       <ds:CanonicalizationMethod  
2543         Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />  
2544       <ds:SignatureMethod
```





2667

## 6 SAML Extensions

2668 The SAML schemas support extensibility. An example of an application that extends SAML assertions is  
2669 the Liberty Protocols and Schema Specification [LibertyProt]. The following sections explain how to use  
2670 the extensibility features in SAML to create extension schemas.

2671 Note that elements in the SAML schemas are blocked from substitution, which means that no SAML  
2672 elements can serve as the head element of a substitution group. However, SAML types are not defined  
2673 as *final*, so that all SAML types MAY be extended and restricted. The following sections discuss only  
2674 elements and types that have been specifically designed to support extensibility.

### 2675 6.1 Assertion Schema Extension

2676 The SAML assertion schema is designed to permit separate processing of the assertion package and the  
2677 statements it contains, if the extension mechanism is used for either part.

2678 The following elements are intended specifically for use as extension points in an extension schema;  
2679 their types are set to *abstract*, and are thus usable only as the base of a derived type:

2680 • <Condition>

2681 • <Statement>

2682 • The following elements that are directly usable as part of SAML MAY be extended:

2683 • <AuthenticationStatement>

2684 • <AuthorizationDecisionStatement>

2685 • <AttributeStatement>

2686 • <AudienceRestrictionCondition>

2687 The following elements are defined to allow elements from arbitrary namespaces within them, which  
2688 serves as a built-in extension point without requiring an extension schema:

2689 • <BaseIdentifier>

2690 • <SubjectConfirmationData>

2691 • <AttributeValue>

2692 • <Advice>

2693 • <AuthnContext>

### 2694 6.2 Protocol Schema Extension

2695 The following SAML protocol elements are intended specifically for use as extension points in an  
2696 extension schema; their types are set to *abstract*, and are thus usable only as the base of a derived  
2697 type:

2698 • <Query>

2699 • <SubjectQuery>

2700 The following elements that are directly usable as part of SAML MAY be extended:

- 2701 • <Request>
- 2702 • <AuthenticationQuery>
- 2703 • <AuthorizationDecisionQuery>
- 2704 • <AttributeQuery>
- 2705 • <Response>

---

## 2706 7 SAML-Defined Identifiers

2707 The following sections define URI-based identifiers for common authentication methods, resource access  
2708 actions, and subject name identifier formats.

2709 Where possible an existing URN is used to specify a protocol. In the case of IETF protocols the URN of  
2710 the most current RFC that specifies the protocol is used. URI references created specifically for SAML  
2711 have one of the following stems:

```
2712 urn:oasis:names:tc:SAML:1.0:  
2713 urn:oasis:names:tc:SAML:1.1:
```

### 2714 7.1 Authentication Method Identifiers

2715 The `AuthenticationMethod` attribute of an `<AuthenticationStatement>` and the  
2716 `<SubjectConfirmationMethod>` element of a SAML subject perform different functions, although  
2717 both can refer to the same underlying mechanisms. An authentication statement with an  
2718 `AuthenticationMethod` attribute describes an authentication act that occurred in the past. The  
2719 `AuthenticationMethod` attribute indicates how that authentication was done. Note that the  
2720 authentication statement does not provide the means to perform that authentication, such as a password,  
2721 key, or certificate.

2722 In contrast, `<SubjectConfirmationMethod>` is a part of the `<SubjectConfirmation>` element,  
2723 which is an optional part of a SAML subject. `<SubjectConfirmation>` is used to allow the SAML  
2724 relying party to confirm that the request or message came from a system entity that corresponds to the  
2725 subject in the statement or query. The `<SubjectConfirmationMethod>` element indicates the method  
2726 that the relying party can use to do this in the future. This may or may not have any relationship to an  
2727 authentication that was performed previously. Unlike the authentication method, the subject confirmation  
2728 method may be accompanied by some piece of information, such as a certificate or key, that will allow  
2729 the relying party to perform the necessary check.

2730 Subject confirmation methods are defined in the SAML profiles in which they are used; see the SAML  
2731 profiles specification [SAMLProf] for more information. Additional methods may be added by defining  
2732 new profiles or by private agreement.

2733 The following identifiers refer to SAML-specified authentication methods.

#### 2734 7.1.1 Password

2735 **URI:** urn:oasis:names:tc:SAML:1.0:am:password

2736 The authentication was performed by means of a password.

#### 2737 7.1.2 Kerberos

2738 **URI:** urn:ietf:rfc:1510

2739 The authentication was performed by means of the Kerberos protocol [RFC 1510], an instantiation of the  
2740 Needham-Schroeder symmetric key authentication mechanism [Needham78].

#### 2741 7.1.3 Secure Remote Password (SRP)

2742 **URI:** urn:ietf:rfc:2945

2743 The authentication was performed by means of Secure Remote Password protocol as specified in [RFC  
2744 2945].

#### 2745 **7.1.4 Hardware Token**

2746 **URI:** urn:oasis:names:tc:SAML:1.0:am:HardwareToken

2747 The authentication was performed using some (unspecified) hardware token.

#### 2748 **7.1.5 SSL/TLS Certificate Based Client Authentication:**

2749 **URI:** urn:ietf:rfc:2246

2750 The authentication was performed using either the SSL or TLS protocol with certificate-based client  
2751 authentication. TLS is described in [RFC 2246].

#### 2752 **7.1.6 X.509 Public Key**

2753 **URI:** urn:oasis:names:tc:SAML:1.0:am:X509-PKI

2754 The authentication was performed by some (unspecified) mechanism on a key authenticated by means  
2755 of an X.509 PKI [X.500][PKIX]. It may have been one of the mechanisms for which a more specific  
2756 identifier has been defined below.

#### 2757 **7.1.7 PGP Public Key**

2758 **URI:** urn:oasis:names:tc:SAML:1.0:am:PGP

2759 The authentication was performed by some (unspecified) mechanism on a key authenticated by means  
2760 of a PGP web of trust [PGP]. It may have been one of the mechanisms for which a more specific  
2761 identifier has been defined below.

#### 2762 **7.1.8 SPKI Public Key**

2763 **URI:** urn:oasis:names:tc:SAML:1.0:am:SPKI

2764 The authentication was performed by some (unspecified) mechanism on a key authenticated by means  
2765 of a SPKI PKI [SPKI]. It may have been one of the mechanisms for which a more specific identifier has  
2766 been defined below.

#### 2767 **7.1.9 XKMS Public Key**

2768 **URI:** urn:oasis:names:tc:SAML:1.0:am:XKMS

2769 The authentication was performed by some (unspecified) mechanism on a key authenticated by means  
2770 of a XKMS trust service [XKMS]. It may have been one of the mechanisms for which a more specific  
2771 identifier has been defined below.

#### 2772 **7.1.10 XML Digital Signature**

2773 **URI:** urn:ietf:rfc:3075

2774 The authentication was performed by means of an XML digital signature [RFC 3075].

2775 **7.1.11 Authentication Context**

2776 **URI:** urn:oasis:names:tc:SAML:2.0:am:authncontext

2777 The authentication method is described by the proximal <AuthnContext> element.

2778 **7.1.12 Unspecified**

2779 **URI:** urn:oasis:names:tc:SAML:1.0:am:unspecified

2780 The authentication was performed by an unspecified means.

2781 **7.2 Action Namespace Identifiers**

2782 The following identifiers MAY be used in the `Namespace` attribute of the <Action> element (see  
2783 Section Element <Action>) to refer to common sets of actions to perform on resources.

2784 **7.2.1 Read/Write/Execute/Delete/Control**

2785 **URI:** urn:oasis:names:tc:SAML:1.0:action:rwedc

2786 Defined actions:

2787 `Read Write Execute Delete Control`

2788 These actions are interpreted as follows:

2789 `Read`

2790 The subject may read the resource.

2791 `Write`

2792 The subject may modify the resource.

2793 `Execute`

2794 The subject may execute the resource.

2795 `Delete`

2796 The subject may delete the resource.

2797 `Control`

2798 The subject may specify the access control policy for the resource.

2799 **7.2.2 Read/Write/Execute/Delete/Control with Negation**

2800 **URI:** urn:oasis:names:tc:SAML:1.0:action:rwedc-negation

2801 Defined actions:

2802 `Read Write Execute Delete Control ~Read ~Write ~Execute ~Delete ~Control`

2803 The actions specified in Section Read/Write/Execute/Delete/Control are interpreted in the same manner  
2804 described there. Actions prefixed with a tilde (~) are negated permissions and are used to affirmatively  
2805 specify that the stated permission is denied. Thus a subject described as being authorized to perform the  
2806 action `~Read` is affirmatively denied read permission.

2807 A SAML authority MUST NOT authorize both an action and its negated form.

### 2808 **7.2.3 Get/Head/Put/Post**

2809 **URI:** urn:oasis:names:tc:SAML:1.0:action:ghpp

2810 Defined actions:

2811 GET HEAD PUT POST

2812 These actions bind to the corresponding HTTP operations. For example a subject authorized to perform  
2813 the GET action on a resource is authorized to retrieve it.

2814 The GET and HEAD actions loosely correspond to the conventional read permission and the PUT and  
2815 POST actions to the write permission. The correspondence is not exact however since an HTTP GET  
2816 operation may cause data to be modified and a POST operation may cause modification to a resource  
2817 other than the one specified in the request. For this reason a separate Action URI reference specifier is  
2818 provided.

### 2819 **7.2.4 UNIX File Permissions**

2820 **URI:** urn:oasis:names:tc:SAML:1.0:action:unix

2821 The defined actions are the set of UNIX file access permissions expressed in the numeric (octal)  
2822 notation.

2823 The action string is a four-digit numeric code:

2824 *extended user group world*

2825 Where the *extended* access permission has the value

2826 +2 if sgid is set

2827 +4 if suid is set

2828 The *user group* and *world* access permissions have the value

2829 +1 if execute permission is granted

2830 +2 if write permission is granted

2831 +4 if read permission is granted

2832 For example, 0754 denotes the UNIX file access permission: user read, write and execute; group read  
2833 and execute; and world read.

## 2834 **7.3 NameIdentifier Format Identifiers**

2835 The following identifiers MAY be used in the `Format` attribute of the `<NameIdentifier>` element (see  
2836 Section ) to refer to common formats for the content of the `<NameIdentifier>` element and the  
2837 associated processing rules, if any.

2838 **Note:** Several identifiers that were deprecated in V1.1 have been removed for V2.0 of  
2839 SAML.

### 2840 **7.3.1 Unspecified**

2841 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified

2842 The interpretation of the content of the element is left to individual implementations.

### 2843 **7.3.2 Email Address**

2844 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress

2845 Indicates that the content of the element is in the form of an email address, specifically "addr-spec" as  
2846 defined in IETF RFC 2822 [RFC 2822] §3.4.1. An addr-spec has the form local-part@domain. Note that  
2847 an addr-spec has no phrase (such as a common name) before it, has no comment (text surrounded in  
2848 parentheses) after it, and is not surrounded by "<" and ">".

### 2849 **7.3.3 X.509 Subject Name**

2850 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName

2851 Indicates that the content of the element is in the form specified for the contents of the  
2852 <ds:X509SubjectName> element in the XML Signature Recommendation [XMLSig]. Implementors  
2853 should note that the XML Signature specification specifies encoding rules for X.509 subject names that  
2854 differ from the rules given in IETF RFC 2253 [RFC 2253].

### 2855 **7.3.4 Windows Domain Qualified Name**

2856 **URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName

2857 Indicates that the content of the element is a Windows domain qualified name. A Windows domain  
2858 qualified user name is a string of the form "DomainName\UserName". The domain name and "\"  
2859 separator MAY be omitted.

### 2860 **7.3.5 Kerberos Principal Name**

2861 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos

2862 Indicates that the content of the element is in the form of a Kerberos principal name using the format  
2863 name[/instance]@REALM. The syntax, format and characters allowed for the name, instance, and  
2864 realm are described in [RFC 1510].

### 2865 **7.3.6 Provider Identifier**

2866 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:provider

2867 Indicates that the content of the element is the identifier of a provider of SAML-based services (such as a  
2868 SAML authority) or a participant in SAML profiles (such as a service provider supporting the browser  
2869 profiles). Such an identifier can be used to make assertions about system entities that can issue SAML  
2870 requests, responses, and assertions.

### 2871 **7.3.7 Federated Identifier**

2872 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:federated

2873 Indicates that the content of the element is a persistent opaque identifier that corresponds to an identity  
2874 federation between an identity provider and a service provider (or affiliation of service providers).  
2875 Federated name identifiers generated by identity providers MUST be constructed using pseudo-random  
2876 values that have no discernible correspondence with the subject's actual identifier (for example,  
2877 username). The intent is to create a non-public pseudonym to prevent the discovery of the subject's  
2878 identity or activities. Federated name identifier values MUST NOT exceed a length of 256 characters.

2879 The element's content MUST contain the most recent identifier of the subject set by the identity provider.

2880 The element's `NameQualifier` attribute, if present, MUST contain the name of the identity provider  
2881 participating in the identity federation. It MAY be omitted if the value can be derived from the context of  
2882 the message containing the element, such as the issuer of an assertion.

2883 The element's `SPNameQualifier` attribute, if present, MUST contain the name of the service provider  
2884 or affiliation of providers participating in the identity federation. It MAY be omitted if the element is  
2885 contained in a message intended only for consumption directly by the service provider, and the value  
2886 would be the name of that service provider.

2887 The element's `SPProvidedIdentifier` attribute MUST contain the alternative identifier of the subject  
2888 most recently set by the service provider or affiliation, if any. If no such identifier has been established,  
2889 than the attribute MUST be omitted.

2890 Federated identifiers are intended as a privacy protection; as such they MUST NOT be shared in clear  
2891 text with providers other than the providers that have established the identity federation. Furthermore,  
2892 they MUST NOT appear in log files or similar locations without appropriate controls and protections.  
2893 Deployments without such requirements are free to use other kinds of identifiers in their SAML  
2894 exchanges.

2895 Note also that while federated identifiers are typically used to reflect an account linking relationship  
2896 between a pair of providers, a service provider is not obligated to recognize or make use of the long term  
2897 nature of the persistent identifier or establish such a link. Such a "one-sided" identity federation is not  
2898 discernibly different and does not affect the behavior of the identity provider or any processing rules  
2899 specific to federated identifiers in the protocols defined in this specification.

### 2900 **7.3.8 Transient Identifier**

2901 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:transient

2902 Indicates that the content of the element is an identifier with transient semantics and SHOULD be treated  
2903 as an opaque and temporary value by the relying party. Transient identifier values MUST be generated  
2904 in accordance with the rules for SAML identifiers (see Section 1.2.3), and MUST NOT exceed a length of  
2905 256 characters.

2906 The `NameQualifier` and `SPNameQualifier` attributes MAY be used to signify that the identifier  
2907 represents a transient and temporary identity federation, as described in Section Federated Identifier. In  
2908 such a case, they MAY be omitted in accordance with the rules specified in that section.

## 2909 **7.4 Attribute NameFormat Identifiers**

2910 The following identifiers MAY be used in the `NameFormat` attribute defined on the  
2911 **AttributeDesignatorType** complex type (see Section x) to refer to the classification of the attribute name  
2912 for purposes of interpreting the name.

### 2913 **7.4.1 Unspecified**

2914 **URI:** urn:oasis:names:tc:SAML:2.0:attname-format:unspecified

2915 The interpretation of the attribute name is left to individual implementations.

## 2916 **7.4.2 URI Reference**

2917 **URI:** urn:oasis:names:tc:SAML:2.0:attname-format:uri

2918 The attribute name follows the convention for URI references [RFC 2396], for example as used in  
2919 XACML [XACML] attribute identifiers. The interpretation of the URI content or naming scheme is  
2920 application-specific. See the Baseline Identities and Attributes specification [SAMLBaseAtts] for a full  
2921 discussion of representing names of XACML, X.500, and LDAP attributes.

## 2922 **7.5 ~~Attribute Value Type Identifiers~~**

2923 ~~The following identifier MAY be used in the `ValueType` attribute defined on the~~  
2924 ~~**AttributeDesignatorType** complex type (see Section x) to refer to the URI-based datatype of the~~  
2925 ~~desired or supplied attribute.~~

### 2926 **7.5.1 Unspecified Value Type**

2927 **URI:** urn:oasis:names:tc:SAML:2.0:valuetype-format:unspecified

2928 ~~Indicates that the datatype of the desired or supplied attribute is application-specific. Note that any~~  
2929 ~~`ValueType` setting (default or explicit) in an attribute query, including this setting, needs to be exactly~~  
2930 ~~matched (in addition to other exact matches) in order for an attribute to be returned.~~

2931

## 8 References

2932 The following works are cited in the body of this specification.

### 8.1 Normative References

- 2934 **[Excl-C14N]** J. Boyer et al. Exclusive XML Canonicalization Version 1.0. World Wide Web  
2935 Consortium, July 2002. <http://www.w3.org/TR/xml-exc-c14n/>.
- 2936 **[Schema1]** H. S. Thompson et al. *XML Schema Part 1: Structures*. World Wide Web  
2937 Consortium Recommendation, May 2001. <http://www.w3.org/TR/xmlschema-1/>.  
2938 Note that this specification normatively references [Schema2], listed below.
- 2939 **[Schema2]** P. V. Biron et al. *XML Schema Part 2: Datatypes*. World Wide Web Consortium  
2940 Recommendation, May 2001. <http://www.w3.org/TR/xmlschema-2/>.
- 2941 **[XML]** T. Bray, et al. *Extensible Markup Language (XML) 1.0 (Second Edition)*. World  
2942 Wide Web Consortium, October 2000. <http://www.w3.org/TR/REC-xml>.
- 2943 **[XMLEnc]** D. Eastlake et al., XML Encryption Syntax and Processing,  
2944 <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>, World Wide Web  
2945 Consortium. Note that this specification normatively references [XMLEnc-XSD],  
2946 listed below.
- 2947 **[XMLEnc-XSD]** XML Encryption Schema. World Wide Web Consortium.  
2948 <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd>.
- 2949 **[XMLNS]** T. Bray et al., *Namespaces in XML*. World Wide Web Consortium, 14 January  
2950 1999. <http://www.w3.org/TR/REC-xml-names>.
- 2951 **[XMLSig]** D. Eastlake et al., *XML-Signature Syntax and Processing*, World Wide Web  
2952 Consortium, February 2002. <http://www.w3.org/TR/xmlsig-core/>. Note that this  
2953 specification normatively references [XMLSig-XSD], listed below.
- 2954 **[XMLSig-XSD]** XML Signature Schema. World Wide Web Consortium.  
2955 [http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-](http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd)  
2956 [schema.xsd](http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd).

### 8.2 Non-Normative References

- 2957
- 2958 **[LibertyProt]** J. Beatty et al., *Liberty Protocols and Schema Specification* Version 1.1, Liberty  
2959 Alliance Project, January 2003,  
2960 [http://www.projectliberty.org/specs/archive/v1\\_1/liberty-architecture-protocols-](http://www.projectliberty.org/specs/archive/v1_1/liberty-architecture-protocols-schema-v1.1.pdf)  
2961 [schema-v1.1.pdf](http://www.projectliberty.org/specs/archive/v1_1/liberty-architecture-protocols-schema-v1.1.pdf).
- 2962 **[Needham78]** R. Needham et al. *Using Encryption for Authentication in Large Networks of*  
2963 *Computers*. Communications of the ACM, Vol. 21 (12), pp. 993-999. December  
2964 1978.
- 2965 **[PGP]** Atkins, D., Stallings, W. and P. Zimmermann. *PGP Message Exchange Formats*.  
2966 IETF RFC 1991, August 1996. <http://www.ietf.org/rfc/rfc1991.txt>.
- 2967 **[PKIX]** R. Housley, W. Ford, W. Polk, D. Solo. *Internet X.509 Public Key Infrastructure*  
2968 *Certificate and CRL Profile*. IETF RFC 2459, January 1999.  
2969 <http://www.ietf.org/rfc/rfc2459.txt>.
- 2970 **[RFC 1510]** J. Kohl, C. Neuman. *The Kerberos Network Authentication Requestor (V5)*.  
2971 IETF RFC 1510, September 1993. <http://www.ietf.org/rfc/rfc1510.txt>.
- 2972 **[RFC 2119]** S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF  
2973 RFC 2119, March 1997. <http://www.ietf.org/rfc/rfc2119.txt>.

2974 [RFC 2246] T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. IETF RFC 2246, January  
2975 1999. <http://www.ietf.org/rfc/rfc2246.txt>.

2976 [RFC 2253] M. Wahl et al. *Lightweight Directory Access Protocol (v3): UTF-8 String*  
2977 *Representation of Distinguished Names*. IETF RFC 2253, December 1997.  
2978 <http://www.ietf.org/rfc/rfc2253.txt>.

2979 [RFC 2396] T. Berners-Lee et al. *Uniform Resource Identifiers (URI): Generic Syntax*. IETF  
2980 RFC 2396, August, 1998. <http://www.ietf.org/rfc/rfc2396.txt>.

2981 [RFC 2630] R. Housley. *Cryptographic Message Syntax*. IETF RFC 2630, June 1999.  
2982 <http://www.ietf.org/rfc/rfc2630.txt>.

2983 [RFC 2822] P. Resnick. *Internet Message Format*. IETF RFC 2822, April 2001.  
2984 <http://www.ietf.org/rfc/rfc2822.txt>.

2985 [RFC 2945] T. Wu. *The SRP Authentication and Key Exchange System*. IETF RFC 2945,  
2986 September 2000. <http://www.ietf.org/rfc/rfc2945.txt>.

2987 [RFC 3075] D. Eastlake, J. Reagle, D. Solo. *XML-Signature Syntax and Processing*. IETF  
2988 3075, March 2001. <http://www.ietf.org/rfc/rfc3075.txt>.

2989 [SAMLAuthnCxt] J. Kemp. *Authentication Context for the OASIS Security Assertion Markup*  
2990 *Language (SAML)*. OASIS, February 2004. Document ID sstc-saml-authn-  
2991 context-2.0. <http://www.oasis-open.org/committees/security/>.

2992 [SAMLBaseAtts] J. Hughes and P. Mishra. *Baseline Identities and Attributes for the OASIS*  
2993 *Security Assertion Markup Language (SAML)*. OASIS, March 2004. Document ID  
2994 sstc-saml-baseline-atts-2.0. <http://www.oasis-open.org/committees/security/>.

2995 [SAMLBind] E. Maler et al. *Bindings for the OASIS Security Assertion Markup Language*  
2996 *(SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-bindings-2.0.  
2997 <http://www.oasis-open.org/committees/security/>.

2998 [SAMLProf] E. Maler et al. *Profiles for the OASIS Security Assertion Markup Language*  
2999 *(SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-profiles-2.0.  
3000 <http://www.oasis-open.org/committees/security/>.

3001 [SAMLConform] E. Maler et al. *Conformance Program Specification for the OASIS Security*  
3002 *Assertion Markup Language (SAML)*. OASIS, September 2003. Document ID  
3003 oasis-sstc-saml-conform-1.1. HYPERLINK "[http://www.oasis-](http://www.oasis-open.org/committees/security/)  
3004 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)"<http://www.oasis-open.org/committees/security/>.

3005 [SAMLCore1.0] E. Maler et al. *Assertions and Protocol for the OASIS Security Assertion Markup*  
3006 *Language (SAML)*. OASIS, November 2002. [http://www.oasis-](http://www.oasis-open.org/committees/download.php/1371/oasis-sstc-saml-core-1.0.pdf)  
3007 [open.org/committees/download.php/1371/oasis-sstc-saml-core-1.0.pdf](http://www.oasis-open.org/committees/download.php/1371/oasis-sstc-saml-core-1.0.pdf).

3008 [SAMLGloss] E. Maler et al. *Glossary for the OASIS Security Assertion Markup Language*  
3009 *(SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-glossary-1.1.  
3010 HYPERLINK "[http://www.oasis-](http://www.oasis-open.org/committees/security/)  
3011 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)"[http://www.oasis-](http://www.oasis-open.org/committees/security/)  
[open.org/committees/security/](http://www.oasis-open.org/committees/security/).

3012 [SAMLPSchema] E. Maler et al. *SAML protocol schema*. OASIS, September 2003. Document ID  
3013 oasis-sstc-saml-schema-protocol-1.1. HYPERLINK "[http://www.oasis-](http://www.oasis-open.org/committees/security/)  
3014 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)"<http://www.oasis-open.org/committees/security/>.

3015 [SAMLSecure] E. Maler et al. *Security and Privacy Considerations for the OASIS Security*  
3016 *Assertion Markup Language (SAML)*. OASIS, September 2003. Document ID  
3017 oasis-sstc-saml-sec-consider-1.1. HYPERLINK "[http://www.oasis-](http://www.oasis-open.org/committees/security/)  
3018 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)"<http://www.oasis-open.org/committees/security/>.

3019 [SAMLXSD] E. Maler et al. *SAML assertion schema*. OASIS, September 2003. Document ID  
3020 oasis-sstc-saml-schema-assertion-1.1. HYPERLINK "[http://www.oasis-](http://www.oasis-open.org/committees/security/)  
3021 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)"<http://www.oasis-open.org/committees/security/>.

3022	<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. <a href="http://www.ietf.org/rfc/rfc2693.txt">http://www.ietf.org/rfc/rfc2693.txt</a> .
3023		
3024		
3025	<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> .
3026		
3027	<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> .
3028		
3029	<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> .
3030		
3031	<b>[X.500]</b>	ITU-T Recommendation X.501: Information Technology - Open Systems Interconnection - The Directory: Models. 1993.
3032		
3033	<b>[XACML]</b>	eXtensible Access Control Markup Language (XACML), product of the OASIS XACML TC. <a href="http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml">http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml</a> .
3034		
3035		
3036	<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> .
3037		
3038		

---

3039 **Appendix A. Acknowledgments**

3040 The editors would like to acknowledge the contributions of the OASIS Security Services Technical  
3041 Committee, whose voting members at the time of publication were:

- 3042 • @@

## Appendix B. Revision History

Rev	Date	By Whom	What
01	20 Oct 2003	Eve Maler	Initial draft. Converted to OpenOffice. <b>CORE-1</b> through <b>CORE-4</b> . Namespaces and schema snippets updated. Non-normative material in Chapter 1 removed.  <a href="http://www.oasis-open.org/committees/download.php/3936/sstc-saml-core-2.0-draft-01.pdf">http://www.oasis-open.org/committees/download.php/3936/sstc-saml-core-2.0-draft-01.pdf</a>
02	4 Jan 2004	Eve Maler	Implemented Scott Cantor's draft-sstc-nameid-07 solution proposal ( <a href="http://www.oasis-open.org/apps/org/workgroup/security/download.php/4587">http://www.oasis-open.org/apps/org/workgroup/security/download.php/4587</a> ) for work item <b>W-2</b> , Identity Federation. Some issues remain (substitution group usage; usage of derivation by restriction; the whole protocol piece hasn't been designed yet).  Fixed <b>CORE-10</b> (the description of subelement occurrence in the <Evidence> element).  <a href="http://www.oasis-open.org/committees/download.php/4866/sstc-saml-core-2.0-draft-02-diff.pdf">http://www.oasis-open.org/committees/download.php/4866/sstc-saml-core-2.0-draft-02-diff.pdf</a>
03	24 Jan 2004	Scott Cantor	Name identifier, issuer, and federation protocol additions/changes. See 03-interim-diff draft for intermediate set of change bars.  <a href="http://www.oasis-open.org/committees/download.php/5181/sstc-saml-core-2.0-draft-03-interim-diff.pdf">http://www.oasis-open.org/committees/download.php/5181/sstc-saml-core-2.0-draft-03-interim-diff.pdf</a> <a href="http://www.oasis-open.org/committees/download.php/5180/sstc-saml-core-2.0-draft-03-diff.pdf">http://www.oasis-open.org/committees/download.php/5180/sstc-saml-core-2.0-draft-03-diff.pdf</a>
04	1 Feb 2004	Eve Maler	Made minor edits to new and existing material; changed new <AssertionRequest> element name to <AssertionIDRequest>; changed new <AssertionArtifact> and <NewIdentifier> element declarations from local to global; made distinction between normative and non-normative references; implemented the blocking of element substitution. The bulk of work item <b>W-2</b> , Identity Federation, is now reflected here. What remains is the federation termination protocol, plus a few other pieces that are covered under other work items.  <a href="http://www.oasis-open.org/committees/download.php/5232/sstc-saml-core-2.0-draft-04-diff.pdf">http://www.oasis-open.org/committees/download.php/5232/sstc-saml-core-2.0-draft-04-diff.pdf</a>
05	17 Feb 2004	Scott Cantor, John Kemp, Eve Maler	Added FedTerm protocol ( <b>W-2</b> ), removed NameID date attributes, clarified Name Reg processing rules, added Extensions facility and Consent attribute. Also moved Signature on assertions to a location consistent with Request and Response. Added session protocol material ( <b>W-1</b> ); still unfinished.  <a href="http://www.oasis-open.org/committees/download.php/5519/sstc-saml-core-2.0-draft-05-diff.pdf">http://www.oasis-open.org/committees/download.php/5519/sstc-saml-core-2.0-draft-05-diff.pdf</a>
06	20 Feb 2004	Scott Cantor, John Kemp, Eve Maler	Added AssertionURIReference ( <b>W-19</b> ), a proposal for ProxyRestrictionCondition, and a proposal for AuthNRequest/Response (related to many work items). Fleshed out LogoutRequest/Response ( <b>W-1</b> ). Implemented the freezing of authZ decision statement functionality ( <b>W-28b</b> ).  <a href="http://www.oasis-open.org/committees/download.php/5600/sstc-saml-core-2.0-draft-06-diff.pdf">http://www.oasis-open.org/committees/download.php/5600/sstc-saml-core-2.0-draft-06-diff.pdf</a>

Rev	Date	By Whom	What
07	7 Mar 2004	Scott Cantor, Eve Maler	<p>Implemented new arrangement for subject information and decision on KeyInfo description, as agreed at 2 Mar 2004 telecon.</p> <p>Adjusted normative language around subject "matching" rules based on subject changes.</p> <p>Revised AuthnRequest proposal based on those changes and feedback from list and focus calls.</p> <p>Incorporated additional schema and processing rules related to ECP and proxying use cases from ID-FF.</p> <p>Added AuthnContext to AuthenticationStatement.</p> <p>Added NameIdentifierMapping protocol (<b>W-2</b>).</p>
<a href="http://www.oasis-open.org/committees/download.php/5790/sstc-saml-core-2.0-draft-07-diff.pdf">http://www.oasis-open.org/committees/download.php/5790/sstc-saml-core-2.0-draft-07-diff.pdf</a>			
08	15 Mar 2004	Scott Cantor, Eve Maler	<p>Added ArtifactRequest/Response pair as a new protocol.</p> <p>Implemented proposed W-28a attribute changes (rev 03 of the proposal, reflecting focus group input).</p>
<a href="http://www.oasis-open.org/committees/download.php/5951/sstc-saml-core-2.0-draft-08-diff.pdf">http://www.oasis-open.org/committees/download.php/5951/sstc-saml-core-2.0-draft-08-diff.pdf</a>			
09	8 Apr 2004	Eve Maler	<p>Minor cleanup, plus decisions from March-April 2004 F2F meeting: Moved Signature element up in Assertion contents. Clarified that DoNotCacheCondition has one-time-use semantics. Made NameFormat on the Attribute element clearly optional. Changed the default ValueType identifier name. Added the ability to put arbitrary attributes on the AttributeDesignator element. Removed Source on the Attribute element. Changed the content of Extensions in the Request element to ##other. Removed the restriction saying only federated identifiers could be replaced and set with the termination protocol. Changed Reason on the LogoutRequest element to be a URI reference. Made SessionIndex in the LogoutRequest element globally declared. Added bibliographic references to the new SAML specs.</p>
<a href="http://www.oasis-open.org/committees/download.php/6323/sstc-saml-core-2.0-draft-09-diff.pdf">http://www.oasis-open.org/committees/download.php/6323/sstc-saml-core-2.0-draft-09-diff.pdf</a>			
10	12 Apr 2004	Scott Cantor, Eve Maler	<p>Allowed assertions to be subjectless. Allowed Audience to reference a specific provider URI. Changed AuthnMethod and AuthnContext handling. Removed RelayState. Added AllowCreate on NameIDPolicy. Consolidated two protocols into the name identifier management protocol. Added a name identifier URI for Kerberos principals.</p>
<a href="http://www.oasis-open.org/committees/download.php/6347/sstc-saml-core-2.0-draft-10-diff.pdf">http://www.oasis-open.org/committees/download.php/6347/sstc-saml-core-2.0-draft-10-diff.pdf</a>			
<a href="#">11</a>	<a href="#">11 May 2004</a>	<a href="#">Eve Maler</a>	<p><a href="#">Updated the wording describing the permissible combinations of assertion vs. protocol versions (issue TECH-2). Removed the proposed ValueType field on AttributeDesignator: how to do this will be described in the Baseline Attributes spec instead.</a></p>

3044

3045

---

## Appendix C. Notices

3046 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that  
3047 might be claimed to pertain to the implementation or use of the technology described in this document or  
3048 the extent to which any license under such rights might or might not be available; neither does it  
3049 represent that it has made any effort to identify any such rights. Information on OASIS's procedures with  
3050 respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights  
3051 made available for publication and any assurances of licenses to be made available, or the result of an  
3052 attempt made to obtain a general license or permission for the use of such proprietary rights by  
3053 implementors or users of this specification, can be obtained from the OASIS Executive Director.

3054 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications,  
3055 or other proprietary rights which may cover technology that may be required to implement this  
3056 specification. Please address the information to the OASIS Executive Director.

3057 **Copyright © OASIS Open 2004. All Rights Reserved.**

3058 This document and translations of it may be copied and furnished to others, and derivative works that  
3059 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published  
3060 and distributed, in whole or in part, without restriction of any kind, provided that the above copyright  
3061 notice and this paragraph are included on all such copies and derivative works. However, this document  
3062 itself may not be modified in any way, such as by removing the copyright notice or references to OASIS,  
3063 except as needed for the purpose of developing OASIS specifications, in which case the procedures for  
3064 copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required  
3065 to translate it into languages other than English.

3066 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors  
3067 or assigns.

3068 This document and the information contained herein is provided on an "AS IS" basis and OASIS  
3069 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY  
3070 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS  
3071 OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR  
3072 PURPOSE.