



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39

Liberty Protocols and Schemas Specification

Version 1.0

11 July 2002

Document Description: liberty-architecture-protocols-schemas-v1.0

39 **Notice**

40

41 Copyright © 2002 ActivCard; American Express Travel Related Services; America Online, Inc.;
42 Bank of America; Bell Canada; Catavault; Cingular Wireless; Cisco Systems, Inc.; Citigroup;
43 Cyberun Corporation; Deloitte & Touche LLP; EarthLink, Inc.; Electronic Data Systems, Inc.;
44 Entrust, Inc.; Ericsson; Fidelity Investments; France Telecom; Gemplus; General Motors; Hewlett-
45 Packard Company; i2 Technologies, Inc.; Intuit Inc.; MasterCard International; Nextel
46 Communications; Nippon Telegraph and Telephone Company; Nokia Corporation; Novell, Inc.;
47 NTT DoCoMo, Inc.; OneName Corporation; Openwave Systems Inc.; PricewaterhouseCoopers LLP;
48 Register.com; RSA Security Inc; Sabre Holdings Corporation; SAP AG; SchlumbergerSema; Sony
49 Corporation; Sun Microsystems, Inc.; United Airlines; VeriSign, Inc.; Visa International; Vodafone
50 Group Plc; Wave Systems. All rights reserved.

51

52 This Specification has been prepared by Sponsors of the Liberty Alliance. Permission is hereby
53 granted to use the Specification solely for the purpose of implementing the Specification. No rights
54 are granted to prepare derivative works of this Specification. Entities seeking permission to
55 reproduce portions of this document for other uses must contact the Liberty Alliance to determine
56 whether an appropriate license for such use is available.

57

58 Implementation of this Specification may involve the use of one or more of the following United
59 States Patents claimed by AOL Time Warner, Inc.: No.5,774,670, No.6,134,592, No.5,826,242, No.
60 5,825,890, and No.5,671,279. The Sponsors of the Specification take no position concerning the
61 evidence, validity or scope of the claimed subject matter of the aforementioned patents.
62 Implementation of certain elements of this Specification may also require licenses under third party
63 intellectual property rights other than those identified above, including without limitation, patent
64 rights. The Sponsors of the Specification are not and shall not be held responsible in any manner for
65 identifying or failing to identify any or all such intellectual property rights that may be involved in
66 the implementation of the Specification.

67

68 **This Specification is provided "AS IS", and no participant in the Liberty Alliance makes any**
69 **warranty of any kind, express or implied, including any implied warranties of merchantability,**
70 **non-infringement or third party intellectual property rights, and fitness for a particular**
71 **purpose.**

72

73 Liberty Alliance Project
74 Licensing Administrator
75 c/o IEEE-ISTO
76 445 Hoes Lane, P.O. Box 1331
77 Piscataway, NJ 08855-1331, USA

78

78 **Editor**

79 John D. Beatty, Sun Microsystems, Inc.

80 **Contributors**

81
82 The following Liberty Alliance Project Sponsor companies contributed to the development of
83 this specification:
84

- | | |
|--|--|
| ActivCard | MasterCard International |
| American Express Travel Related Services | Nextel Communications |
| America Online, Inc. | Nippon Telegraph and Telephone Company |
| Bank of America | Nokia Corporation |
| Bell Canada | Novell, Inc. |
| Catavault | NTT DoCoMo, Inc. |
| Cingular Wireless | OneName Corporation |
| Cisco Systems, Inc. | Openwave Systems Inc. |
| Citigroup | PricewaterhouseCoopers LLP |
| Cyberun Corporation | Register.com |
| Deloitte & Touche LLP | RSA Security Inc |
| EarthLink, Inc. | Sabre Holdings Corporation |
| Electronic Data Systems, Inc. | SAP AG |
| Entrust, Inc. | SchlumbergerSema |
| Ericsson | Sony Corporation |
| Fidelity Investments | Sun Microsystems, Inc. |
| France Telecom | United Airlines |
| Gemplus | VeriSign, Inc. |
| General Motors | Visa International |
| Hewlett-Packard Company | Vodafone Group Plc |
| i2 Technologies, Inc. | Wave Systems |
| Intuit Inc. | |

85

86

87

88

88 **Table of Contents**

89	1	Introduction	5
90	1.1	Notation.....	5
91	1.2	Overview.....	6
92	2	Schema Declarations	6
93	2.1	Schema Header and Namespace Declarations.....	6
94	2.2	Type and Element Declarations	6
95	3	Protocols	7
96	3.1	General Requirements.....	7
97	3.1.1	XML Signature.....	7
98	3.1.2	Protocol and Assertion Versioning	7
99	3.1.3	Provider ID Uniqueness	7
100	3.1.4	Name Identifier Construction.....	7
101	3.1.5	Security.....	8
102	3.2	Single Sign-On and Federation Protocol	8
103	3.2.1	Request.....	8
104	3.2.2	Response.....	11
105	3.2.3	Processing Rules	13
106	3.2.4	Request Envelope.....	14
107	3.2.5	Response Envelope	16
108	3.3	Name Registration Protocol.....	17
109	3.3.1	Request.....	17
110	3.3.2	Response.....	18
111	3.3.3	Processing Rules	19
112	3.4	Federation Termination Notification Protocol.....	19
113	3.4.1	Message.....	19
114	3.4.2	Processing Rules	20
115	3.5	Single Logout Protocol	20
116	3.5.1	Message.....	20
117	3.5.2	Processing Rules	21
118	4	Provider Metadata Schema.....	22
119	4.1	Generic Provider Descriptor	22
120	4.2	Service Provider Descriptor.....	22
121	4.2.1	Example.....	23
122	4.3	Identity Provider Descriptor	23
123	4.3.1	Example.....	24
124	5	Schema Definition.....	24
125	6	References	27
126			
127			

127 1 Introduction

128 This specification defines the abstract Liberty protocols for identity federation, single sign-on, name
129 registration, federation termination, and single logout. Several concrete bindings and profiles of these
130 protocols are defined in [[LibertyBindProf](#)].

131 1.1 Notation

132 This specification uses schema documents conforming to W3C XML Schema (see [[Schema1](#)]) and
133 normative text to describe the syntax and semantics of XML-encoded SAML assertions and protocol
134 messages. Note: Phrases and numbers in brackets [] refer to other documents; details of these
135 references can be found in Section 6 (at the end of this document).

136 The key words “MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,”
137 “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” in this specification are to be
138 interpreted as described in [[RFC2119](#)]: “they MUST only be used where it is actually required for
139 interoperation or to limit behavior which has potential for causing harm (e.g., limiting
140 retransmissions).”

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

145 Listings of schemas appear like this.

146
147 Listings of instance fragments appear like this.

148
149 The following namespaces are referred to in this document:

- 150 • The prefix `lib:` stands for the Liberty namespace
151 (`http://schemas.projectliberty.org/schemas/core/2002/05/`). This
152 namespace is the default for instance fragments, type names, and element names in this
153 document.
- 154 • The prefix `saml:` stands for the SAML assertion namespace
155 (`urn:oasis:names:tc:SAML:1.0:assertion`).
- 156 • The prefix `samlp:` stands for the SAML protocol namespace
157 (`urn:oasis:names:tc:SAML:1.0:protocol`).
- 158 • The prefix `ds:` stands for the W3C XML signature namespace
159 (`http://www.w3.org/2000/09/xmldsig#`).
- 160 • The prefix `xsd:` stands for the W3C XML schema namespace
161 (`http://www.w3.org/2001/XMLSchema`). In schema listings, this is the default
162 namespace and no prefix is shown.
- 163 • The prefix `xsi:` stands for the W3C XML schema instance namespace
164 (`http://www.w3.org/2001/XMLSchema-instance`).

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

167 For readability, when an XML Schema type is specified to be `xsd:boolean`, this document
168 discusses the values as “true” and “false” rather than the “1” and “0” which will exist in the
169 document instance.

170 Definitions for Liberty-specific terms can be found in [[LibertyGloss](#)].

171 1.2 Overview

172 This specification defines a set of protocols that collectively provide a solution for identity federation
173 management, cross-domain authentication, and session management. This specification also defines
174 provider metadata schemas that may be used for making a priori arrangements between providers.

175 The Liberty architecture contains three actors: Principal, identity provider, and service provider. A
176 Principal is an entity (for example, an end user) that has an identity provided by an identity provider.
177 A service provider provides services to the Principal.

178 Once the Principal is *authenticated* to the identity provider, the identity provider can provide an
179 authentication assertion to the Principal, who can present the assertion to the service provider. The
180 Principal is then also authenticated to the service provider if the service provider trusts the assertion.
181 An *identity federation* is said to exist between an identity provider and a service provider when the
182 service provider accepts authentication assertions regarding a particular Principal from the identity
183 provider. This specification defines a protocol where the identity of the Principal can be *federated*
184 between the identity provider and the service provider.

185 This specification relies on the SAML specification in [[SAMLCore](#)]. In SAML terminology, an
186 identity provider acts as an Asserting Party and an Authentication Authority, while a service provider
187 acts as a Relying Party.

188 2 Schema Declarations

189 This document specifies an XML schema for Liberty. The schema header along with namespace,
190 type, and element declarations are in 2.1 and 2.2.

191 2.1 Schema Header and Namespace Declarations

192 The following schema fragment defines the XML namespaces and other header information for the
193 Liberty schema:

```
194 <schema targetNamespace="http://www.projectliberty.org/schemas/core/2002/05"  
195 xmlns:lib="http://www.projectliberty.org/schemas/core/2002/05"  
196 xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"  
197 xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"  
198 xmlns:ac="http://www.projectliberty.org/schemas/authctx/2002/05"  
199 xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"  
200 attributeFormDefault="unqualified">  
201   <import namespace="urn:oasis:names:tc:SAML:1.0:assertion" schemaLocation="http://www.oasis-  
202   open.org/committees/security/docs/draft-sstc-schema-assertion-31.xsd"/>  
203   <import namespace="urn:oasis:names:tc:SAML:1.0:protocol" schemaLocation="http://www.oasis-  
204   open.org/committees/security/docs/draft-sstc-schema-protocol-31.xsd"/>  
205   <import namespace="http://www.w3.org/2000/09/xmldsig#"   
206   schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd"/>  
207   <import namespace="http://www.projectliberty.org/schemas/authctx/2002/05" schemaLocation="draft-  
208   liberty-architecture-authentication-context-06.xsd"/>
```

209 2.2 Type and Element Declarations

210 Declarations for types and elements that are subsequently referred to in this document are as follows:

```
211 <element name="ProviderID" type="anyURI"/>
```

212 3 Protocols

213 The Liberty protocol suite consists of the following protocols:

- 214 • Single Sign-On and Federation: The protocol by which identities are federated and by which
215 single sign-on occurs.
- 216 • Name Registration: The protocol by which a service provider can register an alternative
217 opaque handle (or *name identifier*) for a Principal.
- 218 • Federation Termination Notification: The protocol by which a provider can notify another
219 provider than a particular identity federation has been terminated (also known as
220 defederation).
- 221 • Single Logout: The protocol by which providers notify each other of logout events for single
222 logout functionality.

223 3.1 General Requirements

224 A set of general requirements applicable to all protocols is found in 3.1.1 through 3.1.5.

225 3.1.1 XML Signature

226 The XML signature specification calls out a general XML syntax for signing data with many
227 flexibilities and choices. All signed XML entities MUST adhere to the “XML Signature Profile”
228 constraints defined in [[SAMLCore](#)].

229 3.1.2 Protocol and Assertion Versioning

230 Version information appears in protocol messages and assertions defined in this specification. This
231 specification defines version 1.0 for the protocol messages and assertions. Version numbering of
232 assertions is independent of the version numbering of the protocol messages.

233 This specification follows the version numbering requirements, processing rules, and error
234 conditions specified in “SAML Versioning” in [[SAMLCore](#)].

235 3.1.3 Provider ID Uniqueness

236 All providers have a URI-based identifier. The provider’s URI-based identifier MUST be unique
237 within the scope of all providers with which it communicates. It is RECOMMENDED that a
238 provider use a URL with its own domain name for this identifier. The URI-based identifier MUST
239 NOT exceed 1024 characters.

240 Some profiles of the protocols contained in this specification may require a succinct 20-byte
241 identifier. In this case, a provider’s succinct identifier also MUST be unique within the scope of all
242 providers with which it communicates. It is RECOMMENDED that a provider derive its succinct
243 identifier by generating the SHA-1 hash of its URI-based identifier.

244 3.1.4 Name Identifier Construction

245 Principals are assigned name identifiers by identity providers and potentially by service providers.
246 When generated by the identity provider, a name identifier MUST be constructed using pseudo-
247 random values that have no discernable correspondence with the Principal’s identifier (e.g.,
248 username) at the identity provider. The intent is to create a nonpublic pseudonym so as to contravene

249 the linkability of the Principal's identity or activities. Service providers SHOULD follow the same
250 construction rules. Name identifier values MUST NOT exceed 256 characters.

251 **3.1.5 Signature Verification**

252 Processing rules for the protocols defined in this document commonly specify digital signature
253 verification. In these cases, it is not sufficient to only verify the signature of the signed object. That is,
254 verifying the `<ds:Signature>` element MUST be performed in accordance with the best practices
255 for the certification path technology in use. For example, when using X.509 v3 public key
256 certificates it is strongly RECOMMENDED that certification path validation be performed in
257 accordance to the PKIX Profile as specified in [\[RFC3280\]](#).

258 **3.1.6 Security**

259 Because this specification defines only abstract protocols and does not define specific protocol
260 profiles or the environment in which protocols will be deployed, most security requirements are
261 deferred to individual profiles. See [\[LibertyBindProf\]](#) for security considerations for the Liberty-
262 defined bindings and profiles. When a general security requirement can be stated for one of the
263 abstract protocols described in this specification, the requirement is stated in line with the specific
264 protocol.

265 **3.2 Single Sign-On and Federation Protocol**

266 The Single Sign-On and Federation Protocol defines a request and response protocol by which single
267 sign-on and identity federation occurs. The protocol works as follows:

- 268 1. A service provider issues an `<AuthnRequest>` request to an identity provider, instructing
269 the identity provider to provide an authentication assertion to the service provider.
270 Optionally, the service provider MAY request that the identity be federated.
- 271 2. The identity provider responds with either an `<AuthnResponse>` containing authentication
272 assertions to the service provider or an artifact that can be dereferenced into an authentication
273 assertion. Additionally, the identity provider potentially federates the Principal's identity at
274 the identity provider with the Principal's identity at the service provider.

275 The resulting authentication statement in the assertion by the identity provider MAY contain an
276 `ReauthenticateOnOrAfter` attribute. If this attribute is included, the service provider MUST
277 send a new `<lib:AuthnRequest>` for the Principal to the identity provider at the next point of
278 interaction with the Principal on or after the time specified by the `ReauthenticateOnOrAfter`
279 attribute. It is then up to the identity provider to authenticate the user. Note: The Principal may
280 already have an authenticated session with the identity provider, in which case a new authentication
281 assertion would be generated by the identity provider without any intervention by the Principal.

282 **3.2.1 Request**

283 The service provider issues an `<AuthnRequest>` request to the identity provider. A set of
284 parameters is included in the request that allows the service provider to specify desired behavior at
285 the identity provider in processing the request. The service provider can control the following
286 identity provider behaviors:

- 287 • Prompt the Principal for credentials if the Principal is not presently authenticated.
- 288 • Prompt the Principal for credentials, even if the Principal is presently authenticated.

- 289 • Federate the Principal’s identity at the identity provider with the Principal’s identity at the
290 service provider.
- 291 • Use a specific protocol profile in responding to the request.
- 292 • Use a specific authentication context (for example, smartcard-based authentication vs.
293 username/password-based authentication).

294 Additionally, the service provider MAY include any desired state information in the request that the
295 identity provider should relay back to the service provider in the response.

296 **3.2.1.1 Element <AuthnRequest>**

297 The <AuthnRequest> request is defined as an extension of **samlp:RequestAbstractType**. The
298 RequestID attribute in **samlp:RequestAbstractType** has uniqueness requirements placed on it by
299 [\[SAMLCore\]](#), which require it to have the properties of a nonce.

300 The elements of the request are as follows:

301 ProviderID [Required]

302 The service provider’s URI-based identifier.

303 IsPassive [Optional]

304 If “true,” specifies that the identity provider MUST NOT interact with the Principal and
305 MUST NOT take control of the user interface from the service provider. If “false,” the
306 identity provider MAY interact with the user and MAY temporarily take control of the user
307 interface for that purpose. If not specified, “true” is presumed.

308 ForceAuthn [Optional]

309 Controls whether the identity provider authenticates the Principal regardless of whether the
310 Principal is already authenticated. This element is specified only when <IsPassive> is
311 “false.” If <ForceAuthn> is “true,” specifies that the identity provider MUST always
312 authenticate the Principal, regardless of whether the Principal is presently authenticated. If
313 “false,” specifies that the identity provider MUST reauthenticate the user only if the Principal
314 is not presently authenticated. If not specified, “false” is presumed.

315 Federate [Optional]

316 Specifies that the service provider wishes to federate the Principal’s identity at the service
317 provider with the Principal’s identity at the identity provider. If the element is not specified, it
318 is presumed that the service provider does not wish to federate the identity.

319 ProtocolProfile [Optional]

320 The protocol profile that the service provider wishes to use for the response. If the element is
321 not specified, the default protocol profile is
322 <http://projectliberty.org/profiles/brws-art> defined in [\[LibertyBindProf\]](#).

323 AuthnContext [Optional]

324 Information regarding which authentication context the service provider desires the identity
325 provider to use in authenticating the Principal.

326 RelayState [Optional]

327 State information that will be relayed back in the response. This data SHOULD be integrity-
328 protected by the request author and MAY have other protections place on it by the request
329 author. An example of such protection is confidentiality.

330 The <AuthnContext> element has the following mutually exclusive elements:

331 AuthnContextClassRef [Optional]

332 The ordered set of authentication context class references the service provider desires the
333 identity provider to use in authenticating the Principal.

334 AuthnContextMinimumClassRef [Optional]

335 The minimum authentication context class the service provider will accept.

336 AuthnContextStatementRef [Optional]

337 The ordered set of exact authentication statements the service provider desires the identity
338 provider to use in authenticating the Principal.

339 The schema fragment defining the element and its type is as follows:

```
340 <element name="AuthnRequest" type="lib:AuthnRequestType"/>  
341 <complexType name="AuthnRequestType">  
342   <complexContent>  
343     <extension base="samlp:RequestAbstractType">  
344       <sequence>  
345         <element ref="lib:ProviderID"/>  
346         <element name="ForceAuthn" type="boolean" minOccurs="0"/>  
347         <element name="IsPassive" type="boolean" minOccurs="0"/>  
348         <element name="Federate" type="boolean" minOccurs="0"/>  
349         <element ref="lib:ProtocolProfile" minOccurs="0"/>  
350         <element ref="lib:AuthnContext" minOccurs="0"/>  
351         <element ref="lib:RelayState" minOccurs="0"/>  
352       </sequence>  
353     </extension>  
354   </complexContent>  
355 </complexType>  
356 <element name="RelayState"/>  
357 <element name="ProtocolProfile" type="anyURI"/>  
358 <element name="AuthnContext">  
359   <complexType>  
360     <choice>  
361       <element name="AuthnContextClassRef" type="anyURI" maxOccurs="unbounded"/>  
362       <element name="AuthnContextStatementRef" type="anyURI" maxOccurs="unbounded"/>  
363       <element name="AuthnContextMinimumClassRef" type="anyURI"/>  
364     </choice>  
365   </complexType>  
366 </element>
```

367 3.2.1.2 Example

```
368 <AuthnRequest RequestID="4e7c3772-4fa4-4a0f-99e8-7d719ff6067c" MajorVersion="1" MinorVersion="0"  
369 IssueInstant="2001-12-17T09:30:47-05:00">  
370   <ds:Signature> ... </ds:Signature>  
371   <ProviderID>http://ServiceProvider.com</ProviderID>  
372   <ForceAuthn>1</ForceAuthn>  
373   <IsPassive>0</IsPassive>  
374   <Federate>1</Federate>  
375   <ProtocolProfile>http://projectliberty.org/profiles/brws-post</ProtocolProfile>  
376   <AuthnContext>  
377     <AuthnContextClassRef>  
378       "http://projectliberty.org/schemas/authctx/2002/05/Password"  
379     </AuthnContextClassRef>  
380   </AuthnContext>  
381   <RelayState>R0lGODlhcgGSALMAAAQCAEMmCZtuMFQxDS8b</RelayState>  
382 </AuthnRequest>
```

383 3.2.2 Response

384 The response is a <AuthnResponse> element containing either a set of authentication assertions or
385 a set of artifacts the service provider can dereference into a set of authentication assertions.

386 All authentication assertions generated by an identity provider for a service provider MUST be of
387 type **AssertionType**. The <saml:Subject> element in any subject statement MUST be of type
388 **SubjectType**. If the service provider registered a name identifier for the Principal (see 3.3), the
389 <saml:NameIdentifier> element in the <saml:Subject> element MUST be the service
390 provider-provided name identifier for the Principal. Otherwise, <saml:NameIdentifier> MUST
391 be the name identifier provided by the identity provider at the time of federation. The
392 <IDPProvidedNameIdentifier> MUST contain the name identifier provided by the identity
393 provider at the time of federation.

394 All authentication statements MUST be of type **AuthenticationStatementType**.

395 Identity providers MUST include a <saml:AudienceRestrictionCondition> element that
396 specifies the intended consumers of the assertion.

397 Identity providers MAY include a *SessionIndex* attribute in resulting authentication statements,
398 which is used to aid the identity provider in managing multiple sessions with the Principal. If the
399 identity provider includes this *SessionIndex* attribute, subsequent messages from the service
400 provider to the identity provider that are session-dependent MUST include this *SessionIndex*
401 attribute.

402 All assertions in the <AuthnResponse> MUST be signed by the identity provider.

403 3.2.2.1 Element <AuthnResponse>

404 The type **AuthnResponseType**, extended from **samlp:ResponseType**, provides for the
405 <RelayState> element. The schema fragment is as follows:

```
406 <element name="AuthnResponse" type="lib:AuthnResponseType"/>  
407 <complexType name="AuthnResponseType">  
408 <complexContent>  
409 <extension base="samlp:ResponseType">  
410 <sequence>  
411 <element ref="lib:RelayState" minOccurs="0"/>  
412 </sequence>  
413 </extension>  
414 </complexContent>  
415 </complexType>
```

416 3.2.2.2 Element <Assertion>

417 Assertions provided in an <AuthnResponse> element MUST be of type **AssertionType**, which is
418 an extension of **saml:AssertionType**, so that the RequestID attribute from the original
419 <AuthnRequest> is included in the InResponseTo attribute in the <Assertion> element. This
420 step is done because the <AuthnResponse> element is not required to be signed, whereas the
421 contained <Assertion> elements are required to be signed. The schema fragment is as follows:

```
422 <element name="Assertion" type="lib:AssertionType"/>  
423 <complexType name="AssertionType">  
424 <complexContent>  
425 <extension base="saml:AssertionType">  
426 <attribute name="InResponseTo" type="saml:IDReferenceType"/>  
427 </extension>  
428 </complexContent>  
429 </complexType>
```

430 3.2.2.3 Type SubjectType

431 The type **SubjectType**, extended from **saml:SubjectType**, is used to include the
432 <IDPProvidedNameIdentifier> element in subject statements. The schema fragment is as
433 follows:

```
434 <complexType name="SubjectType">  
435 <complexContent>  
436 <extension base="saml:SubjectType">  
437 <sequence>  
438 <element ref="lib:IDPProvidedNameIdentifier"/>  
439 </sequence>  
440 </extension>  
441 </complexContent>  
442 </complexType>
```

443 3.2.2.4 Type AuthenticationStatementType

444 The type **AuthenticationStatementType** is an extension of **saml:AuthenticationStatementType**,
445 which allows for the following elements and attributes:

446 AuthnContext [Optional]

447 The authentication context that the identity provider used in the authentication event that
448 yielded this authentication statement. Contains either an authentication context statement or a
449 reference to an authentication context statement. Optionally contains a reference to an
450 authentication context class.

451 ReauthenticateOnOrAfter [Optional]

452 The time at which the service provider reauthenticates the Principal with the identity
453 provider.

454 SessionIndex [Optional]

455 The index into the particular session between the Principal and the identity provider under
456 which this authentication statement is being issued.

457 The schema fragment is as follows:

```
458 <complexType name="AuthenticationStatementType">  
459 <complexContent>  
460 <extension base="saml:AuthenticationStatementType">  
461 <sequence>  
462 <element name="AuthnContext" minOccurs="0">  
463 <complexType>  
464 <sequence>  
465 <element name="AuthnContextClassRef" type="anyURI" minOccurs="0"/>  
466 <choice>  
467 <element ref="ac:AuthenticationContextStatement"/>  
468 <element name="AuthnContextStatementRef" type="anyURI"/>  
469 </choice>  
470 </sequence>  
471 </complexType>  
472 </element>  
473 </sequence>  
474 <attribute name="ReauthenticateOnOrAfter" type="dateTime" use="optional"/>  
475 <attribute name="SessionIndex" type="string" use="optional"/>  
476 </extension>  
477 </complexContent>  
478 </complexType>
```

479 3.2.2.5 Example

```
480 <AuthnResponse ResponseID="a10072f3-4dae-479d-9357-03b95327fd77" InResponseTo="4e7c3772-4fa4-4a0f-  
481 99e8-7d719ff6067c" MajorVersion="1" MinorVersion="0" IssueInstant="2001-12-17T09:30:47-05:00">  
482 <samlp:Status>  
483 <samlp:StatusCode Value="saml:Success"/>  
484 </samlp:Status>
```

```

485 <saml:Assertion MajorVersion="1" MinorVersion="0" AssertionID="e06e5a28-bc80-4ba6-9ecb-
486 712949db686e" Issuer="http://IdentityProvider.com" IssueInstant="2001-12-17T09:30:47-05:00"
487 InResponseTo="4e7c3772-4fa4-4a0f-99e8-7d719ff6067c" xsi:type="AssertionType">
488   <saml:Conditions NotBefore="2001-12-17T09:30:47-05:00" NotOnOrAfter="2001-12-17T09:35:47-
489 05:00">
490     <saml:AudienceRestrictionCondition>
491       <saml:Audience>http://ServiceProvider.com</saml:Audience>
492     </saml:AudienceRestrictionCondition>
493   </saml:Conditions>
494   <saml:AuthenticationStatement AuthenticationInstant="2001-12-17T09:30:47-05:00"
495 SessionIndex="3" ReauthenticateOnOrAfter="2001-12-17T11:30:47-05:00"
496 xsi:type="AuthenticationStatementType">
497     <saml:Subject xsi:type="SubjectType">
498       <saml:NameIdentifier>342ad3d8-93ee-4c68-be35-cc9e7db39e2b</saml:NameIdentifier>
499       <IDPProvidedNameIdentifier>342ad3d8-93ee-4c68-be35-
500 cc9e7db39e2b</IDPProvidedNameIdentifier>
501     </saml:Subject>
502   </saml:AuthenticationStatement>
503   <ds:Signature>...</ds:Signature>
504 </saml:Assertion>
505 <RelayState>R01GODlhcgSALMAAAQCAEMmCZtuMFQxDS8b</RelayState>
506 </AuthnResponse>

```

507 3.2.3 Processing Rules

508 When an identity provider receives an authentication request, it MUST process the request according
509 to the following rules:

- 510 • The <ProviderID> MUST be the Provider ID of a known service provider with which the
511 identity provider has established a relationship. The <ProviderID> MUST be resolvable
512 to an assertion consumer service URL at the service provider that the identity provider may
513 use when returning the corresponding assertion reference.
- 514 • <ds:Signature>, if present, MUST be the signature of the service provider as specified by
515 the <ProviderID>.
- 516 • If <IsPassive> is “false,” the identity provider MUST NOT interact with the Principal and
517 MUST NOT take control of the user interface (if applicable).
- 518 • The identity provider MUST attempt to authenticate the Principal if <ForceAuthn> is
519 “true,” regardless of whether the Principal is presently authenticated, unless <IsPassive>
520 is “true.”
- 521 • The identity provider MUST attempt to authenticate the Principal if <ForceAuthn> is
522 “false” and the Principal is not presently authenticated, unless <IsPassive> is “true.”
- 523 • The identity provider MAY federate the Principal’s identity at the service provider with the
524 user’s identity at the identity provider if <Federate> is “true” and the Principal has
525 consented for such an action to occur. The identity provider MUST NOT federate if
526 <Federate> is “false.”
- 527 • The identity provider MUST respond using the specified <ProtocolProfile>.
- 528 • If <RelayState> contains a value, the identity provider MUST include this value in
529 unmodified form in the <RelayState> element of the returned authentication assertion.
- 530 • The InResponseTo attribute in all generated <Assertion> elements in the
531 <AuthnResponse> element MUST be set to the value of the RequestID attribute in the
532 corresponding <AuthnRequest> element.

533 Additionally, if the <AuthnContext> element is specified, the identity provider MUST authenticate
534 the Principal according to the following rules:

- 535 • If one or more `<AuthnContextClassRef>` elements are included, then the resulting
536 authentication statement in the assertion (if any) MUST contain an authentication statement
537 that conforms to one of the specified classes. Additionally, the set of
538 `<AuthnContextClassRef>` elements MUST be evaluated as an ordered set, where the
539 first element is the most preferred authentication context class. If none of the specified
540 authentication context classes can be satisfied, the identity provider MUST not include an
541 authentication statement in the resulting assertion.
- 542 • If `<AuthnContextMinimumClassRef>` is specified, then the resulting authentication
543 statement in the assertion (if any) MUST have an authentication statement that is deemed to
544 be at least as strong as the stated minimum class, as deemed by the identity provider. If the
545 minimum class cannot be satisfied, the identity provider MUST not include an authentication
546 statement in the resulting assertion.
- 547 • If one or more `<AuthnContextStatementRef>` elements are included, then the resulting
548 authentication statement in the assertion (if any) MUST have an authentication statement that
549 exactly matches one of the specified authentication context statement(s). If this requirement
550 cannot be satisfied, the identity provider MUST not include an authentication statement in
551 the resulting assertion.

552 If an identity is being federated, the identity provider MUST adhere to the following rules in
553 generating the name identifier:

- 554 • The name identifier MUST be unique across all Principals in the scope of the service
555 provider-identity provider pairwise relationship.
- 556 • The name identifier for the specific Principal MUST be unique across all service providers
557 with which an identity federation exists with the identity provider.

558 Failure to either authenticate the Principal and/or federate the identity is indicated by a status code
559 other than “Success.” For failures, assertions MUST NOT appear in the `<AuthnResponse>`.

560 In some profiles, an intermediary is active between the service provider’s authentication request and
561 the identity provider’s authentication response. If an error occurs in the processing at the
562 intermediary, the following status code values are defined for the `<saml:Status>` element:

- 563 • `lib:NoAvailableIDP`: Used to indicate that none of the identity provider URLs from the
564 `<IDPList>` can be resolved or that none of the identity providers is available.
- 565 • `lib:NoSupportedIDP`: Used to indicate that none of the identity providers is supported by
566 the intermediary.

567 **3.2.4 Request Envelope**

568 Some profiles MAY wrap the `<AuthnRequest>` element in an envelope. This envelope allows for
569 extra processing by an intermediary between the service provider and the identity provider. An
570 example of an intermediary is a user agent or proxy.

571 **3.2.4.1 Element `<AuthnRequestEnvelope>`**

572 The authentication request envelope contains the following elements:

573 `AuthnRequest` [Required]

574 The enveloped authentication request.

575 ProviderID [Required]

576 The ProviderID of the requestor.

577 ProviderName [Optional]

578 The human-readable name of the requestor.

579 AssertionConsumerServiceURL [Required]

580 The service provider's URL where the authentication response should be sent.

581 IDPList [Optional]

582 A list of identity providers from which a provider may be chosen to service the authentication
583 request.

584 IsPassive [Optional]

585 If "true," specifies that any intermediary between the service provider and identity provider
586 MUST NOT interact with the Principal. If not specified, "true" is presumed.

587 The schema fragment is as follows:

```
588 <element name="AuthnRequestEnvelope" type="lib:AuthnRequestEnvelopeType"/>
589 <complexType name="AuthnRequestEnvelopeType">
590 <complexContent>
591 <extension base="lib:RequestEnvelopeType">
592 <sequence>
593 <element ref="lib:AuthnRequest"/>
594 <element ref="lib:ProviderID"/>
595 <element name="ProviderName" type="string" minOccurs="0"/>
596 <element name="AssertionConsumerServiceURL" type="anyURI"/>
597 <element ref="lib:IDPList" minOccurs="0"/>
598 <element name="IsPassive" type="boolean" minOccurs="0"/>
599 </sequence>
600 </extension>
601 </complexContent>
602 </complexType>
603 <complexType name="RequestEnvelopeType">
604 <sequence>
605 <any processContents="skip" minOccurs="0" maxOccurs="unbounded"/>
606 </sequence>
607 </complexType>
```

608 3.2.4.2 Element <IDPList>

609 In the request envelope, some profiles may wish to allow the service provider to transport a list of
610 identity providers to the user agent. This specification provides a schema that profiles SHOULD use
611 for this purpose. The elements are as follows:

612 IDPList

613 The container element for an IDP List.

614 IDPEntries

615 Contains a list of identity provider entries.

616 IDPEntry

617 Describes an identity provider that the service provider supports.

618 ProviderID

619 The identity provider's ProviderID.

620 ProviderName

621 The identity provider's human-readable name.

622 Loc
623 The URI at which the identity provider can be contacted for sending the authentication
624 request.

625 GetComplete
626 If the identity provider list is not complete, this element is included with a URI that points to
627 where the complete list can be retrieved.

628 The schema fragment is as follows:

```

629 <element name="IDPList" type="lib:IDPListType"/>
630 <complexType name="IDPListType">
631   <sequence>
632     <element ref="lib:IDPEntries"/>
633     <element ref="lib:GetComplete" minOccurs="0" maxOccurs="unbounded"/>
634   </sequence>
635 </complexType>
636 <element name="IDPEntity">
637   <complexType>
638     <sequence>
639       <element ref="lib:ProviderID"/>
640       <element name="ProviderName" type="string" minOccurs="0"/>
641       <element name="Loc" type="anyURI"/>
642     </sequence>
643   </complexType>
644 </element>
645 <element name="IDPEntries">
646   <complexType>
647     <sequence>
648       <element ref="lib:IDPEntity" maxOccurs="unbounded"/>
649     </sequence>
650   </complexType>
651 </element>
652 <element name="GetComplete" type="anyURI"/>

```

653 3.2.4.3 Example

```

654 <AuthnRequestEnvelope>
655   <AuthnRequest> ... </AuthnRequest>
656   <ProviderID>http://ServiceProvider.com</ProviderID>
657   <ProviderName>Service Provider X</ProviderName>
658   <AssertionConsumerServiceURL>http://ServiceProvider.com/lecp_assertion_consumer</AssertionConsum
659   rServiceURL>
660   <IDPList>
661     <IDPEntries>
662       <IDPEntity>
663         <ProviderID>http://IdentityProvider.com</ProviderID>
664         <ProviderName>Identity Provider X</ProviderName>
665         <Loc>http://www.IdentityProvider.com/liberty/sso</Loc>
666       </IDPEntity>
667     </IDPEntries>
668     <GetComplete>https://ServiceProvider.com/idplist?id=604be136-fe91-441e-afb8-f88748ae3b8b
669   </GetComplete>
670   </IDPList>
671   <IsPassive>0</IsPassive>
672 </AuthnRequestEnvelope>

```

673 3.2.5 Response Envelope

674 As with the <AuthnRequest> element, some profiles MAY wrap the <AuthnResponse> element
675 in an envelope. This envelope allows for extra processing by an intermediary between the identity
676 provider and the service provider. An example of an intermediary is a user agent or proxy.

677 3.2.5.1 Element <AuthnResponseEnvelope>

678 The authentication response envelope contains a single <AuthnResponse> element, along with any
679 number of arbitrary elements. The schema fragment is as follows:

```
680 <element name="AuthnResponseEnvelope" type="lib:AuthnResponseEnvelopeType"/>
681 <complexType name="AuthnResponseEnvelope">
682   <complexContent>
683     <extension base="lib:ResponseEnvelopeType">
684       <sequence>
685         <element ref="lib:AuthnResponse"/>
686       </sequence>
687     </extension>
688   </complexContent>
689 </complexType>
690 <complexType name="ResponseEnvelopeType">
691   <sequence>
692     <any processContents="skip" minOccurs="0" maxOccurs="unbounded"/>
693   </sequence>
694 </complexType>
```

695 3.2.5.2 Example

```
696 <AuthnResponseEnvelope>
697   <AuthnResponse> ... </AuthnResponse>
698   <idp:Foo></idp:Foo>
699 </AuthnResponseEnvelope>
```

700 3.3 Name Registration Protocol

701 At the time of federation, the identity provider generates an opaque handle that serves as the name
702 identifier the service provider and the identity provider use in referring to the Principal when
703 communicating with each other. This name identifier is termed the
704 IDPProvidedNameIdentifier. Subsequent to federation, however, the service provider MAY
705 register a different opaque handle with the identity provider. This opaque handle is termed the
706 SPPProvidedNameIdentifier.

707 Subsequent to name registration, the identity provider MUST use the
708 SPPProvidedNameIdentifier for <saml:NameIdentifier> elements when communicating to
709 the service provider about the Principal. The service provider MUST continue using the
710 IDPProvidedNameIdentifier for <saml:NameIdentifier> elements when communicating
711 to the identity provider about the Principal.

712 The service provider MAY register a new name identifier for a Principal with the identity provider at
713 any time subsequent to federation.

714 The name identifier specified by the service provider SHOULD adhere to the following guidelines:

- 715 • The name identifier SHOULD be unique across the identity providers with which the
716 Principal's identity is federated.
- 717 • The name identifier SHOULD be unique across all name identifiers that have been registered
718 with the identity provider by this service provider.

719 3.3.1 Request

720 To register a SPPProvidedNameIdentifier with an identity provider, the service provider sends a
721 <RegisterNameIdentifierRequest> message.

722 3.3.1.1 Element <RegisterNameIdentifierRequest>

723 The elements of the message are as follows:

724 ProviderID [Required]

725 The service provider's identifier.

726 IDPProvidedNameIdentifier [Required]

727 The name identifier the identity provider provided at the time of federation.

728 SPPProvidedNameIdentifier [Required]

729 The name identifier the identity provider will use when communicating to the service
730 provider.

731 The schema fragment is as follows:

```
732 <element name="RegisterNameIdentifierRequest" type="lib:RegisterNameIdentifierRequestType"/>
733 <complexType name="RegisterNameIdentifierRequestType">
734 <complexContent>
735 <extension base="samlp:RequestAbstractType">
736 <sequence>
737 <element ref="lib:ProviderID"/>
738 <element name="IDPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
739 <element name="SPPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
740 </sequence>
741 </extension>
742 </complexContent>
743 </complexType>
744 <element name="IDPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
745 <element name="SPPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
```

746 3.3.1.2 Example

```
747 <RegisterNameIdentifierRequest RequestID="eb20e77f-d982-44f9-936e-dd135bf437d4" MajorVersion="1"
748 MinorVersion="0" IssueInstant="2001-12-17T09:30:47-05:00">
749 <ds:Signature>...</ds:Signature>
750 <ProviderID>http://ServiceProvider.com</ProviderID>
751 <IDPProvidedNameIdentifier>342ad3d8-93ee-4c68-be35-cc9e7db39e2b</IDPProvidedNameIdentifier>
752 <SPPProvidedNameIdentifier>e958019a</SPPProvidedNameIdentifier>
753 </RegisterNameIdentifierRequest>
```

754 3.3.2 Response

755 The identity provider MUST respond with <RegisterNameIdentifierResponse>, which is of
756 type **StatusResponseType**. **StatusResponseType** is an extension of **samlp:ResponseAbstractType**
757 so that only a <samlp:Status> element exists in the body.

758 3.3.2.1 Element <RegisterNameIdentifierResponse>

759 The elements of the message are as follows:

760 Status [Required]

761 The status of the request processing.

762 The schema fragment is as follows:

```
763 <element name="RegisterNameIdentifierResponse" type="lib:StatusResponseType"/>
764 <complexType name="StatusResponseType">
765 <complexContent>
766 <extension base="samlp:ResponseAbstractType">
767 <sequence>
768 <element ref="samlp:Status"/>
769 </sequence>
770 </extension>
771 </complexContent>
772 </complexType>
```

773 3.3.2.2 Example

```
774 <RegisterNameIdentifierResponse ResponseID="74ffec0f-1165-4fa3-b088-3dd2c2388b91"
775 InResponseTo="eb20e77f-d982-44f9-936e-dd135bf437d4" MajorVersion="1" MinorVersion="0"
776 IssueInstant="2001-12-17T09:30:47-05:00" Recipient="http://ServiceProvider.com">
777 <ds:Signature>...</ds:Signature>
778 <samlp:Status>
```

```
779 <samlp:StatusCode Value="saml:Success"/>
780 </samlp:Status>
781 </RegisterNameIdentifierResponse>
```

782 3.3.3 Processing Rules

783 The identity provider MUST validate the signature on the message. To be considered valid, the
784 signature on the message MUST be the signature of the <ProviderID> contained in the message.

785 If the request includes an <IDPProvidedNameIdentifier> for which no federation exists
786 between the service provider and the identity provider, the Provider MUST respond with a
787 <saml:Status> with a <saml:StatusCode> of lib:FederationDoesNotExist.

788 Otherwise, the identity provider MUST use <SPProvidedNameIdentifier> when subsequently
789 communicating to the service provider regarding this Principal.

790 3.4 Federation Termination Notification Protocol

791 When the Principal terminates an identity federation between a service provider and an identity
792 provider from the service provider, the service provider MUST send a
793 <FederationTerminationNotification> message to the identity provider. Semantically, the
794 service provider is stating that it will no longer accept authentication assertions from the identity
795 provider for the specified Principal.

796 Likewise, when the Principal terminates an identity federation from the identity provider, the identity
797 provider MUST send a <FederationTerminationNotification> message to the service
798 provider. Semantically, the identity provider is stating that it will no longer provide authentication
799 assertions to the service provider for the specified Principal.

800 This notification message is a one-way asynchronous message, and reasonable best-effort delivery
801 MUST be employed by both service providers and identity providers sending the message. The
802 message MUST be signed.

803 3.4.1 Message

804 The provider sends a <FederationTerminationNotification> to the provider with which it is
805 terminating a federation.

806 3.4.1.1 Element <FederationTerminationNotification>

807 The elements are as follows:

808 ProviderID [Required]

809 The identifier of the provider that is sending the notification of federation termination.

810 NameIdentifier [Required]

811 The name identifier of the Principal to which the notification pertains.

812 The schema fragment is as follows:

```
813 <element name="FederationTerminationNotification"
814 type="lib:FederationTerminationNotificationType"/>
815 <complexType name="FederationTerminationNotificationType">
816 <complexContent>
817 <extension base="samlp:RequestAbstractType">
818 <sequence>
819 <element ref="lib:ProviderID"/>
820 <element ref="saml:NameIdentifier"/>
821 </sequence>
```

822
823
824

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

825 3.4.1.2 Example

826
827
828
829
830
831

```
<FederationTerminationNotification RequestID="9ec2-eb65-4bce-ab8f-4becdf229815" MajorVersion="1"  
MinorVersion="0" IssueInstant="2001-12-17T09:30:47-05:00">  
  <ds:Signature>...</ds:Signature>  
  <ProviderID>http://IdentityProvider.com</ProviderID>  
  <saml:NameIdentifier>e958019a</saml:NameIdentifier>  
</FederationTerminationNotification>
```

832 3.4.2 Processing Rules

833 The receiving provider MUST validate the signature on the message. The signature on the message
834 MUST be the signature of the <ProviderID> contained in the message. If the signature is not valid,
835 the provider MUST ignore the message.

836 If a provider receives a federation termination notification message that refers to a federation that
837 does not exist from the perspective of the provider, the provider MUST ignore the message.
838 Otherwise, the provider MAY perform any maintenance with the knowledge that the federation has
839 been terminated.

840 3.5 Single Logout Protocol

841 The Single Logout Protocol provides a one-way asynchronous message exchange protocol by which
842 all sessions authenticated by a particular identity provider are near-simultaneously terminated. The
843 Single Logout Protocol is used either when a Principal logs out at a service provider or when the
844 Principal logs out at an identity provider.

845 When the Principal invokes the single logout process at a service provider, the service provider
846 MUST send a <LogoutNotification> message to the identity provider that provided the
847 authentication service for the session.

848 When either the Principal invokes a logout at the identity provider or a service provider sends a
849 logout notification to the identity provider specifying that Principal, the identity provider MUST
850 send a <LogoutNotification> message to each service provider to which it provided
851 authentication assertions in the current session with the Principal, with the exception of the service
852 provider that sent the <LogoutNotification> message to the Identity Provider.

853 This notification is a one-way asynchronous message, and reasonable best-effort delivery SHOULD
854 be employed by both service providers and identity providers.

855 This message SHOULD be signed.

856 3.5.1 Message

857 The <LogoutNotification> message indicates to the message receiver that a Principal's session
858 was terminated. The message includes an optional <SessionIndex> element that MUST be
859 specified if and only if the authentication statement in the assertion that the service provider used in
860 establishing the session with the Principal contained a SessionIndex attribute.

861 3.5.1.1 Element <LogoutNotification>

862 NameIdentifier [Required]

863 The name identifier of the Principal that logged out.

864 ProviderID [Required]

865 The identifier of the provider that is making the request.

866 SessionIndex [Optional]

867 The session index specified in the authentication statement in the assertion that was used in
868 establishing the session being terminated.

869 The schema fragment is as follows:

```
870 <element name="LogoutNotification" type="lib:LogoutNotificationType"/>
871 <complexType name="LogoutNotificationType">
872   <complexContent>
873     <extension base="samlp:RequestAbstractType">
874       <sequence>
875         <element ref="lib:ProviderID"/>
876         <element ref="saml:NameIdentifier"/>
877         <element name="SessionIndex" type="string" minOccurs="0"/>
878       </sequence>
879     </extension>
880   </complexContent>
881 </complexType>
```

882 3.5.1.2 Example

```
883 <LogoutNotification RequestID="47693d03-7c33-4d65-931f-ddeb19fa6a73" MajorVersion="1"
884 MinorVersion="0" IssueInstant="2001-12-17T09:30:47-05:00">
885   <ds:Signature>...</ds:Signature>
886   <ProviderID>http://ServiceProvider.com</ProviderID>
887   <saml:NameIdentifier>342ad3d8-93ee-4c68-be35-cc9e7db39e2b</saml:NameIdentifier>
888   <SessionIndex>3</SessionIndex>
889 </LogoutNotification>
```

890 3.5.2 Processing Rules

891 Unique processing rules apply based on whether the message receiver is an identity provider or a
892 service provider.

893 3.5.2.1 Identity Provider Processing Rules

894 When an identity provider receives the <LogoutNotification> message, the identity provider
895 MUST validate that the signature on the message is the signature of a service provider to which the
896 identity provider provided an authentication assertion for the current session. If that holds, the
897 identity provider SHOULD do the following:

- 898 • Send a <LogoutNotification> message to each service providers for which the identity
899 provider provided authentication assertions in the current session.
- 900 • Terminate the Principal's current session as specified by the <saml:NameIdentifier>
901 element.

902 3.5.2.2 Service Provider Processing Rules

903 When the service provider receives the <LogoutNotification> message, the service provider
904 MUST validate the identity provider's signature contained in the <ds:Signature> element. If the
905 signature is that of the identity provider that provided the authentication for the Principal's current
906 session, the service provider MUST invalidate the Principal's session referred to in the
907 <saml:NameIdentifier> element.

908 4 Provider Metadata Schema

909 For providers to communicate with each other, they must a priori have obtained metadata regarding
910 each other. These provider metadata include items such as X.509 certificates and service endpoints.
911 This specification defines metadata schemas for identity providers and service providers that may be
912 used for provider metadata exchange. However, provider metadata exchange protocols are outside
913 the scope of this specification.

914 4.1 Generic Provider Descriptor

915 Certain provider metadata are generic to both service providers and identity providers. The complex
916 type **ProviderDescriptorType** contains the following elements:

917 **ProviderID** [Required]

918 The provider's URI-based identifier.

919 **ProviderSuccinctID** [Required]

920 The provider's succinct identifier.

921 **KeyInfo** [Optional]

922 The provider's public key.

923 **SoapEndpoint** [Optional]

924 The provider's SOAP endpoint URI.

925 **SingleLogoutServiceURL** [Optional]

926 The URL used for user-agent-based Single Logout Protocol profiles.

927 **SingleLogoutServiceReturnURL** [Optional]

928 The URL to which the provider redirects at the end of user-agent-based Single Logout
929 Protocol profiles.

930 **FederationTerminationServiceURL** [Optional]

931 The URL used for user-agent-based Federation Termination Notification Protocol profiles.

932 **FederationTerminationServiceReturnURL** [Optional]

933 The URL to which the provider redirects at the end of user-agent-based Federation
934 Termination Notification Protocol profiles.

935 The schema fragment is as follows:

```
936 <complexType name="ProviderDescriptorType">  
937   <sequence>  
938     <element name="ProviderID" type="anyURI"/>  
939     <element name="ProviderSuccinctID" type="hexBinary"/>  
940     <element ref="ds:KeyInfo" minOccurs="0"/>  
941     <element name="SoapEndpoint" type="anyURI" minOccurs="0"/>  
942     <element name="SingleLogoutServiceURL" type="anyURI" minOccurs="0"/>  
943     <element name="SingleLogoutServiceReturnURL" type="anyURI" minOccurs="0"/>  
944     <element name="FederationTerminationServiceURL" type="anyURI" minOccurs="0"/>  
945     <element name="FederationTerminationServiceReturnURL" type="anyURI" minOccurs="0"/>  
946   </sequence>  
947 </complexType>
```

948 4.2 Service Provider Descriptor

949 The additional service provider-specific metadata are as follows:

950 AssertionConsumerServiceURL [Required]

951 The service provider's URL for consuming assertions from identity providers.

952 FederationTerminationNotificationProtocolProfile [Required]

953 The Federation Termination Notification Protocol profile that the identity provider should use
954 when communicating with the service provider.

955 SingleLogoutProtocolProfile [Required]

956 The Single Logout Protocol profile that the identity provider should use when communicating
957 with the service provider.

958 AuthnRequestsSigned [Required]

959 Specifies whether the service provider will always sign authentication requests it sends to the
960 identity provider.

961 The schema fragment is as follows:

```
962 <element name="SPDescriptor" type="lib:SPDescriptorType"/>
963 <complexType name="SPDescriptorType">
964   <complexContent>
965     <extension base="lib:ProviderDescriptorType">
966       <sequence>
967         <element name="AssertionConsumerServiceURL" type="anyURI"/>
968         <element name="FederationTerminationNotificationProtocolProfile" type="anyURI"/>
969         <element name="SingleLogoutProtocolProfile" type="anyURI"/>
970         <element name="AuthnRequestsSigned" type="boolean"/>
971       </sequence>
972     </extension>
973   </complexContent>
974 </complexType>
```

975 4.2.1 Example

```
976 <SPDescriptor>
977   <ProviderID>http://ServiceProvider.com</ProviderID>
978   <ProviderSuccinctID>A9FD64E12C</ProviderSuccinctID>
979   <ds:KeyInfo>...</ds:KeyInfo>
980   <SoapEndpoint>http://ServiceProvider.com/soap</SoapEndpoint>
981   <SingleLogoutServiceURL>http://ServiceProvider.com/liberty/slo</SingleLogoutServiceURL>
982   <SingleLogoutServiceReturnURL>http://ServiceProvider.com/liberty/slo_return</SingleLogoutServiceR
983   eturnURL>
984   <FederationTerminationServiceURL>http://ServiceProvider.com/liberty/term</FederationTerminationSe
985   rviceURL>
986   <FederationTerminationServiceReturnURL>http://ServiceProvider.com/liberty/term_return</Federation
987   TerminationServiceReturnURL>
988   <AssertionConsumerServiceURL>http://ServiceProvider.com/liberty/assertion_consumer</AssertionCons
989   umerServiceURL>
990   <FederationTerminationNotificationProtocolProfile>http://projectliberty.org/profiles/fedterm_soap
991 </FederationTerminationNotificationProtocolProfile>
992   <SingleLogoutProtocolProfile>http://projectliberty.org/profiles/slo_soap</SingleLogoutProtocolPro
993   file>
994   <AuthnRequestsSigned>1</AuthnRequestsSigned>
995 </SPDescriptor>
```

996 4.3 Identity Provider Descriptor

997 The additional identity provider-specific metadata are as follows:

998 SingleSignOnServiceURL [Required]

999 The identity provider's URL for accepting authentication requests for the Single Sign-On and
1000 Federation Protocol.

1001 The schema fragment is as follows:

```
1002 <element name="IDPDescriptor" type="lib:IDPDescriptorType"/>
```

```

1003 <complexType name="IDPDescriptorType">
1004 <complexContent>
1005 <extension base="lib:ProviderDescriptorType">
1006 <sequence>
1007 <element name="SingleSignOnServiceURL" type="anyURI"/>
1008 </sequence>
1009 </extension>
1010 </complexContent>
1011 </complexType>

```

4.3.1 Example

```

1013 <IDPDescriptor>
1014 <ProviderID>http://IdentityProvider.com</ProviderID>
1015 <ProviderSuccinctID>A9FD64E12C</ProviderSuccinctID>
1016 <ds:KeyInfo>...</ds:KeyInfo>
1017 <SoapEndpoint>http://IdentityProvider.com/soap</SoapEndpoint>
1018 <SingleLogoutServiceURL>http://IdentityProvider.com/liberty/slo</SingleLogoutServiceURL>
1019 <SingleLogoutServiceReturnURL>http://IdentityProvider.com/liberty/slo_return</SingleLogoutService
1020 ReturnURL>
1021 <FederationTerminationServiceURL>http://IdentityProvider.com/liberty/term</FederationTerminationS
1022 erviceURL>
1023 <FederationTerminationServiceReturnURL>http://IdentityProvider.com/liberty/term_return</Federatio
1024 nTerminationServiceReturnURL>
1025 <SingleSignOnServiceURL>http://IdentityProvider.com/liberty/sso</SingleSignOnServiceURL>
1026 </IDPDescriptor>

```

5 Schema Definition

```

1028 <?xml version="1.0" encoding="UTF-8"?>
1029 <schema targetNamespace="http://www.projectliberty.org/schemas/core/2002/05"
1030 xmlns:lib="http://www.projectliberty.org/schemas/core/2002/05"
1031 xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
1032 xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
1033 xmlns:ac="http://www.projectliberty.org/schemas/authctx/2002/05"
1034 xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
1035 attributeFormDefault="unqualified">
1036 <import namespace="urn:oasis:names:tc:SAML:1.0:assertion" schemaLocation="http://www.oasis-
1037 open.org/committees/security/docs/draft-sstc-schema-assertion-31.xsd"/>
1038 <import namespace="urn:oasis:names:tc:SAML:1.0:protocol" schemaLocation="http://www.oasis-
1039 open.org/committees/security/docs/draft-sstc-schema-protocol-31.xsd"/>
1040 <import namespace="http://www.w3.org/2000/09/xmldsig#"
1041 schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd"/>
1042 <import namespace="http://www.projectliberty.org/schemas/authctx/2002/05" schemaLocation="draft-
1043 liberty-architecture-authentication-context-06.xsd"/>
1044 <!-- Begin protocols schema -->
1045 <element name="ProviderID" type="anyURI"/>
1046 <element name="AuthnRequest" type="lib:AuthnRequestType"/>
1047 <complexType name="AuthnRequestType">
1048 <complexContent>
1049 <extension base="samlp:RequestAbstractType">
1050 <sequence>
1051 <element ref="lib:ProviderID"/>
1052 <element name="ForceAuthn" type="boolean" minOccurs="0"/>
1053 <element name="IsPassive" type="boolean" minOccurs="0"/>
1054 <element name="Federate" type="boolean" minOccurs="0"/>
1055 <element ref="lib:ProtocolProfile" minOccurs="0"/>
1056 <element ref="lib:AuthnContext" minOccurs="0"/>
1057 <element ref="lib:RelayState" minOccurs="0"/>
1058 </sequence>
1059 </extension>
1060 </complexContent>
1061 </complexType>
1062 <element name="RelayState"/>
1063 <element name="ProtocolProfile" type="anyURI"/>
1064 <element name="AuthnContext">
1065 <complexType>
1066 <choice>
1067 <element name="AuthnContextClassRef" type="anyURI" maxOccurs="unbounded"/>
1068 <element name="AuthnContextStatementRef" type="anyURI" maxOccurs="unbounded"/>
1069 <element name="AuthnContextMinimumClassRef" type="anyURI"/>
1070 </choice>
1071 </complexType>
1072 </element>
1073 <element name="AuthnRequestEnvelope" type="lib:AuthnRequestEnvelopeType"/>

```

```

1074 <complexType name="AuthnRequestEnvelopeType">
1075 <complexContent>
1076 <extension base="lib:RequestEnvelopeType">
1077 <sequence>
1078 <element ref="lib:AuthnRequest"/>
1079 <element ref="lib:ProviderID"/>
1080 <element name="ProviderName" type="string" minOccurs="0"/>
1081 <element name="AssertionConsumerServiceURL" type="anyURI"/>
1082 <element ref="lib:IDPList" minOccurs="0"/>
1083 <element name="IsPassive" type="boolean" minOccurs="0"/>
1084 </sequence>
1085 </extension>
1086 </complexContent>
1087 </complexType>
1088 <complexType name="RequestEnvelopeType">
1089 <sequence>
1090 <any processContents="skip" minOccurs="0" maxOccurs="unbounded"/>
1091 </sequence>
1092 </complexType>
1093 <element name="AuthnResponseEnvelope" type="lib:AuthnResponseEnvelopeType"/>
1094 <complexType name="AuthnResponseEnvelopeType">
1095 <complexContent>
1096 <extension base="lib:ResponseEnvelopeType">
1097 <sequence>
1098 <element ref="lib:AuthnResponse"/>
1099 </sequence>
1100 </extension>
1101 </complexContent>
1102 </complexType>
1103 <complexType name="ResponseEnvelopeType">
1104 <sequence>
1105 <any processContents="skip" minOccurs="0" maxOccurs="unbounded"/>
1106 </sequence>
1107 </complexType>
1108 <complexType name="SubjectType">
1109 <complexContent>
1110 <extension base="saml:SubjectType">
1111 <sequence>
1112 <element ref="lib:IDPProvidedNameIdentifier"/>
1113 </sequence>
1114 </extension>
1115 </complexContent>
1116 </complexType>
1117 <element name="AuthnResponse" type="lib:AuthnResponseType"/>
1118 <complexType name="AuthnResponseType">
1119 <complexContent>
1120 <extension base="samlp:ResponseType">
1121 <sequence>
1122 <element ref="lib:RelayState" minOccurs="0"/>
1123 </sequence>
1124 </extension>
1125 </complexContent>
1126 </complexType>
1127 <complexType name="AuthenticationStatementType">
1128 <complexContent>
1129 <extension base="saml:AuthenticationStatementType">
1130 <sequence>
1131 <element name="AuthnContext" minOccurs="0">
1132 <complexType>
1133 <sequence>
1134 <element name="AuthnContextClassRef" type="anyURI" minOccurs="0"/>
1135 <choice>
1136 <element ref="ac:AuthenticationContextStatement"/>
1137 <element name="AuthnContextStatementRef" type="anyURI"/>
1138 </choice>
1139 </sequence>
1140 </complexType>
1141 </element>
1142 </sequence>
1143 <attribute name="ReauthenticateOnOrAfter" type="dateTime" use="optional"/>
1144 <attribute name="SessionIndex" type="string" use="optional"/>
1145 </extension>
1146 </complexContent>
1147 </complexType>
1148 <element name="RegisterNameIdentifierRequest" type="lib:RegisterNameIdentifierRequestType"/>
1149 <complexType name="RegisterNameIdentifierRequestType">

```

```

1150     <complexContent>
1151       <extension base="sampl:RequestAbstractType">
1152         <sequence>
1153           <element ref="lib:ProviderID"/>
1154           <element name="IDPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
1155           <element name="SPPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
1156         </sequence>
1157       </extension>
1158     </complexContent>
1159   </complexType>
1160   <element name="IDPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
1161   <element name="SPPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
1162   <element name="RegisterNameIdentifierResponse" type="lib:StatusResponseType"/>
1163   <complexType name="StatusResponseType">
1164     <complexContent>
1165       <extension base="sampl:ResponseAbstractType">
1166         <sequence>
1167           <element ref="sampl:Status"/>
1168         </sequence>
1169       </extension>
1170     </complexContent>
1171   </complexType>
1172   <element name="FederationTerminationNotification"
1173 type="lib:FederationTerminationNotificationType"/>
1174   <complexType name="FederationTerminationNotificationType">
1175     <complexContent>
1176       <extension base="sampl:RequestAbstractType">
1177         <sequence>
1178           <element ref="lib:ProviderID"/>
1179           <element ref="saml:NameIdentifier"/>
1180         </sequence>
1181       </extension>
1182     </complexContent>
1183   </complexType>
1184   <element name="LogoutNotification" type="lib:LogoutNotificationType"/>
1185   <complexType name="LogoutNotificationType">
1186     <complexContent>
1187       <extension base="sampl:RequestAbstractType">
1188         <sequence>
1189           <element ref="lib:ProviderID"/>
1190           <element ref="saml:NameIdentifier"/>
1191           <element name="SessionIndex" type="string" minOccurs="0"/>
1192         </sequence>
1193       </extension>
1194     </complexContent>
1195   </complexType>
1196   <!-- End protocols schema -->
1197   <!-- Begin assertion schema -->
1198   <element name="Assertion" type="lib:AssertionType"/>
1199   <complexType name="AssertionType">
1200     <complexContent>
1201       <extension base="saml:AssertionType">
1202         <attribute name="InResponseTo" type="saml:IDReferenceType"/>
1203       </extension>
1204     </complexContent>
1205   </complexType>
1206   <!-- End assertion schema -->
1207   <!-- Begin IDP list schema -->
1208   <element name="IDPList" type="lib:IDPListType"/>
1209   <complexType name="IDPListType">
1210     <sequence>
1211       <element ref="lib:IDPEntries"/>
1212       <element ref="lib:GetComplete" minOccurs="0" maxOccurs="unbounded"/>
1213     </sequence>
1214   </complexType>
1215   <element name="IDPEntry">
1216     <complexType>
1217       <sequence>
1218         <element ref="lib:ProviderID"/>
1219         <element name="ProviderName" type="string" minOccurs="0"/>
1220         <element name="Loc" type="anyURI"/>
1221       </sequence>
1222     </complexType>
1223   </element>
1224   <element name="IDPEntries">
1225     <complexType>

```

```

1226     <sequence>
1227         <element ref="lib:IDPEntry" maxOccurs="unbounded"/>
1228     </sequence>
1229 </complexType>
1230 </element>
1231 <element name="GetComplete" type="anyURI"/>
1232 <!-- End IDP list schema -->
1233 <!-- Begin provider metadata schema -->
1234 <complexType name="ProviderDescriptorType">
1235     <sequence>
1236         <element name="ProviderID" type="anyURI"/>
1237         <element name="ProviderSuccinctID" type="hexBinary"/>
1238         <element ref="ds:KeyInfo" minOccurs="0"/>
1239         <element name="SoapEndpoint" type="anyURI" minOccurs="0"/>
1240         <element name="SingleLogoutServiceURL" type="anyURI" minOccurs="0"/>
1241         <element name="SingleLogoutServiceReturnURL" type="anyURI" minOccurs="0"/>
1242         <element name="FederationTerminationServiceURL" type="anyURI" minOccurs="0"/>
1243         <element name="FederationTerminationServiceReturnURL" type="anyURI" minOccurs="0"/>
1244     </sequence>
1245 </complexType>
1246 <element name="SPDescriptor" type="lib:SPDescriptorType"/>
1247 <complexType name="SPDescriptorType">
1248     <complexContent>
1249         <extension base="lib:ProviderDescriptorType">
1250             <sequence>
1251                 <element name="AssertionConsumerServiceURL" type="anyURI"/>
1252                 <element name="FederationTerminationNotificationProtocolProfile" type="anyURI"/>
1253                 <element name="SingleLogoutProtocolProfile" type="anyURI"/>
1254                 <element name="AuthnRequestsSigned" type="boolean"/>
1255             </sequence>
1256         </extension>
1257     </complexContent>
1258 </complexType>
1259 <element name="IDPDescriptor" type="lib:IDPDescriptorType"/>
1260 <complexType name="IDPDescriptorType">
1261     <complexContent>
1262         <extension base="lib:ProviderDescriptorType">
1263             <sequence>
1264                 <element name="SingleSignOnServiceURL" type="anyURI"/>
1265             </sequence>
1266         </extension>
1267     </complexContent>
1268 </complexType>
1269 <!-- End provider metadata schema -->
1270 </schema>

```

1271 6 References

- 1272 [LibertyBindProf] Rouault, J., "Liberty Bindings and Profiles Specification."
- 1273 [LibertyGloss] Mauldin, H., "Liberty Glossary," [https://66.34.4.93/members/technology](https://66.34.4.93/members/technology_expert_group/draft-liberty-tech-glossary-01.doc)
1274 [expert group/draft-liberty-tech-glossary-01.doc](https://66.34.4.93/members/technology_expert_group/draft-liberty-tech-glossary-01.doc), April 2002.
- 1275 [RFC2119] S. Bradner, Key words for use in RFCs to Indicate Requirement Levels,
1276 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- 1277 [RFC3280] R. Housley, Internet X.509 Public Key Infrastructure Certificate and
1278 Certificate Revocation List (CRL) Profile,
1279 <http://www.ietf.org/rfc/rfc3280.txt>, IETF RFC 3280, April 2002.
- 1280 [SAMLCore] Hallam-Baker, P. et al., Assertions and Protocol for the OASIS Security
1281 Assertion Markup Language (SAML), [http://www.oasis-](http://www.oasis-open.org/committees/security/docs/draft-sstc-core-27.pdf)
1282 [open.org/committees/security/docs/draft-sstc-core-27.pdf](http://www.oasis-open.org/committees/security/docs/draft-sstc-core-27.pdf), OASIS,
1283 December 2001.
- 1284 [Schema1] H. S. Thompson et al., "XML Schema Part 1: Structures,"
1285 <http://www.w3.org/TR/xmlschema-1/>, World Wide Web Consortium
1286 Recommendation, May 2001.